Metropolia Ammattikorkeakoulu
Elektroniikan koulutusohjelma

**Juha Vaaksio**

# Ultraäänipesurin suunnittelu

Insinöörityö 25.4.2010

Ohjaaja: lehtori Janne Mäntykoski
Ohjaava opettaja: lehtori Janne Mäntykoski

**Metropolia Ammattikorkeakoulu Insinöörityön tiivistelmä**

| | |
|---|---|
| Tekijä<br>Otsikko<br><br>Sivumäärä<br>Aika | Juha Vaaksio<br>Ultraäänipesurin suunnittelu<br><br>45 sivua<br>25.4.2010 |
| Koulutusohjelma | elektroniikan koulutusohjelma |
| Tutkinto | insinööri (AMK) |
| Ohjaaja<br>Ohjaava opettaja | lehtori Janne Mäntykoski<br>lehtori Janne Mäntykoski |


Insinöörityössä oli tavoitteena suunnitella pienen ultraäänipesurin käytännön toteutus. Pesuri päätettiin toteuttaa mikroprosessorin ympärille ja ultraäänikaiuttimien ohjaus toteuttaa puolikkaan H-sillan avulla.

Sopivien ratkaisujen löytämiseksi tutkittiin kirjallisia lähteitä ja vertailtiin eri vaihtoehtoja suorittaa käytännön toteutus.

Mikroprosessorille kirjoitettiin koodi, jonka avulla saatiin aikaan pesuun tarvittava noin 40 kilohertsin taajuus. Mikroprosessorilla hoidettiin myös muu tarpeellinen toiminta, kuten pesuajan lukeminen ja näkyville tulostaminen ja sen laskeminen ja vähentäminen.

Kaiuttimille mietittiin sopivaa ohjaustapaa ja tutkittiin vaihtoehtoja. Ohjaus päädyttiin toteuttamaan valmiin ajuripiirin avulla.

Tuloksissa tutkittiin kaiuttimien ohjauksen toimimista valitulla tavalla ja todettiin valitun ohjaustavan soveltuvan jännitteen ohjaamiseen.


| | |
|---|---|
| Hakusanat | ultraäänet, ultraäänillä puhdistaminen, ultraäänien tuottaminen |

**Helsinki Metropolia University of Applied Sciences** **Abstract**

| | |
|---|---|
| Author | Juha Vaaksio |
| Title | Design of ultrasonic cleaner |
| Number of Pages | 45 |
| Date | 25 April 2010 |
| Degree Programme | Electronics Engineering |
| Degree | Bachelor of Engineering |
| Instructor | Janne Mäntykoski, Lecturer |
| Supervisor | Janne Mäntykoski, Lecturer |


The primary objective in this final year project was to design a small ultrasonic cleaner. The driving of the ultrasonic transducers was dealt with a special IC, a half-bridge driver from International Rectifier. This driver is able to control voltages up to 200 volts.

When choosing the way to control the ultrasonic transducers, several different methods were considered.

In the center of the cleaner there is a microcontroller from ATMEL. It takes care of producing the frequency needed for ultrasonic transducers (around 40 kilohertz) and other functions such as displaying time and counting it.

The code for the microcontroller was written and tested. The performance of the driving circuit for ultrasonic transducers was tested. Both the code and the circuit seemed to work properly.



| | |
|---|---|
| Keywords | ultrasonic cleaning, ultrasonics, creating ultrasonics |

## Sisällys

# 1. johdanto

Työn tarkoituksena oli suunnitella ultraäänipesurin teoriatason ja käytännön toteutus. Pesuri päätettiin toteuttaa itse, koska vakavasti otettavat kaupalliset sovellukset ovat harrastekäyttöön liian kalliita. Yleensä kuluttajakäyttöön olevat ultraäänipesurit on tarkoitettu korujen puhdistukseen, joten niiden tilavuudet vaihtelevat puolesta litrasta litraan.

Tällaiset pesurit maksavat yleensä halvimmillaan muutamia kymmeniä euroja. Teollisuuskäyttöön tarkoitettujen pesureiden tilavuudet ovat yleensä kymmeniä tai satoja litroja ja näin ollen niiden hinnat ovat satoja tai tuhansia euroja.
Teollisissa pesureissa on yleensä myös mahdollisuus pesuliuoksen lämmittämiseen.

Pesuri päätettiin toteuttaa mikroprosessorin ympärille, jolloin halutut toiminnot oli mahdollista toteuttaa ohjelmoinnin avulla ja pesurin toimintaa olisi mahdollista tarvittaessa muuttaa myöhemmin. Sopivien kaiuttimien ja niiden myyjän löydyttyä alettiin tutkia ja vertailla tapoja niiden ohjaamiseksi. Sopivan tavan löydyttyä kirjoitettiin koodi mikroprosessorille, koottiin kytkennät ja testattiin niiden toimivuutta.

# 2. Ultraäänet

## 2.1 Ultraäänet ja niiden sovelluksia

Ultraääniksi kutsutaan ääniä, joiden taajuus on välillä 20 kilohertsiä ja noin 10 terahertsiä. Alaraja on seurausta ihmisen normaalin kuuloalueen ylärajasta. [1, s. 158.]

Ultraääniä käytetään useisiin eri tarkoituksiin kuten materiaalien tutkimiseen, kaikuluotaukseen, hitsaukseen ja puhdistamiseen. Eräitä yleisesti tunnettuja sovelluksia ovat aineiden tutkiminen ja tarkastaminen ja lääketieteellinen kuvantaminen. Puhdistuksessa käytettävät taajuudet ovat yleensä välillä 20 – 500 kilohertsiä. [1. s. 160, 4.]

## 2.2 Ultraäänien tuottaminen

Yleisimpiä tapoja tuottaa ultraääniä ovat pietsosähköiset kiteet ja magnetostriktioilmiöön perustuvat kaiuttimet. Myös muita tapoja tuottaa ultraääniä on olemassa, esimerkiksi erilaisiin pneumaattisiin ratkaisuihin perustuvat pillit. [1, s. 160–161.]

### 2.2.1 Pietsosähköiset värähtelijät

Pietsosähköiset värähtelijät ovat eniten käytetty tapa tuottaa ultraääniä. Ne tarjoavat laajan taajuusalueen alkaen 20 kilohertsistä aina pariin tuhanteen kilohertsiin asti. Pietsosähköisessä ilmiössä jännite pietsosähköisessä materiaalissa aiheuttaa materiaalin dimensionaalisen koon muuttumisen. [2, s. 219; 8, s. 186.]

Pietsosähköiset värähtelijät ovat edullisempia ja helpompia valmistaa kuin magnetostriktiset värähtelijät, ne ovat magnetostriktisiä parempia hyötysuhteeltaan ja niiden tarvitsema ohjauselektroniikka on yksinkertaisempaa. [8. s, 236–237.]

Pietsosähköiset värähtelijät omaavat nopeamman taajuusvasteen kuin magnetostriktiset värähtelijät, ja näin ollen ne mahdollistavat erikoiset taajuusvaihtelut sovelluksissa, joissa niitä tarvitaan. Pietsosähköiset kiteet vanhenevat logaritmisesti ajan funktiona. Tämän vuoksi pietsosähköisiä kiteitä vanhennetaan joskus keinotekoisesti valmistuksen yhteydessä korkean lämmön avulla. [5; 8, s. 232.]

Pietsosähköisiä värähtelijöitä käytetään muun muassa korkeajännitegeneraattoreina esimerkiksi sytyttimissä, ultraäänipesureissa, mikrofoneissa ja kaiuttimina, lääketieteellisessä diagnostiikassa ja erilaisissa antureissa. [8, s. 233.]

### 2.2.2 Magnetostriktiset kaiuttimet

Magnetostriktiset kaiuttimet tarjoavat pietsosähköisiä värähtelijöitä paremman fyysisen kestävyyden. Ne ovat pitempi-ikäisiä kuin pietsosähköiset kaiuttimet ja niitä on mahdollista jäähdyttää vedellä suuritehoisissa sovelluksissa. Ne kestävät myös pietsosähköisiä kaiuttimia paremmin lämpöä ja kemikaaleja. Magnetostriktisiä kaiuttimia käytetään myös puhdistussovelluksissa. [8, s. 236–237, 271.]

Magnetostriktisten kaiuttimien käyttö on yleisintä kohteissa, joiden olosuhteita pietsosähköiset kaiuttimet eivät kestä. Niitä voidaan käyttää esimerkiksi lähetettäessä ultraääniä suliin, nestemäisiin materiaaleihin. [8, s. 271.]

### 2.2.3 Pneumaattiset ja muut mekaaniset kaiuttimet

Ultraääniä voidaan tuottaa myös paineilmalla toimivien pillien avulla. Niitä käytetään muun muassa vaahdon rikkomisessa, lämmölle herkkien materiaalien kuivauksessa ja pienten kappaleiden päällystämisessä. [8. s. 273–274.]

Pillien tuottama äänenpaine on ihmiselle vaarallinen ja niitä käytetäänkin yleensä äänieristetyissä tiloissa ja huolehtien niiden lähellä työskentelevien ihmisten kuulon suojauksesta. [8, s. 274]

# 3. Ultraäänillä puhdistaminen

## 3.1 Puhdistaminen ultraäänillä

Ultraäänipuhdistuksessa puhdistettavat esineet upotetaan säiliöön, joka on täytetty pesuliuoksella. Tähän altaaseen johdetaan ultraääniä, jotka kavitaation avulla puhdistavat esineet. Ultraäänipuhdistus on turvallinen tapa puhdistaa herkästi rikkoontuvia esineitä ja elektroniikkaa. Ultraäänipuhdistuksessa esineet pidetään korissa tai jonkinlaisessa telineessä. Tämä siksi, että pestävät esineet eivät saa olla kosketuksissa altaan pohjan tai reunojen kanssa. [5; 8, s. 328.]

## 3.2 Kavitaatio ja sen syntyminen

Ultraäänillä puhdistaminen perustuu kavitaatioon. Kavitaatiolla tarkoitetaan ilmiötä, jossa neste alkaa kiehua paineen alenemisen johdosta. Ultraäänipuhdistuksessa tämä ilmiö tapahtuu, kun suurienergiset ultraääniaallot kulkevat nesteessä ja kohdatessaan pestävän kappaleen pinnan aiheuttavat siihen paineaaltoja.

Ultraääniaallot tuottavat kohdatessaan pestävän kappaleen pieniä kuplia. Kuplien koot riippuvat taajuudesta. Taajuuden noustessa kuplien koot pienenevät, mutta samalla niiden määrä nousee. Kupla puhkeaa kohdatessaan puhdistettavan esineen. Puhjetessa sitä ympäröivä neste täyttää syntyneen aukon. Tämä aiheuttaa voimakkaan shokkiaallon, jonka seurauksena lika irtoaa. [1, s. 160; 2, s. 217–218; 4.]

Toinen ultraäänien aiheuttama ilmiö ovat akustiset virtaukset. Niillä tarkoitetaan nesteeseen ääniaaltojen vaikutuksesta syntyviä virtauksia. Kavitaation ja akustisten virtausten suhde riippuu käytetystä taajuudesta. Pienemmillä taajuuksilla kavitaatiota on enemmän kuin

virtauksia, kun taas suurilla taajuuksilla kavitaation vaikutus on pienempi ja akustisten virtausten voimakkaampaa. [3. s. 222–223; 4].

### 3.3 Puhdistuksessa käytetyt taajuudet

Ultraäänipuhdistuksessa käytetään yleensä taajuuksia väliltä 20 – 200 kilohertsiä. Koska pienemmät taajuudet tuottavat isompia ja voimakkaampia kuplia, on niitä suositeltavaa käyttää isojen kappaleiden puhdistukseen. Suurempia taajuuksia taas käytetään helpommin särkyvien tai pienempien esineiden puhdistukseen. [2, s. 219.] Suurempien taajuuksien on tutkimuksissa todettu toimivan paremmin alle mikronin kokoluokkaa olevin partikkelien puhdistuksessa. Esimerkiksi verrattaessa 68 ja 40 kilohertsin taajuuksia yhden mikronin likapartikkeleilla oli 68 kilohertsin puhdistustehokkuus 95 prosenttia, kun 40 kilohertsin taajuudella se oli 88 prosenttia. [7.]

### 3.4 Puhdistukseen tarvittava pesuteho

Ultraäänitehon pesuliuoksessa tulee ylittää tietty taso, jotta kavitaatiota tapahtuu. Tämä teho on suurimmassa osassa tapauksia noin 0,5 – 0,6/W per $cm^2$. Varsinaiselle pesuteholle pidetään yleensä suosituksena pesutehoa 53/W per litra. [8, s. 328].

### 3.5 Puhdistettavat epäpuhtaudet

Pestävät epäpuhtaudet jaetaan kolmeen ryhmään: orgaanisiin, epäorgaanisiin ja partikkelimaisiin epäpuhtauksiin. Epäpuhtaudet voivat olla myös edellisten sekoituksia, jolloin ne eivät välttämättä kuulu suoraan mihinkään edellä mainituista ryhmistä. [7].

Epäpuhtaudet ovat yleensä joko vesiliukoisia tai veteen liukenemattomia. Yleensä orgaaniset epäpuhtaudet kuten öljyt, rasvat, polymeerit, vahat, maalit tai pinnoitteet ovat vettä hylkiviä. Orgaaniset epäpuhtaudet voidaan jakaa karkeasti kolmeen eri luokkaan:

pitkä-, keskipitkä- ja lyhytketjuisiin molekyyleihin. Pesuliuos tulee valita lian tyypin mukaan. [7].

### 3.6 Mahdollisia ongelmia prosessissa

Ultraäänipuhdistusta pidetään turvallisena puhdistustapana, mutta joitakin asioita on syytä ottaa huomioon. Esimerkiksi huokeat materiaalit saattavat vaurioitua matalia taajuuksia käytettäessä. Myös mekaaniset vauriot huokeille materiaaleille ovat mahdollisia. Tapoja ehkäistä vaurioita ovat taajuuden vaihtelu ja pestävien kappaleiden liikuttelu pesun aikana. [3, s. 222].

Käytettäessä tulenarkoja pesuliuoksia on syytä huolehtia paloturvallisuudesta. Alhaisen leimahduspisteen omaavia pesuliuoksia tulee välttää. Samoin vahvoja happoja ja emäksiä ilman erikoisjärjestelyjä tulee välttää. Näitä voidaan kuitenkin erityistapauksissa käyttää erillisissä astioissa pesuliuoksena. [6.]

Myös meluhaittoja saattaa aiheutua käytettäessä alhaisia taajuuksia (alle 30 kilohertsiä). Ne saattavat aiheuttaa aliharmonisia ääniä, jotka ovat ihmisen kuultavissa. [3. s 222].
Myös pestäville esineille tapahtuva rasitus pesuneste-ilmarajapinnassa on syytä huomioida. Siksi on siis syytä huolehtia, että koko pestävä esine on pestävässä liuoksessa.

Suurin huomio vaurioiden ehkäisyssä on kiinnitettävä pesuliuoksen valintaan. Vääränlainen pesuliuos saattaa vahingoittaa pestäviä esineitä. [3. s.222; 6]. Ilman kautta pesurista kulkeutuvien ultraäänien intensiteetti ei tiettävästi ole ihmiselle vaarallinen. [5].

# 4. Pesuprosessi

### Kuplien poistaminen

Kaasujen poistaminen pesuliuoksesta on tärkeää. Poistamattomat liuenneet kaasut liittyvät kavitaation aiheuttamiin kupliin ja vähentävät niiden puhdistavaa vaikutusta. Kaasujen poistaminen voidaan tehdä lämmittämällä pesuliuosta tai käyttämällä pesuria pelkän pesuliuoksen kanssa ilman pestäviä esineitä. Tarvittava aika vaihtelee välillä 10 - 30 minuuttia riippuen altaan koosta, pesuliuoksen lämpötilasta ja pesuliuoksesta. [5.]

Onnistuneesti kaasuista tyhjennetty liuos voidaan havaita siitä, ettei pestessä pesuliuoksen pintaan nouse pieniä kuplia [5]. Poistettavat kaasut ovat muun muassa happea [9].

### Taajuuden vaihtelu

Taajuuden vaihtelulla ehkäistään seisovia aaltoja pesuliuoksessa ja pestäville pinnoille aiheutuvia vaurioita. Taajuuden vaihtelu on pientä, luokkaa 2-3 kilohertsiä. Toinen tapa ehkäistä pestävien esineiden vaurioitumista pesurissa on liikutella niitä pesemisen aikana. [3, s. 222].

### Tarvittavat pesuajat

Vaadittu pesuajan kesto riippuu lian määrästä ja laadusta, pesuliuoksen lämpötilasta ja halutusta puhtauden tarkkuudesta [6]. Lämpötilan kasvaessa on kavitaatiokuplien syntyminen helpompaa ja näin ollen puhdistuminen tapahtuu nopeammin [3, s. 222].

**Pesuliuoksen lämpötilat**

Parhaat tulokset saavutetaan yleensä lämpötila-alueella 50 - 65 °C [6].
Korkeampi lämpötila auttaa rasvojen liukenemista ja nopeuttaa kaasujen poistoa [9].

**Pesuliuokset**

Pesuliuos tulee valita puhdistettavan lian mukaan. Vesi on paras liotin orgaanisille ja epäorgaanisille materiaaleille. Vesi ei kuitenkaan käy veteen liukenemattomille epäpuhtauksille, kuten joillekin kiillotustahnoille. [7.]

**Huuhtelu**

Pestyt kappaleet on syytä huuhtoa pesun jälkeen pesuliuoksen ja likapartikkelijäämien poistamiseksi [5].

**Pesureiden altaat**

Muovisia altaita ei suositella käytettäväksi, koska ne saattavat absorboida ultraäänien energiaa. Säiliön tulisi olla mitoitettu niin, että pestävät asiat voidaan pitää korissa. Sen olisi suotavaa olla muutaman sentin etäisyydellä altaan reunoista ja muutaman sentin korkeudella altaan pohjasta. Korin tulisi olla valmistettu ruostumattomasta teräksestä, koska ohuemmat ja kevyemmät materiaalit saattavat absorboida ultraääniaaltoja. [3. s. 225; 9.]

# 5. Pesuri

## 5.1 Pesurin suunnittelu

Pesuri toimii mikroprosessorin ympärillä. Mikroprosessori hoitaa kaiuttimien ohjaamisen ja muun tarvittavan toiminnan. Mikroprosessorilla tuotetaan 40 kilohertsin perustaajuus, joka heiluu välillä 39,5–40,5 kilohertsiä. Mikroprosessorilta saatu kanttiaalto ohjataan ajuripiirin kautta kaiuttimille.

Mikroprosessori hoitaa myös 7-segmenttinäytön ohjaamisen apupiirin avulla. Näytöllä esitetään jäljellä oleva pesuaika. Aika luetaan painonapeilla. Samoin painonapilla käynnistetään itse pesu. Suurin mahdollinen pesuaika on 999 minuuttia.

### Käyttöjännitteet

Mikroprosessorin ja muun elektroniikan käyttöjännite tuotetaan hakkurimoduulilla. Moduulista saatuja jännitteitä ovat viisi volttia mikroprosessorille ja 12 volttia MOSFETtien ohjainpiireille. 100 voltin jännite kaiuttimille toteutettiin erillisen rengassydänmuuntajan avulla.

### Mikroprosessori

Mikroprosessoriksi valittiin Atmelin ATMEGA32-16PU-mikroprosessori sen saatavuuden, riittävien ominaisuuksien ja ohjelmointilaitteen yhteensopivuuden vuoksi. Kyseisessä mikroprosessorissa on riittävä määrä ajastimia (kaksi 8-bittistä ja yksi 16-bittinen), sopiva käyttöjännitealue (2,7 – 5,5 V), sopiva kotelointi (40-pinninen läpiladottava PDIP-kotelo) ja JTAG-liitäntä helpompaa ohjelmointia ja prototyyppitestausta varten.

**Kaiuttimet ja niiden ohjaus**

Kaiuttimet hankittiin internetkaupan kautta. Ne ovat mallia MPI-4538D-40H. Sovitettu taajuus on 40 kHz (+- 0,5khz), impedanssi 15 ohmia ja kapasitanssi 3700- 4500 pikofaradia ± 10 prosenttia. Kaiuttimien suotavaa käyttöjännitettä ei ilmoitettu, ainoa ilmoitettu arvo jännitteelle oli maksimiarvo, 500 volttia.

Kaiuttimien ohjaus hoidetaan International Rectifierin valmistaman IRS2004-puolisilta-ajuripiirin avulla. Tämä siksi, että näin toimimalla saadaan minimoitua oheiskomponenttien tarve ja kaiuttimet voivat toimia omalla, 100 voltin jännitteellään.
Valmiilla ohjauspiirillä varmistetaan myös se, että puolisillan molemmat ohjaimet eivät ole johtavana samaan aikaan. Tätä varten IRS2004-ohjaimessa on kiinteä, tyypillisesti 520 nanosekunnin dead-time, jolloin kumpikaan MOSFETeistä ei ole johtavassa tilassa. Ajuripiiri on koteloitu 8-jalkaiseen läpiladottavaan PDIP-koteloon. Se kykenee ohjaamaan korkeintaan 200 voltin jännitettä. Ohjainpiiri ohjaa kahta IRF640–MOSFETtiä. Ne valittiin riittävän jännitteen- ja virrankeston perusteella. Myös kotelointi (TO-220), saatavuus ja hinta olivat sopivia.

**Käyttöliittymä**

Käyttöliittymänä toimii 7-segmenttinäyttö, 3 painonappia ja yksi led-valo. 7-segmenttinäytöllä esitetään jäljellä oleva aika. Aika asetetaan kahdella painonapilla, joita luetaan mikroprosessorilla. Myös pesu käynnistetään painonapin avulla, ja led-valo toimii pesemisen käynnissä olemisen merkkivalona. Näiden lisäksi käyttäjälle on näkyvissä päävirtakytkin koko laitteelle.

**Allas**

Allas tehtiin ruostumattomasta teräksestä. Sisämitoilla 220 x 120 x 200mm saadaan altaan tilavuudeksi 5,28 litraa.

$V = P_a * H = 26400\ mm^2 * 120\ mm = 5280000\ mm^3 = 5{,}28\ l$

Tällöin häviöttömäksi pesutehoksi tulee 50.53 W/l, joka täyttää teholle asetetun tavoitteen. Koska allasta ei kuitenkaan tulla täyttämään ylärajaan asti, voidaan käyttökelpoisena tilavuutena pitää noin 4,5 litraa. Tällöin pesutehoksi per litra muodostuu noin 59 wattia. Määritellyillä sisämitoilla saadaan altaaseen mahtumaan kori, jonka pohjan pinta-ala on 200 x 100millimetriä, mikä mahdollistaa jo hieman kookkaampien kappaleiden pesemisen.

### 5.2 Pesurin käytännön toteutus

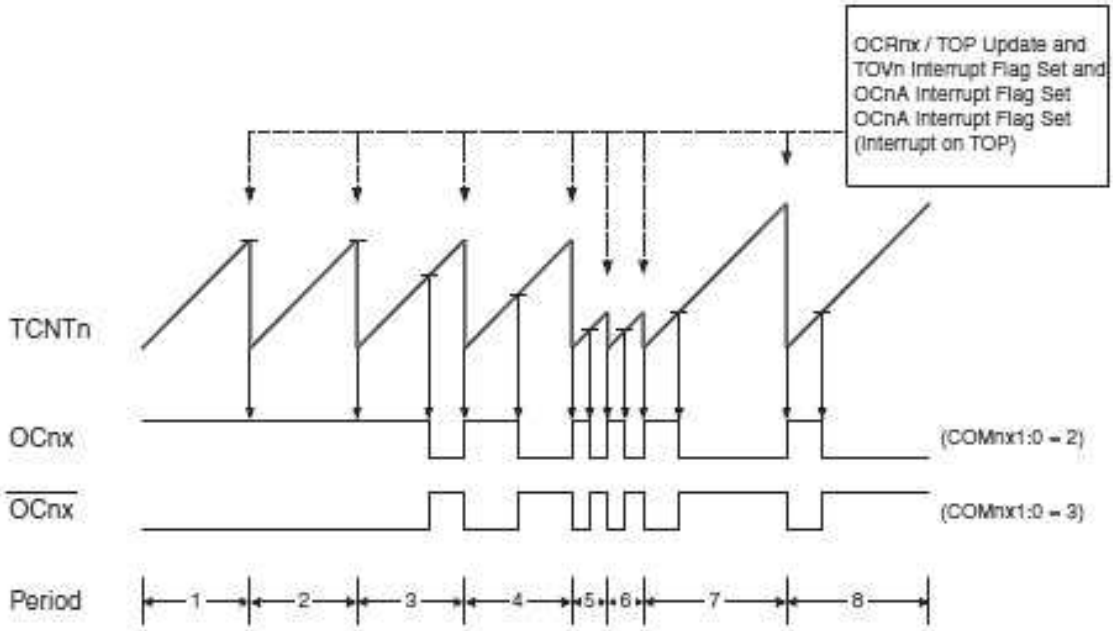
#### 5.2.1 Mikroprosessorin kytkennät

Mikroprosessorille kytkettiin käyttöjännite jalkoihin VCC ja AVCC. Maa kytkettiin jalkoihin GND (jalka 11) ja GND (jalka 31). Näiden välille kytkettiin ohituskondensaattoriksi 0.1 mikrofaradin kondensaattori. 8 megahertsin kideoskillaattori kytkettiin jalkojen XTAL1 (jalka 12) ja XTAL2 (jalka 13) välille. Kiteen jaloista kytkettiin sen tarvitsemat kondensaattorit maihin. Reaaliaikakellon tarvitsema 32768 hertsin kide kytkettiin mikroprosessorin jalkojen TOSC1 (jalka 28) ja TOSC2 (jalka 29) välille. Reaaliaikakellon kiteelle ei ole tarpeen käyttää kondensaattoreita. [10, s. 27,31].

Painonapit ajalle ja pesurin käynnistykselle kytkettiin mikroprosessorin jalkoihin INT0 (jalka 16), INT1 (jalka 17) ja TXD (jalka 15). Merkkivalo pesun käynnissä olemiselle kytkettiin vastuksen kautta käyttöjännitteeseen ja jalkaan PA0 (jalka 40). Pesurin ja mikroprosessorin kytkentäkaavio on liitteenä 1. Mikroprosessorin koodit ovat liitteenä 2.

### 5.2.2 Taajuus

Taajuus on toteutettu mikroprosessorin ajastin/laskuri 1:n (TIMER/COUNTER1) avulla. Laskurille ei käytetä jakajaa, joten saadaan käytettäväksi 8,000,000 pulssia per kellojakso. Taajuuden tuottamiseksi käytetään nopeaa pulssinleveysmodulaatiota (Fast Pulse Width Modulation) ja ajastimen moodia 15. Tässä tilassa laskuri laskee nollasta asetettuun ylärajan arvoon ja palaa sen jälkeen takaisin nollaan. [10, s.104]




*Kuva 1: Ajastimen TIMER1 ajoitukset (10, s. 105)*

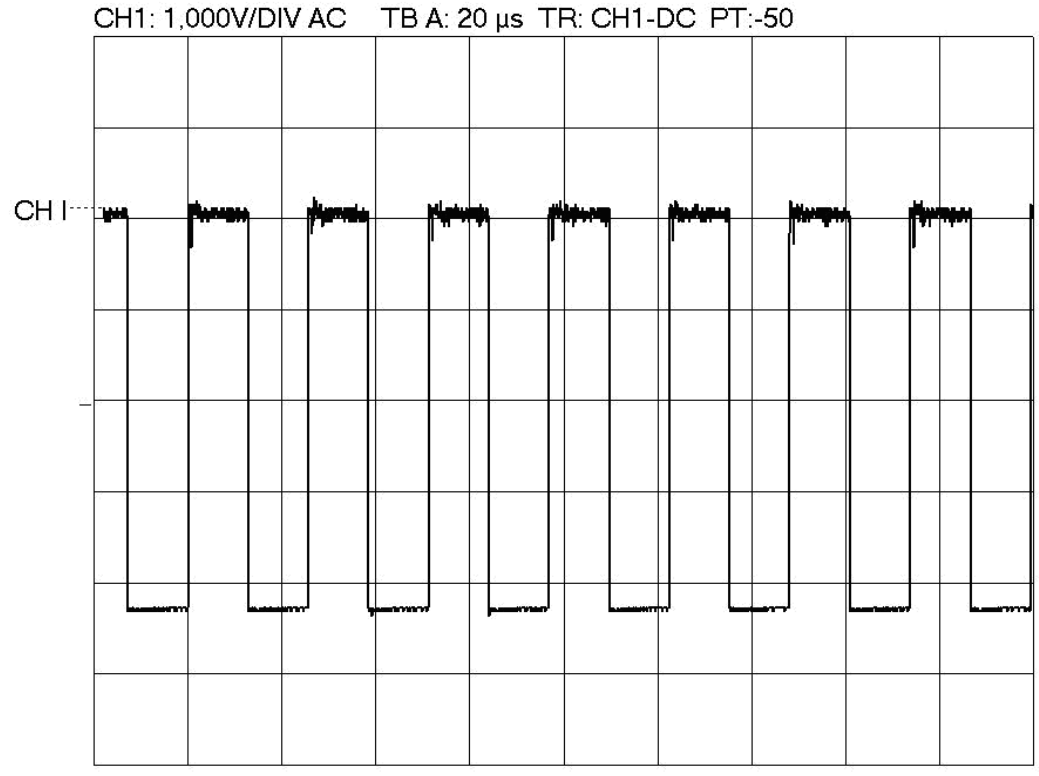
Ajastimelle valitaan toimintamoodi 15 asettamalla valintabitit WGM10, WGM11, WGM12 ja WGM13 ykkösiksi. Bitit WGM10 ja WGM11 ovat rekisterissä TCCR1A ja bitit WGM12 ja WGM13 rekisterissä TCCR1B. [10, s. 112,114.]

Bitillä CS10 rekisterissä TCCR1B asetetaan jakajaksi 1. Tällöin laskuri toimii suoraan mikroprosessorin kellotaajuudella. [10, s. 115.]

Ulos tuleva kellotaajuus saadaan seuraavan kaavan mukaan:

$$f_{osc} = \frac{Kellotaajuus}{2 * Jakaja * Yläraja}$$

Rekisteriin OCR1A asetetaan laskennan yläraja. Kellotaajuuden ollessa 8 megahertziä, saadaan muuttamalla ylärajaa välillä 98 – 102 ulostulevaksi taajuudeksi kuvan 2 mukaiset 39215,69 – 40816,32 hertsiä.




*Kuva 2: Mikroprosessorilta saatu kanttiaalto*

### 5.2.3 Ajan lukeminen

Ajan lukeminen toteutettiin keskeytyksien INT0 ja INT1 avulla. Keskeytys saadaan, kun mikroprosessorin jalkaan kytketty ja käyttöjännitteeseen ylösvedetty painonappi alas painettuna aiheuttaa mikroprosessorin sisääntuloon loogisen nollan.

Keskeytykset INT0 ja INT1 asettavat lipun, jonka mukaan ohjelma toimii. Kun lippu on asetettu, ohjelma lisää tai vähentää aikaa painetun napin mukaan niin kauan kuin kyseinen nappi on painettuna. Ajan lisäämisessä on 150 millisekunnin viive, koska muuten ajan tarkka asettaminen olisi vaikeaa.

### 5.2.4 Pesun käynnistys

Pesu käynnistetään painokytkimen avulla. Se on kytketty käyttöjännitteeseen ylösvedettynä mikroprosessorin jalkaan PD1. Painonappia painettaessa ohjelma tarkistaa, onko aika nollaa suurempi. Jos aikaa ei ole asetettu tai se on nolla, ei mikroprosessori ala lähettää taajuutta kaiuttimien ohjauspiirille. Jos aika on nollaa suurempi, lähetetään taajuus kaiuttimien ohjauspiirille ja aloitetaan ajan vähentäminen.

Taajuuden lähettäminen kaiuttimille aloitetaan käynnistämällä ajastin TIMER1. Se tapahtuu asettamalla kyseisen ajastimen jakaja rekisteriin TCCR1B. Jakajan ollessa nolla ei ajastin ole päällä. Ajan vähentäminen puolestaan käynnistetään asettamalla ajastimen TIMER2 jakaja. Ajastin TIMER2 huolehtii kellonajan laskemisesta.

### 5.2.5 Ajan vähennys

Ajan vähennys hoidetaan ajastimen TIMER2 avulla. Se toimii ulkoisen 32768 hertzin kiteen avulla. Tämä taajuus jaetaan suurimmalla jakajalla (1024). Tästä saadaan tulokseksi

luku 32. Ajastimen TIMER2 avulla lasketaan keskeytyksiä. Kun niitä on saatu 32, on kulunut yksi sekunti. Tämän jälkeen aletaan laskea sekunteja, ja kun niitä on kulunut 60, vähennetään aikaa minuutilla.

### 5.2.6 Näytön ohjaaminen

Näytön ohjaaminen on tarkoitus toteuttaa Maximin valmistaman MAX7221-ohjainpiirin avulla. Se on SPI-väylää käyttävä ohjainpiiri korkeintaan kahdeksaa 7-segmenttinäyttöä varten. Käyttämällä ohjainpiiriä 7-segmenttinäyttöjen ohjaamiseen saadaan vähennettyä mikroprosessorilta näytölle vedettävien johtimien määrää. Tässä tapauksessa käyttämällä SPI-väylää selvitään kolmella johtimella. Kyseisen ohjainpiirin vaikean saatavuuden vuoksi päädyttiin testausvaiheessa näyttöjen ohjaus toteuttamaan multipleksaamalla. Näyttö toteutettiin käyttämällä 7-segmenttinäyttöjä. Niissä saadaan seitsemän eri segmentin avulla muodostettua heksadesimaaliluvut 0-F. 7-segmenttinäytöt koostuvat periaatteessa seitsemästä eri ledistä, ja tämän peruja on niiden lajittelu CC- ja CA-tyyppisiin. CC (Common Cathode, yhteinen katodi) -tyyppisellä näytöllä ledien katodit ovat kytketyt yhteen jolloin niitä ohjataan kytkemällä yhteinen katodi maihin ja kytkemällä halutun segmentin anodi käyttöjännitteeseen. CA (Common Anode, yhteinen anodi) -tyyppisessä näytössä taas ledien anodit on kytketty yhteen. Niitä ohjataan kytkemällä yhteinen anodi käyttöjännitteeseen ja kytkemällä halutun segmentin katodi maihin.

Näytön ohjaaminen toteutettiin testausvaiheessa kytkemällä 7-segmenttinäyttöjen anodit mikroprosessorin jalkoihin PA1-PA7. Desimaalipisteet jätettiin kytkemättä, koska niitä ei tässä tapauksessa tarvita. Näytön vaihtaminen ja näin ollen oikean luvun kirjoittaminen oikeaan näyttöön hoidettiin multipleksaamalla.

Näin toimimalla saatiin kytkettyä kaikkien näyttöjen anodit samalla kertaa ja eri näyttöjen ohjaamiseen tarvittiin vain kolme eri signaalia sen sijaan, että kaikkien näyttöjen jokaista

segmenttiä olisi ohjattu itsenäisesti. Tässä toteutuksessa näyttöjä ohjataan yksi kerrallaan. Kun tämä toteutetaan tarpeeksi suurella taajuudella, niin käyttäjälle näyttäisi, kuin kaikki näytöt palaisivat samanaikaisesti. Koodi näyttöjen ohjaamiseen muokattiin valmiina saatavilla olevasta koodista.

### 5.2.7 WDT eli vahtikoira-ajastin

Kyseisen ajastimen tarkoitus on vahtia mikroprosessorin toimintaa. Mahdollisissa vikatilanteissa se huomaa mikroprosessorin toiminnan pysähtymisen ja käynnistää mikroprosessorin tarvittaessa uudelleen. Ajastimelle asetettiin suurin mahdollinen jakaja, jolloin ajastin tarkistaa tilansa noin 2.1 sekunnin välein. Vahtikoira nollataan pääohjelmassa ja aikaa vähentävissä tai lisäävissä aliohjelmissa.

# 6. Oheiskomponenttien valinnat

## 6.1 Oheiskomponenttien valinnat

Seuraavaksi käsitellään oheiskomponenttien valinnat ja niiden tarvitsemien komponenttien mitoitukset. Käsiteltävät komponentit ovat kaiuttimien ajuri-piiri, kaiuttimien ajuripiirin oheiskomponentit, ohituskondensaattorit, kaiuttimien käyttöjännitteen luomiseen käytetyt komponentit ja käyttöjännitteen suodatukseen käytetyt komponentit.

## 6.2 Kaiuttimien ajuripiirien valinta

Kaiuttimien ohjaaminen toteutettiin puolikkaan H-sillan avulla. Tässä tapauksessa ohjataan kahta N-kanavaista MOSFETtiä, joilla kytketään kaiutin joko käyttöjännitteeseen tai maihin. Näin saatiin toteutettua helpoiten konfiguraatio, jossa mikroprosessorilta saadulla 0-5 voltin jännitteellä pystytään ohjaamaan korkeampaa jännitettä. Tässä tapauksessa ohjattava jännite on kaiuttimille käytettävä 100 voltin jännite.

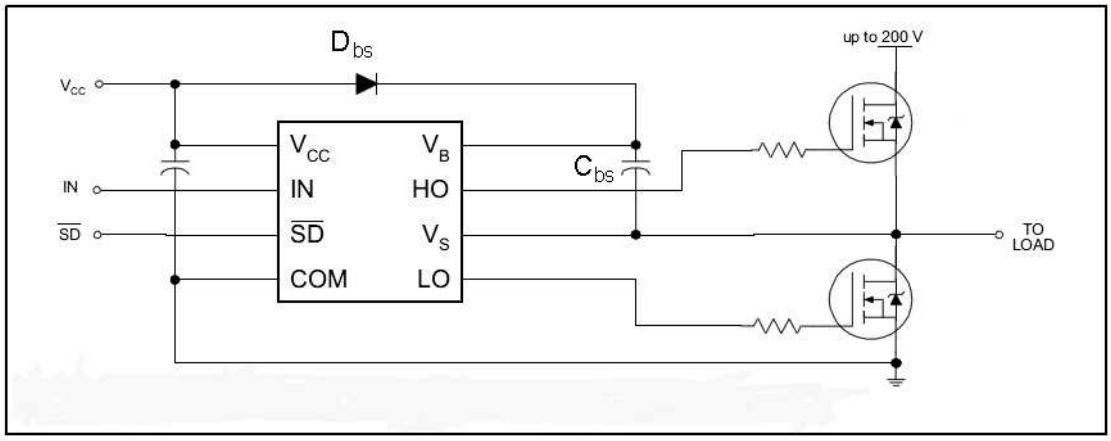
Muita harkittuja vaihtoehtoja olivat high- tai low-side-ohjaimet ja omat sovitusmuuntajat kullekin kaiuttimelle. Low-side-ohjainpiiristä luovuttiin niiden sopimattomuuden vuoksi. Ne eivät yleensä kestä riittävän suurta ohjattavaa jännitettä. High-side-ohjaimet taas kestäessään korkeamman ohjattavan jännitteen eivät tarjoa sovellukseen riittävää nopeutta toivotussa koteloinnissa. Kaiutinkohtaisista sovitusmuuntajista luovuttiin ratkaisun toteutuksen monimutkaisuuden vuoksi.

Kaiuttimien ohjauspiirin tuli olla saatavilla, half-bridge tyyppinen, hyväksyä sopivaa ohjausjännitettä, kestää riittävän korkeaa ohjattavaa jännitettä, olla pakkaustyypiltään sopiva ja mahdollistaa ohjata päälle tai pois erillisellä signaalilla.

Ohjainpiiriksi valittiin International Rectifierin valmistama IRS2004-ajuripiiri. Se kykenee ohjaamaan 200 voltin jännitettä, hyväksyy 3,3 tai 5 voltin jännitteen logiikkasisääntuloonsa, on mahdollinen ohjata päälle tai pois yhden jalan kautta ja on saatavilla joko 8-jalkaisena SOIC- tai PDIP-piirinä. [11.]

### 6.3 Ajuripiirin oheiskomponentit

Ajuripiiri IRS2004 tarvitsee toimiakseen kolme ulkoista komponenttia: Ohituskondensaattori kytkettiin käyttöjännitteen ja maan välille pitämään käyttöjännite vakaana ja niin sanotun bootstrap-kondensaattorin $C_{bs}$ ja diodin $D_{bs}$. Komponenttien sijoittelu on näkyvillä kuvassa 3.




*Kuva 3: MOSFET-ajuripiirin kytkentä*

Bootstrap-kondensaattorin ja diodin muodostamaa kytkentää käytetään huolehtimaan ohjaimen high-side-ohjauslogiikan käyttöjännitteestä. Kytkennässä esiintyvällä MOSFETin gate-vastuksella säädetään kytkentäaikoja päälle ja pois. Yleensä ne ovat arvoltaan 10–500 ohmia. [12.]

### 6.3.1 Bootstrap–kytkennän latausdiodi

Valittavalta diodilta vaadittiin seuraavat ominaisuudet: riittävä jännitteenkesto, nopeus, tehonkesto ja sopiva kotelointi. [13.] Nämä vaatimukset silmällä pitäen päädyttiin valitsemaan On Semiconductorin valmistama MUR120-diodi. Sen vastasuuntainen jännitteenkesto on 200 volttia, toipumisaika on 75 nanosekuntia, tehonkesto 1 ampeeri ja kotelointi normaali läpiladottava pakkaus. [18.]

### 7.3.2 Bootstrap-kytkennän kondensaattorin valinta

Bootstrap-kondensaattorin valinnassa otettiin huomioon, että kondensaattorin tulee olla kapasitanssiltaan riittävä ylläpitämään tarvittavaa jännitettä high-side-ohjaimen tarvitseman ajan. Lisäksi sen tulee olla riittävän matala ESR:äinen (Equivalent Series Resistance). [14.]

Kondensaattorin arvo lasketaan kaavalla 1:

$$C_{boot} = \frac{Q_{tot}}{\Delta V_{bs}}, \quad (1)$$

,jossa $Q_{tot}$ on kytkennässä esiintyvien varausten summa ja $\Delta V_{bs}$ on suurin sallittu jännitteen putoama.

$Q_{TOT}$ lasketaan kaavalla 2:

$$Q_{TOT} = Q_G + Q_{LS} + (I_{LK\_GE} + I_{QBS} + I_{LK} + I_{LK\_DIODE} + I_{LK\_CAP} + I_{DS-}) \cdot T_{HON}\ , \quad (2)$$

jossa $Q_G$ on MOSFETin hilan varaus, $Q_{LS}$ on ajuripiirin sisäisten tasonvaihtajien tarvitsema lataus, $I_{GSS}$ on MOSFETin vuotovirta hilalta lähteelle, $I_{QBS}$ on ajuripiirin kelluvan osion kuluttama virta, $I_{LK}$ on ajuripiirin kelluvan osion vuotovirta $I_{LK\_DIODE}$ on bootstrap diodin vuotovirta, $I_{LK\_CAP}$ on bootstrap kondensaattorin vuotovirta, $I_{DS-}$ on ajuripiirin diodin biasvirta sen pälläollessa ja $T_{HON}$ on aika, jonka ajuripiirin high-side puoli on päällä.

$\Delta V_{BS}$ lasketaan kaavasta 3:

$$\Delta VBS \leq V_{CC} - V_F - V_{GE\min} - V_{CEon}\,, \qquad (3)$$

jossa $V_{CC}$ on käyttöjännite, $V_F$ on bootstrap-diodin myötäsuuntainen jännite, $V_{GS\_MIN}$ on alin haluttu jännite MOSFETin hilalta lähteelle ja $V_{CEon}$ on MOSFETin yli häviävä jännite.

| Laskennassa käytettiin arvoja | lähde |
|---|---|
| $Q_G$ = 67 nC | datasheet IRF640N |
| $I_{QBS}$ = 55 µA | datasheet IRS2004 |
| $I_{CBS}$ = 0 keraamisella kondensaattorilla | |
| $V_{LS}$ = 1,3V | datasheet IRF640N |
| $V_{MIN}$ = 8 V | datasheet IRS2004 |
| $Q_{LS}$ = 5 nC, kun kysymyksessä on alle 600 voltin MGD (MOSFET gate driver) | |
| $I_{LK}$ = 50 µA | datasheet IRS2004 |
| $I_{Lk\ diode}$ = 100 µA | alle 100 ns toipumisaikaisella diodilla |
| $I_{DS-}$ = 0 µA | 0 keraamisella kondensaattorilla |
| $T_{HON}$ = 12,5 µS | |
| $V_{CC}$ = 15V | |
| $V_F$ = 1,04V | datasheet MUR120 |
| $V_{GSMIN}$ = 8 V | datasheet IRF640N |
| $V_{SD}$ = 1,3 V | datasheet IRF640N |

$\Delta V_{BS} \leq V_{CC} - V_F - V_{GEmin} - V_{CEon}$

$\Delta V_{BS}$ = 12 V – 1,04 V – 8 V– 1,3 V = 0,66 V

$$Q_{TOT} = Q_G + Q_{LS} + (I_{LK\_GE} + I_{QBS} + I_{LK} + I_{LK\_DIODE} + I_{LK\_CAP} + I_{DS-}) \cdot T_{HON}$$

$Q_{TOT}$ = 67nC + 5nC + (100nA + 55µA + 50µA + 100 µA + 0+ 0) * 12,5µs = $7{,}456*10^{-8}$ F

$$C_{boot} = \frac{Q_{tot}}{\Delta V_{bs}} = \frac{7.456*10^{-8} nC}{0.66V} = 1.129*10^{-7} F$$ [15, s. 2-3.]

Laskettu arvo on minimi, ja peukalosäännön perusteella kerrotaan saatu arvo viidellätoista [16, s. 3].

$1{,}129 * 10^{-7}\,F$ * 15 = $1{,}694 * 10^{-6}$ F = 1,69 µF

Kytkennässä suositellaan käytettäväksi rinnan keraamista ja elektrolyyttikondensaattoria, koska pienen keraamisen ja suuren elektrolyyttikondensaattorin rinnankytkentä on yleensä paras kompromissi. Tällaisessa kytkennässä keraamisen kondensaattorin pieni sisäinen sarjaresistanssi takaa nopean latautumisen ja rajoittaa jännitteen nousua ajan suhteen, kun taas suuri elektrolyyttikondensaattori takaa riittävän pienen jännitteen putoamisen halutulla jännitealueella. [15, s. 5.]

Kytkennässä päädyttiin käyttämään kahta rinnankytkettyä kondensaattoria: 1,5 µF mikrofaradin elektrolyyttikondensaattoria ja 0,22 mikrofaradin keraamista kondensaattoria. Keraamisen kondensaattorin sisäisen resistanssi on huomattavasti elektrolyyttikondensaattoria pienempi, kun taas elektrolyyttikondensaattori auttaa pitämään jännitteen tarvittavalla korkeudella. Rinnan kytkettäessä kondensaattorien kapasitanssit lasketaan yhteen, joten saatu kapasitanssi on 1,72 mikrofaradia, joka vastaa riittävän tarkasti laskemalla saatua tavoitetta.

### 6.3.3 Ohituskondensaattorit

Ohituskondensaattoriksi ohjauspiirille valittiin 0.1µF tantaalikondensaattori. Samaa kondensaattoria käytettiin myös muiden mikropiirien ohituskondensaattoreina. Tämän lisäksi mikroprosessorille kytkettiin 10 µF elektrolyyttikondensaattori tasaamaan käyttöjännitettä.

### 6.3.4 MOSFETtien valinta

MOSFETteja valittaessa tutkittiin neljää eri parametria: MOSFETtien tuli olla jännitteenkestoltaan riittävät, helposti ohjautuvat ja tehonkestoltaan riittävät ja niiden tuli omata riittävän pieni tarvittava ohjausjännite. Näiden vaatimusten perusteella valittiin International Rectifierin valmistamat IRF640N Hexfet –sarjan teho-MOSFETit. [17.]

## 6.4 Elektroniikan käyttöjännite

Elektroniikan käyttöjännite tuotetaan modulaarisella hakkurivirtalähteellä. Siitä saadaan viiden voltin jännite viiden ampeerin maksimivirralla ja 12 voltin jännite 2 ampeerin maksimivirralla. Valmiilla moduulilla saadaan teholähteelle myös helposti oikosulku-, ylijännite- ja ylikuormitussuojaus. Teholähteelle luvataan 78 % hyötysuhde ja 80 mVp-p rippelijännite sekä viiden että 12 voltin jännitteelle. [19.]

## 6.5 Kaiuttimien käyttöjännite

Kaiuttimien käyttöjännite 100 volttia tuotetaan kytkemällä 2x50 voltin rengassydänmuuntajan toisiot sarjaan kuvan 4 mukaisesti. Näin saadaan 100 voltin jännite 300 volttiampeerin teholla. Rengassydän valittiin pakkamuuntajaan nähden pienemmän koon ja painon sekä pienemmän häiriöisyyden vuoksi.

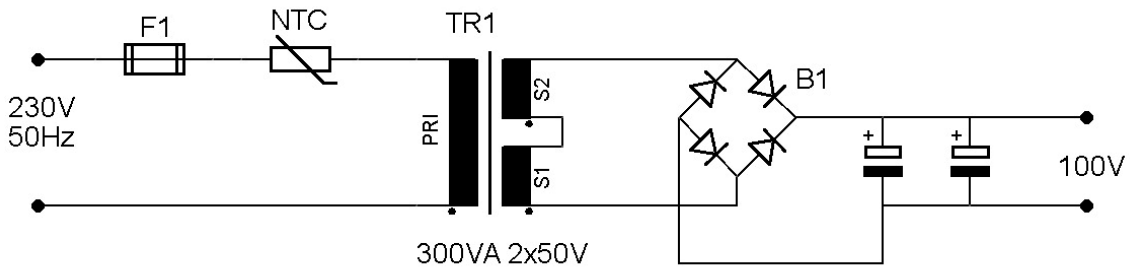
Muuntajalta saatu 100 voltin vaihtojännite tasasuunnataan. Tämän jälkeen huippujännite on noin 140 volttia. Tasasuuntaussillaksi valittiin Comchipin valmistama MP1004G-tasasuuntaussilta. Se kestää 10 ampeerin jatkuvan tehon, hetkellisesti jopa 200 ampeeria. Jännitehäviötä tasasuuntaussillan yli syntyy 1,1 volttia.

Lisäksi kytketään tarvittavat suotokondensaattorit. Niiden määräksi suositellaan arvoa 2200 mikrofaradia per ampeeri. Täten tarvittava määrä suodatusta olisi vähintään 5280 mikrofaradia. Suodatus toteutettiin kytkemällä rinnan kaksi 3300 mikrofaradin kondensaattoria.

Saadun jännitteen aaltoilu voidaan laskea kaavalla $\Delta u = \frac{Iout}{2fC}$, mistä seuraa, että

$\Delta u = \frac{3A}{2*50*6600*10^{-6}F} = 4.545V$. Jännitteen heilunta on hyväksyttävissä rajoissa. [21, s. 239]

Rengassydänmuuntajan ensiön kanssa sarjaan kytkettiin NTC-termistori rajoittamaan käynnistysvirtapiikkiä. Tähän valittiin Epcosen valmistama B57237 NTC (Negative Temperature Coefficient)-termistori. Sulakkeeksi valittiin neljän ampeerin hidas 5 x 20mm lasiputkisulake.




*Kuva 4: Kaiuttimien virtalähteen kytkentä*

### 6.5 Käyttöjännitteen suodatus

Laitteen käyttöjännitteen suodatus hoidetaan Qualtek Electronicsin valmistamalla EMI-suodatinmoduulilla. Kyseinen moduuli hoitaa sekä elektroniikan hakkuriteholähteen että kaiuttimien rengassydänmuuntajan jännitteen suodattamisen. Moduulille luvataan 10 ampeerin tehonkesto, joka tässä sovelluksessa on riittävä. [20].

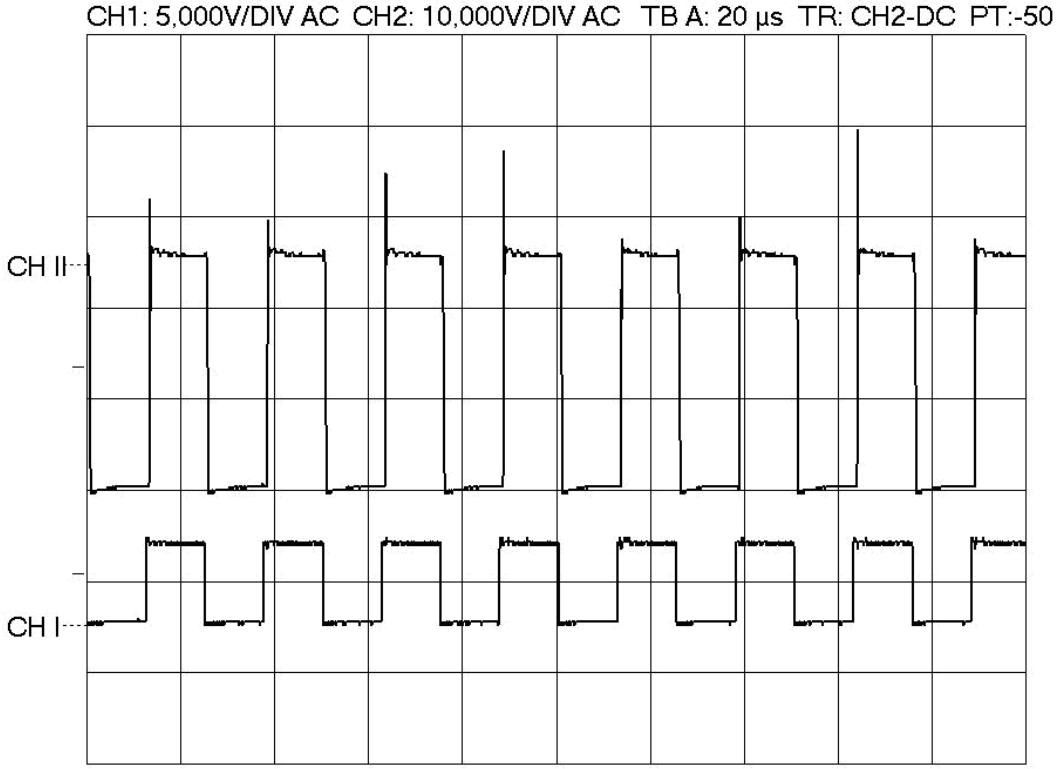
### 6.6 Laitteen kokoonpano ja kotelointi

Laite tullaan kokoamaan metallirunkoiseen koteloon. Kotelon runko tullaan kokoamaan kahden millin vahvuisesta L-profiilista. Laitteen pohja tehdään riittävän paksusta teräslevystä. Laitteen aktiivinen tuuletus tullaan hoitamaan 12 voltin jännitteellä toimivalla 120 x 120mm tuulettimella.

Kaiuttimet kiinnitetään altaan pohjaan ja kylkeen. Kaksi kaiuttimista kiinnitetään altaan pohjaan ja yksi sen sivuun. Laitteen kyljet ja kansi tullaan valmistamaan tarkoitukseen sopivasta ja kestävyydeltään riittävästä materiaalista. Vaihtoehtoina ovat ohut alumiinilevy, ohut ruostumattomasta teräksestä tehty levy tai polykarbonaatista tehty levy.

## 7. Ohjainpiirin toiminta käytännössä

Ohjainpiirin toimintaa testattiin käytännössä ohjaamalla piirillä 25 voltin jännitettä käyttäen kuormana 75 watin hehkulamppua. Kuvassa 5 on esitettynä sekä ohjaava 5 voltin jännite että ohjattava 50 voltin jännite.




*Kuva 5: Ohjaava 5 voltin jännite ja ohjattava 50 voltin jännite*

MOSFETtien kytkennässä on havaittavissa pientä kytkentävärähtelyä (kuva 6). Tämän poistamiseen tullaan kiinnittämään myöhemmin huomiota. Tällä hetkellä kytkennässä käytetään 100 ohmin vastusta hilalla. Tämän vastuksen kokoa muuttamalla, mahdollisella vastuksen ja diodin sarjakytkennällä ja piirilevysuunnittelulla kytkentävärähtelyt pitäisi saada kuriin.


*Kuva 6 : MOSFETin kytkentävärähtely*

Myös se saadaanko kaiuttimista suurin mahdollinen pesuteho käytetyllä jännitteellä, jäi selvittämättä. Mutta koska ohjattava jännite ei pysyessään komponenttien kestorajojen sisäpuolella vaikuta ohjauspiirin toimintaan, voidaan kaiuttimien ohjausjännitettä tarvittaessa nostaa.

## 8. Yhteenveto

Työn tavoitteena oli suunnitella pieni ultraäänipesuri. Tavoitteisiin päästiin. Eniten ongelmia tuotti kaiuttimien ohjaamiseen valittavan tavan valitseminen. Eri vaihtoehtojen eduista ja haitoista oli vaikeaa saada selkeää eroa varsinkaan käytännössä. Ajuripiirin kytkentävärähtelyjen poistamiseen tulee tulevaisuudessa kiinnittää huomiota. Muita käyttötarkoituksia testatunlaiselle ohjauskytkennällä saattaisivat olla erilaiset lämmityssovellukset, valaisuratkaisut pwm-säädöllä ja moottorinohjaukset. Selostuksen valmistumishetkellä (25.4.2010) pesurin käytännön toteutus on vielä kesken eräiden komponenttien ja ennen kaikkea sopivan kokoisen säiliön vaikean saatavuuden takia.

## Lähteet


1 Peltonen, Hannu, Perkkiö, Juha, Vierinen, Kari. Insinöörin (AMK) fysiikka. Osa II. Lahti: Lahden Teho-Opetus Oy, 2004.

2 Kanegsberg, Barbara, Kanegsberg, Edward. Handbook of Critical Cleaning. Florida. CRC PRESS. 2002

3 Welker, R. W., Nagarajan, Ramamurthy, Newberg, Carl E. Contamination and ESD control in high-technology manufacturing. Wiley-IEEE Press. 2006

4 Lawrence, Azar. Cavitation in ultrasonic cleaning and cell distruption. (WWW-dokumentti). <http://www.cemag.us/articles.asp?pid=808>. 2009. Luettu 29.1.2010

5 Frequently asked questions: Parts Washers: Ultrasonic cleaning. (WWW-dokumentti). <http://www.ctgclean.com/faq.php>. 2009. Luettu 30.1.2010

6 Cole-Parmer Technical library. (WWW-dokumentti.) <http://www.coleparmer.co.uk/techinfo/techinfo.asp?htmlfile=ultrasoniccleaner_faq.htm>. 2010. Luettu 30.1.2010

7 Awad, Sami. Ultrasonic Cavitations and Precision cleaning. Precision Cleaning, 2006.

8 Esnminger, Dale; Stulen, Foster B. Ultrasonics – Data, equations, and their practical uses. Florida. CRC Press. 2009

9 Ultrasonic Cleaning Process. (WWW-dokumentti). <http://www.tmasc.com/ultrasonic_cleaning_process.htm>. 2002. Luettu 3.2.2010

10 ATmega32A – Datasheet. (www-dokumentti). < http://www.atmel.com/dyn/resources/prod_documents/doc8155.pdf>. 2009. Luettu 6.4.2010

11 IRS2004PBF. Datasheet. (www-dokumentti). <http://www.irf.com/product-info/datasheets/data/irs2004pbf.pdf>. 2006. Luettu 6.4.2010

12 High side gate driver ;boot strap operation. (WWW-dokumentti). < http://irf.custhelp.com/cgi-bin/irf.cfg/php/enduser/popup_adp.php?p_faqid=14>. 2009. Luettu 4.4.2010

13 Controlling a H bridge with two half bridge drivers. (WWW-dokumentti). 2009. luettu 23.3.2010

14 Values of Capacitors and Diodes for Gate Drivers. (WWW-dokumentti). < http://irf.custhelp.com/cgi-bin/irf.cfg/php/enduser/std_adp.php?p_faqid=31>. Luettu 23.3.2010

15 A. Merello, A. Rugginenti and M. Grasso. *Using monolithic high voltage gate drivers (WWW-dokumentti). < www.irf.com/technical-info/designtp/dt04-4.pdf>. Luettu 6.4.2010*

*16* Jonathan Adams. Bootstrap Component Selection For Control IC's (WWW-dokumentti). www.irf.com/technical-info/designtp/dt98-2.pdf>. *2001. Luettu 6.4.2010*

17 IRF640N. Datasheet. (WWW-dokumentti). <*www.irf.com/product-info/datasheets/data/irf640n.pdf*>.*2004. Luettu 6.4.2010*

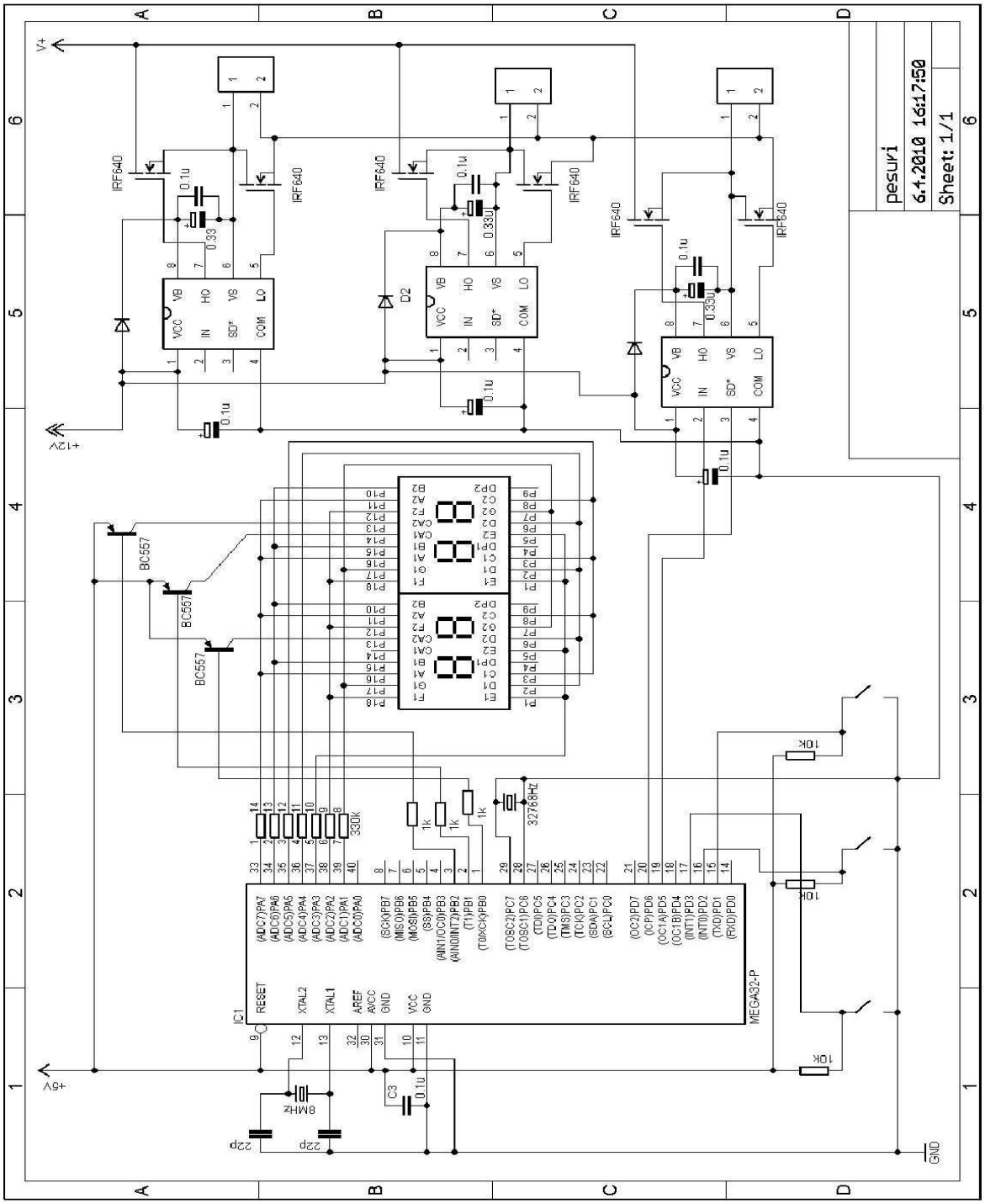
18 MUR120. Datasheet (WWW-Dokumentti). <http://www.onsemi.com/pub/Collateral/MUR120-D.PDF>. 2006. Luettu 6.4.2010

19 RPD-60. Datasheet. (WWW-Dokumentti). <http://www.vekoy.com/UserFiles/File/PDF-liitteet/MW-RPD60.pdf>. 2008. Luettu 6.4.2010

20 858-06/011 – Emi filter datasheet. (WWW-dokumentti). <http://www.qualtekusa.com/Catalog/EMI_Filters/pdf/85806011.pdf>. 2005. Luettu 6.4.2010

21 Silvonen, Kimmo. Elektroniikka ja puolijohdekomponentit. Helsinki: Hakapaino, 2009

## Liite 1: Pesurin kytkentäkaavio



## Liite 2: Mikroprosessorin koodit

```
/* Atmega32, 8MHz.
Ultraäänipesurin pääohjelma.
Luetaan aikaa, ja niinkauan kuin se on yli nollan, ajetaan taajuutta. */

#include <avr/io.h>
#include <util/delay.h>
#include <avr\interrupt.h>
#include <avr\wdt.h>

/* Määrittelyt */
#define F_CPU 8000000UL

/* Funktioiden prototyypit */
void init_aika();
void init_taajuus();
void init_sekunnit();
void init_naytto();
void Print(uint16_t num);
int get_taajuus();
int laske0();
int laske1();

/* Muuttujia */
volatile uint16_t aika = 0;
uint8_t sekunti = 0;
volatile uint8_t lippu0;
volatile uint8_t lippu1;

/* Pääohjelma alkaa */
int main(void)
{
init_sekunnit();                    // Timer2:sen initialisointi, lasketaan sekunteja

init_naytto();                      // 7-segmenttinäytön initialisointi

init_aika();                        // Ajanluvun initialisointi

init_taajuus();                     // Timer1:sen initialisointi, taajuus tällä
```

```
WDTCR = (1 << WDE) | (1 << WDP2) | (1 << WDP1) | (1 << WDP0);
                                    // Watchdog päälle, suurin jakaja -> väli noin 2.1s
while(1)
{
laske0();
laske1();

Print(aika);                        // Tulostetaan aika näkyviin
wdt_reset();                        // Resetoidaan watchdog

OCR1A = 98;                         // Joten taajuus on ~39200-40800 Hz
_delay_ms(500);
OCR1A = 99;
_delay_ms(500);
OCR1A = 100;
_delay_ms(500);
OCR1A = 101;
_delay_ms(500);
OCR1A = 102;
_delay_ms(500);

if(bit_is_clear(PIND,1))            // Käynnistetään painonappulalla
{
if(aika == 0)                       // Jos aika = 0, ei käynnistetä taajuuden ajoa.
TCCR1B &= ~(0 << CS10);
else
{
TCCR2 |= (1 << CS22) | (1 << CS21) | (1 << CS20) ;
                                    // Käynnistetään timer2
TCCR1B |= (1 << CS10);
                                    // Käynnistetään timer1
PORTA &= ~_BV(0);
                                    // Sytytetään pesemisen merkkiled
PORTD |= _BV(6)
                                    // Enable ajuripiirille
}
}
else
if(aika == 0)                       // Jos aika on 0, niin timer1 pois päältä
{
TCCR1B &= ~(1 << CS10)              // = ei taajuutta kaiuttimille.
```

```
PORTA |= _BV(0);                    // Pesun merkkiled sammuneena, jossei pesua
                                    käynnissä
PORTD &= ~_BV(6)                    // Ajuripiirin enable nollaksi.
}
}
}
void init_sekunnit()                // Timer2:sella reaaliaikakello
{
                                    // Ulkoiselta 32.768 kiteeltä kello

ASSR = (1 << AS2);                  // Kellon lähteeksi TOSC1, mihin kide.

TCCR2 = (1<<WGM21) | (0 << CS22) | (0 << CS21) | (0 << CS20);
                                    // Jakajaksi 1024 -> 32768/1024 = 32
                                    // CTC - Mode (Clear Timer on Compare)
TIFR = (1<<OCF2);
                                    // Asetetaan OCF2 ->nollataan keskeytykset
TIMSK = (1<<OCIE2);
                                    // Timer/Counter2 Output Compare Match Interrupt
                                    Enable
OCR2 = 32;
                                    // Vertoarvoksi 32, kun laskettu on kulunut 1 sekunti.
while(ASSR&(1<<OCR2UB));
                                    // Odotetaan rekisterin päivittymistä
sei();                              // Globaalit keskeytykset päälle
}

void init_taajuus()
        {
DDRD |= _BV(5);                     // Pd5 ulostuloksi

TCCR1A = (1<<COM1A0) | (1<<WGM11) | (1<<WGM10);
                                    // COM1A0 asettaa duty cycleksi 50%, kun käytetään
                                    ajastimen moodia 15.
                                    // WGM10-13 asettavat moodin.

TCCR1B = (0<<CS10) | (1<<WGM13) | (1<<WGM12);
                                    // CS10 asettaa jakajan, tässä 1.
}

void init_aika()
```

```
{
DDRD = 0x00;                    // PD2 ja PD3 sisääntuloiksi

PORTD = 0x00;

GICR |= 1<<INT0;                // INT0 sallittu
GICR |= 1<<INT1;                // INT1 sallittu
MCUCR |= 1<<ISC01;              // INT0 triggaa laskevalla reunalla
MCUCR |= 1<<ISC11;              // INT1 triggaa laskevalla reunalla

sei();                          // Sallitaan keskeytykset globaalisti
}
int laske0()
        {
if(lippu0)                      // keskeytys int0:sta = lisää aikaa
{
do                              // Niinkauan kuin nappi on painettu, lisätään aikaa
{
aika++;
_delay_ms(150);                 // Joka 150:s millisekunti.
Print(aika);
}
while(bit_is_clear(PIND,2));

lippu0 = 0;                     // Nollataan lippu0
}
wdt_reset();                    // Resetoidaan watchdog
return aika;
}
int laske1()
        {
if(lippu1)                      // keskeytys int1:sta = aikaa vähemmäksi
{
Do                              // Niinkauan kuin nappi on painettu, vähennetään
                                aikaa
{
aika--;
_delay_ms(150);                 // Tähänkin 150ms viive
Print(aika);
}
while(bit_is_clear(PIND,3));
```

```
lippu1 = 0;                     // Nollataan lippu1
}
wdt_reset();                    // Resetoidaan watchdog
return aika;
}

ISR(INT0_vect)                  // Asetetaan lippu0, jos tulee kesketys (INT0)
{
lippu0 = 1;
}

ISR(INT1_vect)                  // Asetetaan lippu1, jos tulee keskeyty (INT1)
{
lippu1 = 1;
}

ISR(TIMER2_COMP_vect)           // Sekuntien laskun keskeytysvektori
{
sekunti++;                      // lisätään joka sekunti yksi sekunti
if(sekunti >= 60)               // Kun kulunut 60 sekuntia = 1 minuutti.
{
aika--;                         // Vähennetään aikaa yhdellä.
sekunti = 0;                    // Nollataan sekunnit.
}
```

## Liite 3: Näytön ohjaamiseen käytetty koodi

```
/*
7-segmenttinäytön ohjaus.
Muokattu Avinash Gupta:n koodista
http://extremeelectronics.co.in/avr-tutorials/multiplexed-seven-segment-displays-part-ii/
luettu 28.2.2010
ATMEGA32, 8MHZ.
*/

#include <avr/io.h>
#include <avr/interrupt.h>

#define SEVEN_SEGMENT_PORT PORTA
#define SEVEN_SEGMENT_DDR DDRA

volatile uint8_t digits[3];
void SevenSegment(uint8_t n,uint8_t dp)
{
/*
Kirjoitetaan n:n määräämä numero näytölle.
Desimaalipiste käytössä, jos dp = 1.
*/
if(n<10)
{
switch (n)
{
case 0:
SEVEN_SEGMENT_PORT=0b00000011;
break;
case 1:
SEVEN_SEGMENT_PORT=0b10011111;
break;
case 2:
SEVEN_SEGMENT_PORT=0b00100101;
break;
case 3:
SEVEN_SEGMENT_PORT=0b00001101;
 break;
case 4:
SEVEN_SEGMENT_PORT=0b10011001;
```

```
break;
case 5:
SEVEN_SEGMENT_PORT=0b01001001;
break;
case 6:
SEVEN_SEGMENT_PORT=0b01000001;
break;
case 7:
SEVEN_SEGMENT_PORT=0b00011111;
break;
case 8:
SEVEN_SEGMENT_PORT=0b00000001;
break;
case 9:
SEVEN_SEGMENT_PORT=0b00001001;
break;
}
if(dp)
{
SEVEN_SEGMENT_PORT&=0b11111110;          // Desimaalipilkun kirjoitus, ei tarvita
}
}
else
{
SEVEN_SEGMENT_PORT=0b11111101;
                                          // Jos n suurempi kuin 9 = ei sovi näyttöön
}
}
void Print(uint16_t num)
{
/*
Pilkotaan aika osiin ja kirjoitetaan kukin omalle näytölleen.
*/
uint8_t i=0;
uint8_t j;
if(num>999) return;

while(num)
{
digits[i]=num%10;
i++;
```

```
num=num/10;
}
for(j=i;j<3;j++) digits[j]=0;
}
void init_naytto()
{
TCCR0|=(1<<CS02);                       // Timerin 0 jakajaksi 1024
TIMSK|=(1<<TOIE0);                      // Ylivuotokeskeytys (Overflow Interrupt
                                        Enable) päälle
TCNT0 = 0;                                      // Alustetaan laskuri
DDRB |= (1 << 0) | (1 << 1) | (1 << 2);         // Portin B bitit 0, 1 ja 2 ulostuloiksi
SEVEN_SEGMENT_DDR=0XFF;                         // Kaikki segmentit pois päältä.
SEVEN_SEGMENT_PORT=0XFF;
sei();                                          // Globaalit keskeytykset päälle.
}
ISR(TIMER0_OVF_vect)
{
/*
Päivitetään näyttöä.
*/
static uint8_t i=0;
if(i==2)                        // Jos ollaan kolmannessa merkissä
{
i=0;                            // Palataan takaisin ensimmäiseen
}
else
{
i++;                            // Siirrytään seuraavaan näyttöön
}

PORTB=~(1<<i);                  // Aktivoidaan i:n ilmoittamana näyttö.

SevenSegment(digits[i],0);      // Kirjoitetaan numero näytölle i
}
```