OULUN SEUDUN
AMMATTIKORKEAKOULU

Kai Käräjäoja

3D-LEDINÄYTTÖ

3D-LEDINÄYTTÖ

Kai Käräjäoja
Opinnäytetyö
Syksy 2012
Tietototekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, elektroniikan suunnittelu ja testaus

---

Tekijä: Kai Käräjäoja
Opinnäytetyön nimi: 3D-ledinäyttö
Työn ohjaaja: Ensio Sieppi
Työn valmistumislukukausi ja -vuosi: syksy 2012 Sivumäärä: 44 + 3 liitettä

Opinnäytetyön aiheena oli tutkia mahdollisuutta rakentaa kolmiulotteista kuvaa ja animaatiota esittävä ledinäyttö. Työn tavoitteena oli rakentaa toimiva prototyyppi, joka näyttäisi liikkuvaa kuvaa.

Työ jaettiin kahteen osaan. Ensimmäinen osa käsitti prototyypin kytkentöjen suunnittelun sekä sen rakentamisen. Toisessa osassa kehitettiin ohjelmakoodi laitteen ytimenä toimivaan Atmel-mikrokontrolleriin. 3D-näyttö kehitettiin POV-näyttöperiaatteella. Siinä pientä määrää ledejä pyöritetään akselin ympäri, jolloin tarkalla ajoituksella ledien sytyttämisessä saadaan muodostettua paljon suurempi määrä ns. virtuaaliledejä.

Tulokseksi saatiin toimiva kolmiulotteinen ledinäyttö. Lopputulos toteutettiin suunnittelemalla ensin kaksiulotteinen ledimatriisinäyttö, jota pyöritettiin yhden akselin ympäri. Näin saatiin muodostettua sylinterimäinen kuva tai animaatio. Lopullisessa animaatiossa on kaksi mailaa, jotka lyövät palloa toisilleen. Pallon liikkuessa edestakaisin se vuorotellen muodostaa ilmaan sanan ”3D” ja pyyhkii sen.

Asiasanat: 3D, ledimatriisi, mikrokontrolleri, POV-näyttö

ABSTRACT

Oulu University of Applied Sciences
Degree Programme of Information Technology, Electronics Design and Testing

---

Author: Kai Käräjäoja
Title of thesis: 3D Led Display
Supervisor: Ensio Sieppi
Term and year of completion: autumn 2012 Pages: 44 + 3 appendices

---

The aim of Bachelor's thesis was to research if is it possible to make a display which could show 3D figures and animations. The aim was to make a working prototype which would be able to show animation.

The work was divided into two parts. The first part consisted of drawing schematics for a prototype and it was made. The second part was a written program code for Atmel-microcontroller which worked as the main unit of the prototype. 3D display was developed by using the principle of the POV-display. In there small amount of leds was rotated around one shaft. When the leds are driven with the exact timing it is possible to form a lot of bigger display. Those leds can be called virtual leds.

The result was successful. The prototype was able to show a 3D animation. It was made by first developing a 2D led-matrix-display which was rotated around one shaft. By using that principle a cylindrical figure or animation was shown. In the final animation there are two paddles which hit one ball to each other forming a text "3D" to the air at the same time.

Keywords: 3D, led-matrix, microcontroller, POV-display

ALKULAUSE

Haluaisin kiittää toimeksiantajaani, Oulun Yritystakomon Juha Lumilaa erittäin mielenkiintoisesta ja juuri minunlaisestani opinnäytetyöaiheesta. Työhön kuului sulautetun järjestelmän suunnittelua, sen rakentamista ja ohjelmakoodin kirjoittamista sille, juuri sellaisia asioita, joiden tekemisestä pidän paljon ja joita haluan tehdä tulevaisuudessakin.

Koulun puolelta haluan kiittää työn ohjaajaa Ensio Sieppiä kannustavasta ohjauksesta.

Suuret kiitokset myös perheelleni, vaimo Neealle sekä pojillemme Otolle ja Jasulle, jotka ovat jaksaneet tukea opinnäytetyöprosessin aikana.

Oulussa syyskuussa 2012

Kai Käräjäoja

SISÄLLYS

1 JOHDANTO

Opinnäytetyön ideana oli selvittää, onko mahdollista kehittää 3D-näyttölaitetta. Lähtökohtana pidettiin POV-näyttöperiaatetta, jossa pientä määrää ledejä pyöritetään akselin ympäri. Tällöin ledit muodostavat suuremman pikselimatriisin, jonka jokaista pikseliä pystytään ohjaamaan erikseen. Kun tehdään ensin 2D-ledimatriisinäyttö ja sitä pyöritetään ympäri, pitäisi laitteen näin ollen pystyä esittämään kolmiulotteisia kuvia ja animaatioita.

Osa työn tiedoista on työn tilaajan kanssa tehdyn salassapitosopimuksen mukaan luottamuksellisia. Näin ollen osa tiedoista löytyy vain työn tilaajalle menevästä versiosta.

2 IHMISEN NÄKÖKYKY

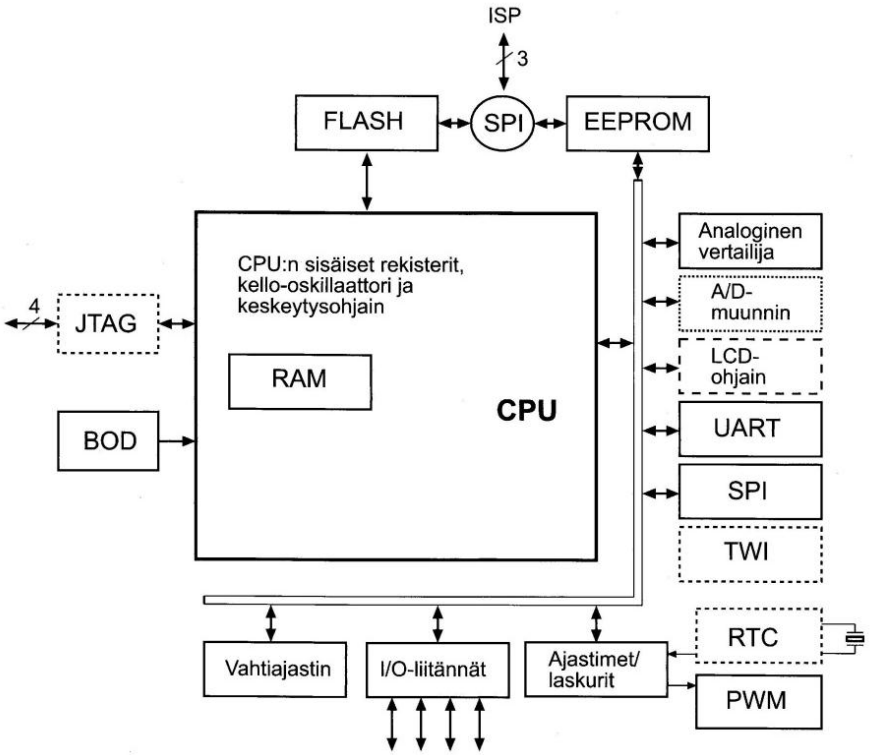
Kuten kaikissa näyttölaitteissa, tässäkin työssä on otettava huomioon eräs ihmissilmän ominaisuuksista: ihmissilmän kyky nähdä liikkuvaa kuvaa. Sitä on tutkittu paljon viimeistään varhaisten videoiden kehittämisestä lähtien. Tärkeimpänä tutkimisen kohteena on ollut videoiden fps:n (frames per second = kehyksiä sekunnissa) eli kuvanopeuden vaikutus katselukokemukseen. Liian hidas kuvanopeus näkyy nykivästi vaihtuvina kuvina ihmissilmään. Toisin sanoen ihminen ehtii huomaamaan yksittäiset kuvat ja pienen viiveen niiden vaihtamisen välissä. Jo varhaisessa vaiheessa on huomattu, että noin 25 fps riittää nykimättömän kuvan näyttämiseen. (1.)

Mikä sitten on riittävä kuvanopeus ihmissilmän huijaamiseen? Tässä asiassa vallitsee kaksi erilaista käsitystä. Ensimmäisen mukaan ihminen pystyy erottamaan vain noin 30 fps, jolloin ei ole mitään hyötyä nostaa kuvataajuutta kun ihmissilmä ei kuitenkaan huomaa eroa. Toisen käsityksen mukaan ihminen pystyy erottamaan jopa 200 fps. Uskon molempien olevan osaltaan oikeassa; videon nykiminen loppuu noin 25–30 fps:n kohdalla ja siitä korkeampi kuvataajuus on kuin hienosäätöä, jolla kuva saadaan entistä sulavammaksi. (2.)

3 ATMELIN AVR-MIKROKONTROLLERIT

3.1 Yleistä

Kaikilla AVR-mikrokontrollereilla on samanlainen ydin. Eroja on muistin koossa, I/O-liitäntöjen määrässä sekä erikoisominaisuuksissa. Kuvan 1 lohkokaaviossa näkyy mikrokontrollereiden sisäinen rakenne. Katkoviivoilla piirretyt laatikot kuvaavat sellaisia ominaisuuksia, joita ei kaikissa malleissa ole. Kontrollerit on jaettu niiden sisältämien ominaisuuksien mukaan neljään eri ryhmään, jotka ovat AVR, tinyAVR, megaAVR sekä XMEGA. (3, s. 137–138.)

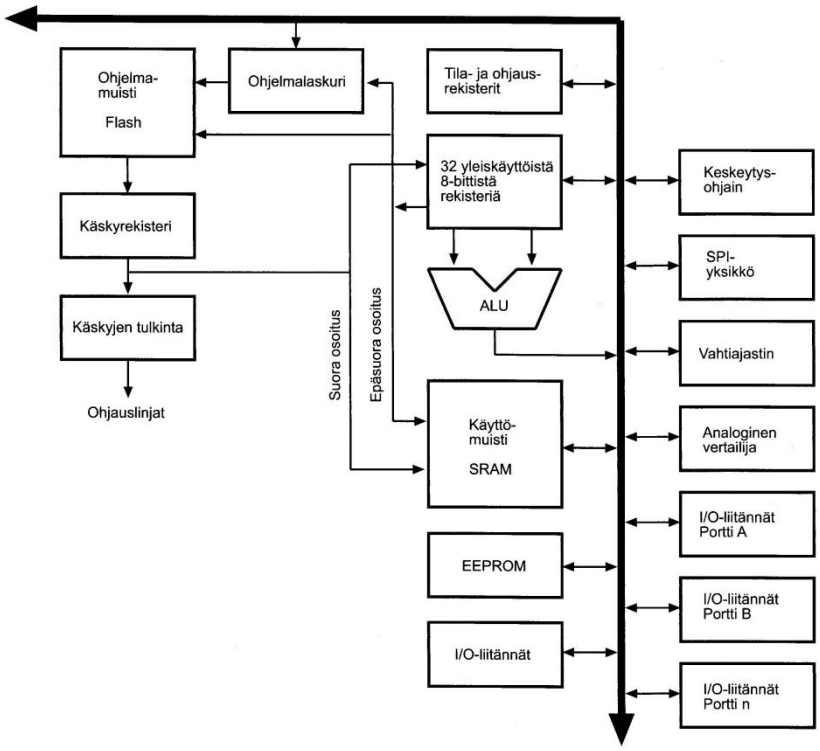


KUVA 1. AVR-mikrokontrollereiden sisäinen rakenne (3, s. 137)

TinyAVR-sarja sisältää nimensä mukaisesti pieniä kontrollereita eli niissä on vain 8–20-jalkaiset kotelot ja vähän muistia. Sarja on suunniteltu sovelluksiin, joissa virrankulutus on oltava mahdollisimman pieni. Toisena äärilaitana on megaAVR-sarja, joka sisältää jopa 100-jalkaisia kontrollereita. Lisäksi niissä on paljon muistia sekä ne voivat sisältää erikoisominaisuuksia, joita ei ole perusmalleissa. AVR-sarja on alkuperäinen ja ensimmäinen AVR-tekniikalla kehitetty ja se sisältää perusmallit. Se sijoittuukin tinyAVR- ja megaAVR-sarjojen väliin

ominaisuuksiensa puolesta. Näiden lisäksi nykyään on myös XMEGA-sarja, joka on kehitetty vieläkin vaativampaan käyttöön. XMEGA-kontrollerin kellotaajuus voi olla jopa 32 MHz. (3, s. 138; 4.)

AVR-mikrokontrollereiden arkkitehtuuri on suunniteltu niin, että erityisesti C-kielestä käännetty ohjelma pystytään suorittamaan mahdollisimman tehokkaasti. AVR-mikrokontrollereiden arkkitehtuuri näkyy kuvassa 2. Prosessorina toimii RISC-tyyppinen ydin. (3, s.141.)



KUVA 2. AVR-mikrokontrollerin arkkitehtuuri (3, s. 142)

3.2 AVR-tekniikka

3.2.1 Kellolähde

Kellolähteeksi AVR-mikrokontrollereissa on muutamia vaihtoehtoja. Haluttu lähde asetetaan niin sanottujen fuse-bittien avulla. Yleensä kontrollereissa on alkuperäisenä asetuksena otettu käyttöön sisäinen 8 MHz:n kello, joka on CKDIV8-bitin avulla hidastettu 1 MHz:iin. Tarvittaessa kellolähde sekä toimintatila muutetaan CKSEL3..1 -biteillä ja sen nousuaika CKSEL0- ja SUT1..0 -biteillä. Eri kellolähteiden vaihtoehdot näkyvät taulukossa 1. Taulukoissa 2 ja 3

on esitetty tarkemmin tässä työssä käytetyn nopean ulkoisen oskillaattorin eri asetusvaihtoehdot. Kellolähteen asetukset saattavat poiketa hieman mallista riippuen. (5, hakusana clock sources.)

TAULUKKO 1. Eri vaihtoehdot kontrollerin kellolähteeksi (5, hakusana table 7-1)

Table 7-1. Device Clocking Options Select[1]

Device Clocking Option	CKSEL3..0
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128 kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

TAULUKKO 2. Tässä työssä käytetyn oskillaattorin toimintataajuuden valinta (5, hakusana table 7-5)

Table 7-5. Full Swing Crystal Oscillator operating modes[2]

Frequency Range[1] (MHz)	CKSEL3..1	Recommended Range for Capacitors C1 and C2 (pF)
0.4 - 20	011	12 - 22

Nousuajalla tarkoitetaan aikaa, joka kontrolleri pidetään resetointitilassa sitä käynnistettäessä. Tällä varmistetaan, että käyttöjännite ehtii nousta kontrollerin vaatimaan minimiin asti. Jos aika on liian lyhyt ja käyttöjännite ei ehdi nousta tarpeeksi, niin kontrolleri ei välttämättä käynnisty ollenkaan. Taulukossa 3 on esitelty eri nousuajan vaihtoehdot. (5, hakusana clock startup sequence.)

TAULUKKO 3. Eri vaihtoehdot nousuajan valinnaksi (5, hakusana table 7-6)

Table 7-6. Start-up Times for the Full Swing Crystal Oscillator Clock Selection

Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT1..0
Ceramic resonator, fast rising power	258 CK	14CK + 4.1 ms[1]	0	00
Ceramic resonator, slowly rising power	258 CK	14CK + 65 ms[1]	0	01
Ceramic resonator, BOD enabled	1K CK	14CK[2]	0	10
Ceramic resonator, fast rising power	1K CK	14CK + 4.1 ms[2]	0	11
Ceramic resonator, slowly rising power	1K CK	14CK + 65 ms[2]	1	00
Crystal Oscillator, BOD enabled	16K CK	14CK	1	01
Crystal Oscillator, fast rising power	16K CK	14CK + 4.1 ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14CK + 65 ms	1	11

3.2.2 I/O-liitännät

I/O-liitäntöjä AVR-kontrollereissa on mallista riippuen 3:sta jopa 54:ään. Ne ovat vapaasti asetettavissa joko tuloiksi tai lähdöiksi. Lähes kaikilla liitännöistä on myös jokin erityistoiminto, kuten esimerkiksi PWM-lähtö. Liitännät on aina jaettu portteihin, joista yhteen voi maksimissaan kuulua 8 liitäntää. (3, s. 148.)

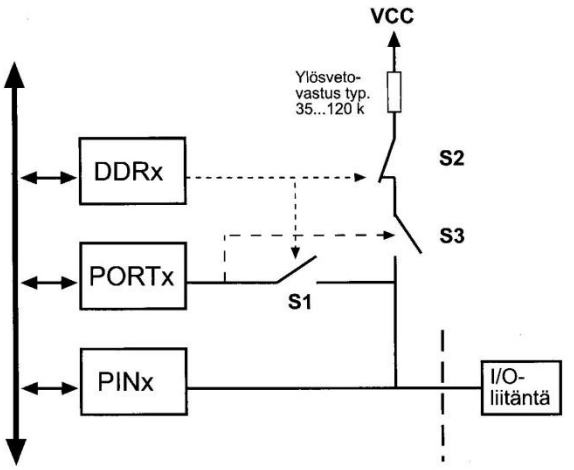
Jokaisen portin toimintaa ohjataan aina kolmella rekisterillä (taulukko 4), jotka PORTC:n tapauksessa ovat DDRC, PORTC ja PINC. DDRx-rekisterillä asetetaan liitännän suunta, joko tulo tai lähtö. Tulon tapauksessa PORTx-rekisterillä ohjataan ylösvetovastusta ja liitännän tila voidaan lukea PINx-rekisteristä. Lähdön tapauksessa PORTx-rekisterillä ohjataan liitännän tilaa. Taulukosta 5 selviää eri vaihtoehdot liitäntöjen alustamiselle ja kuvasta 3 näkyy periaate, miten rekisterien tilat vaikuttavat käytännössä kytkentään. (3, s. 148.)

TAULUKKO 4. I/O-liitäntöjen rekisterien merkitykset (3, s. 148)

Rekisteri	Merkitys
DDRx	tiedonsuuntarekisteri
PORTx	lähtöportin tietorekisteri
PINx	tuloportin liitäntöjen tila

TAULUKKO 5. I/O-liitäntöjen eri alustusvaihtoehdot (3, s. 148)

DDRx	PORTx	Liitännän suunta	Ylösveto-vastus	Selite
0	0	Tulo	Ei	Tulo on suurim-pedanss. tilassa
0	1	Tulo	On	Tulo antaa hieman virtaa
1	0	Lähtö	Ei	Lähtö alatilassa
1	1	Lähtö	Ei	Lähtö ylätilassa

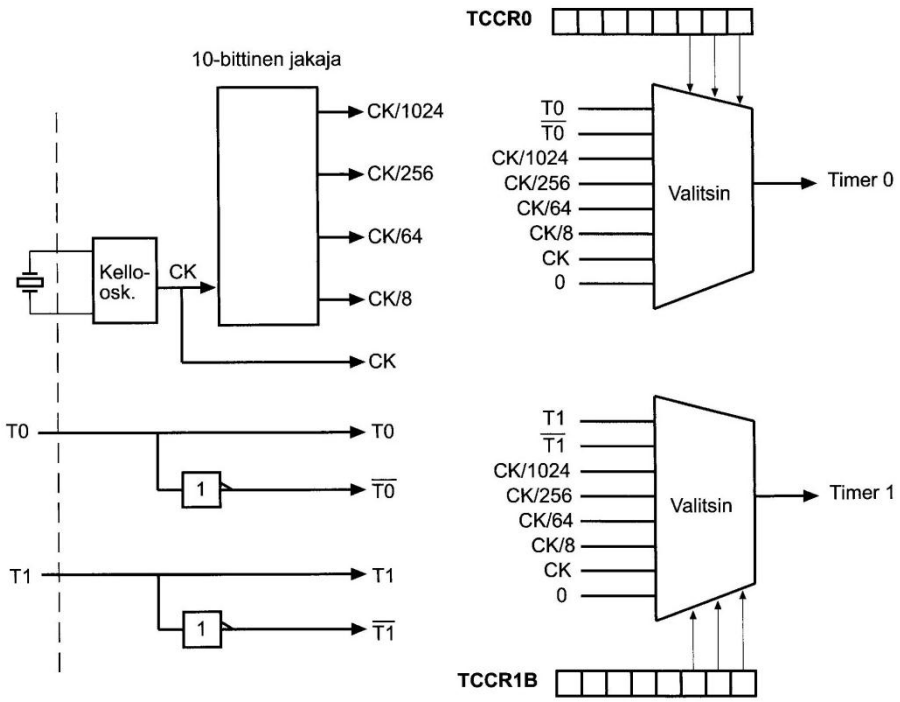


KUVA 3. I/O-liitännän toimintaperiaate (3, s. 148)

3.2.3 Laskurit

Suurimmassa osassa AVR-mikrokontrollereita on vähintään sekä 8-bittinen että 16-bittinen laskuri. 8-bittisellä voidaan laskea joko sisäisiä kellopulsseja tai ulkoiseen liitäntänastaan tulevia pulsseja. 16-bittinen laskuri on paljon monipuolisempi ja se voi toimia myös vertailutilassa, pulssileveysmodulaattorina tai sieppaustilassa. Laskentakapasiteetti on 8-bittisellä 0–255 pulssia ja 16-bittisellä 0–65 535 pulssia. (3, s. 150–151.)

Jotta laskurin laskentaväliä saataisiin skaalattua suuremmaksi, on siihen lisätty esijakaja. Se jakaa kellopulsseja eri luvuilla. Esijakajan ja laskentapulssien lähteen valitsimen toimintaperiaate näkyy kuvassa 4. T0 ja T1 tarkoittavat I/O-liitäntää, jotka myös voidaan asettaa laskentapulssien lähteeksi. Laskuri 0:n tapauksessa lähde valitaan TCCR0-rekisterissä (kuva 5) sijaitsevilla kolmella CS-bitillä (Clock Select). Eri vaihtoehdot lähteeksi on lueteltu taulukkoon 6. CS-bittien ollessa '0' laskuri on pysähtyneenä ja se käynnistetään asettamalla jokin muista vaihtoehdoista. (3, s. 150.)



KUVA 4. Esijakajan ja valitsimen toimintaperiaate (3, s. 150)

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

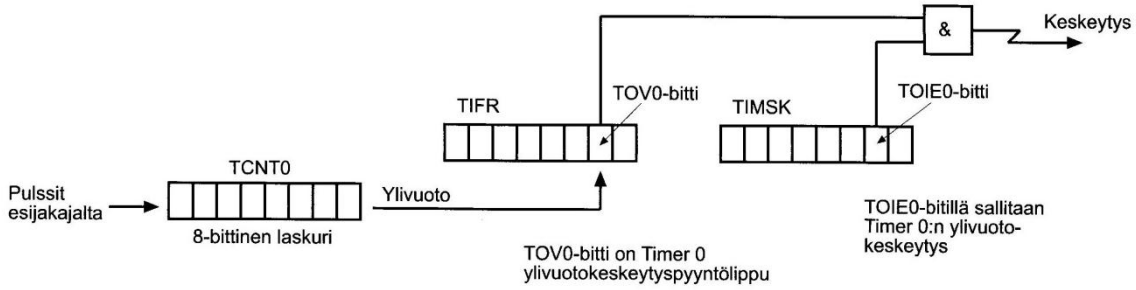
KUVA 5. TCCR0B-rekisterin sisältö (5, hakusana TCCR0B)

TAULUKKO 6. Eri vaihtoehdot laskentapulssien lähteeksi (5, hakusana table 14-6)

Table 14-6. Clock Select Bit Description

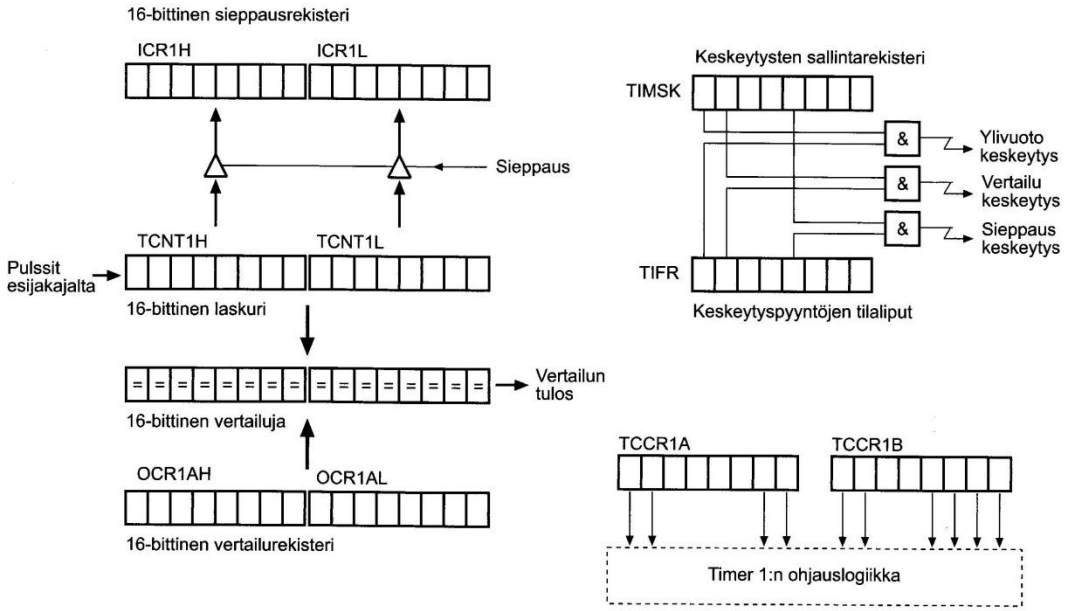
CSn2	CSn1	CSn0	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge.
1	1	1	External clock source on Tn pin. Clock on rising edge.

Kuvassa 6 on havainnollistettu 8-bittisten laskurien toimintaperiaate. Pulssit valitusta lähteestä kasvattavat TCNT0-rekisterin arvoa. Kun sen arvo on 255, niin seuraavan pulssin tullessa rekisteri pyörähtää ympäri ja aloittaa laskemisen jälleen alusta. Ympäri pyörähtäminen eli ylivuoto asettaa TIFR-rekisterin TOV0-bitin (Timer Overflow) '1'-tilaan. Jos TIMSK-rekisterin TOIE0-bitti sekä yleinen keskeytysten sallinta ovat asetettu aktiivisiksi, niin aiheutuu keskeytys ja ohjelma siirtyy suorittamaan sitä varten kirjoitettua keskeytysohjelmaa.



KUVA 6. Laskuri 0:n toimintaperiaate (3, s. 151)

16-bittinen Laskuri 1 voi toimia myös sieppaus- tai vertailutilassa sekä pulssileveysmodulaattorina (kuva 7), mutta tämän työn puitteissa niitä ei ole tarpeen käsitellä tarkemmin. Työssä Laskuri 1 toimii kuten Laskuri 0:kin, vain sillä erolla, ettei se aiheuta keskeytystä. Se siis vain pelkästään laskee pulsseja TCNT1-rekisteriin, joka on jaettu kahteen 8-bittiseen osaan. Tietyllä hetkellä rekisterin arvo sitten tallennetaan taulukkoon.



KUVA 7. Laskuri 1:n eri toimintavaihtoehdot (3, s. 151)

3.2.4 Keskeytykset

Erilaisia mahdollisuuksia keskeytyksen lähteeksi AVR-mikrokontrollereissa on todella monia. Edellisessä kappaleessa mainitun laskurikeskeytyksen lisäksi muita lähteitä ovat esimerkiksi ulkoiset keskeytykset tietyissä pinneissä, vahti-koira-ajastin ja keskeytykset tiedonsiirtoväylien sekä analogisen vertailijan tie-tyistä toiminnoista. Tässä työssä käytetään laskurikeskeytyksen lisäksi optohaa-rukalta tulevaa ulkoista keskeytystä.

Laskureissa tärkeässä osassa on TIMSK-rekisteri (Timer/Counter Interrupt Mask Register). Sen sisältämä TOIE-bitti täytyy asettaa ylätilaan, jos haluaa laskurin aiheuttavan keskeytyksen sen pyörähtäessä ympäri. Kuvassa 8 näkyy 8-bittisen Laskuri 0:n TIMSK0-rekisterin sisältö.

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 8. Laskuri 0:n TIMSK-rekisterin sisältö (5, hakusana TIMSK0)

Ulkoisia keskeytyksiä käytettäessä täytyy EICRA- (kuva 9) ja EIMSK-rekisterien (kuva 10) bittejä asettaa halutun toiminnan aikaan saamiseksi. EICRA-rekisterin

biteillä asetetaan keskeytykselle haluttu toimintatila ja keskeytys sallitaan EIMSK-rekisteristä. Ulkoisen keskeytyksen eri toimintavaihtoehdot ovat luetel-tuna taulukkoon 7.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 9. EICRA-rekisterin sisältö (5, hakusana EICRA)

TAULUKKO 7. Eri toimintavaihtoehdot ulkoisille keskeytyksille (5, hakusana table 11-1)

Table 11-1. Interrupt Sense Control(1)

ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any edge of INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

Note: 1. n = 3, 2, 1or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	INT2	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

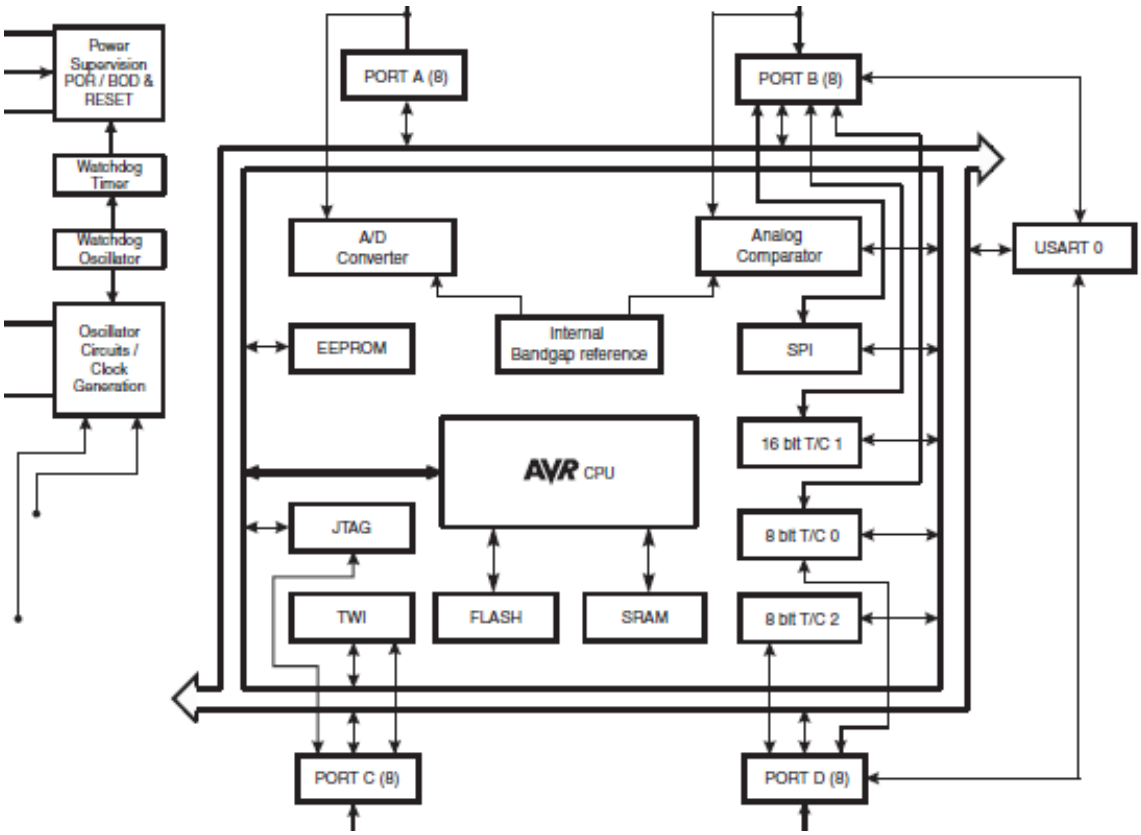
KUVA 10. EIMSK-rekisterin sisältö (5, hakusana EIMSK)

3.3 ATmega644-mikrokontrolleri

Kontrolleri kuuluu Atmelin ATmega-sarjaan, joten siitä löytyy todella paljon eri ominaisuuksia. Tätä työtä ajatellen niistä oleellisimpia ovat kuitenkin

- jopa 20 MHz:n kellotaajuus
- 64 kB FLASH-muistia
- 32 ohjelmoitavaa I/O-linjaa
- 2 kpl 8-bittisiä laskuria
- 1 kpl 16-bittisiä laskuria (5).

Täydelliset tiedot kontrollerista löytyvät liitteestä 1. Kuvassa 11 on esitetty kontrollerin koko sisäinen rakenne.



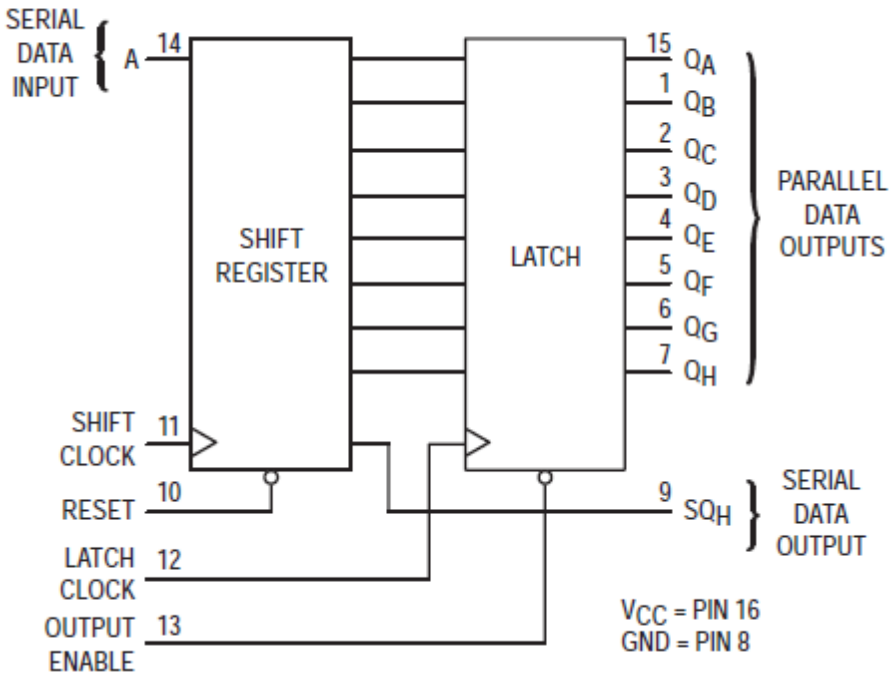
KUVA 11. ATmega644-mikrokontrollerin lohkokaavio (5, hakusana block diagram).

4 LED-MATRIISI

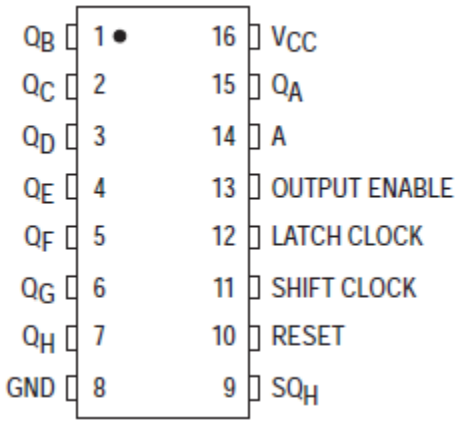
Olennainen osa 3D-ledinäyttöä on 8x8-ledimatriisi, jota mikrokontrollerilla ohjataan. Seuraavissa kappaleissa esitellään matriisin rakentamiseen käytetyt komponentit sekä selvitetään, millä periaatteella sitä ohjataan.

4.1 74HC595-siirtorekisteripiiri

74HC595-piiri koostuu 8-bittisestä siirtorekisteristä sekä 8-bittisestä D-tyypin lukkopiiristä. Data viedään sarjamuotoisena siirtorekisteriin, joka antaa sen rinnakkaismuodossa lukkopiirille. Siirtorekisterin sarjamuotoisen lähdön avulla piirejä voidaan kytkeä monta peräkkäin. Siirtorekisterillä ja lukkopiirillä on omat kellotulonsa, joilla datan liikkumista hallitaan. Kuvissa 12 ja 13 on esitetty 74HC595-piirin lohkokaavio sekä pinnijärjestys. (6.)



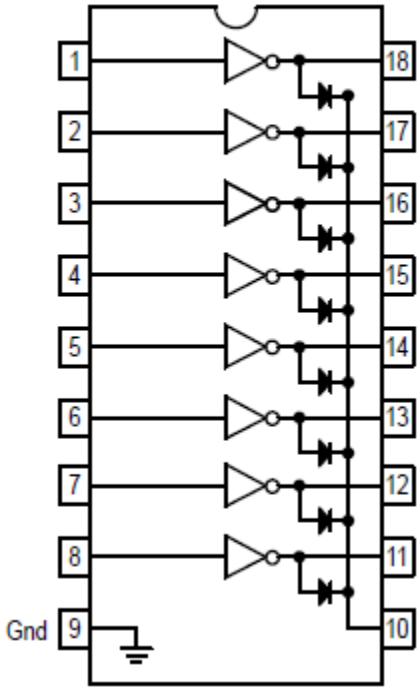
KUVA 12. 74HC595-piirin lohkokaavio (6)



KUVA 13. 74HC595-piirin pinnijärjestys (6)

4.2 ULN2803-puskuripiiri

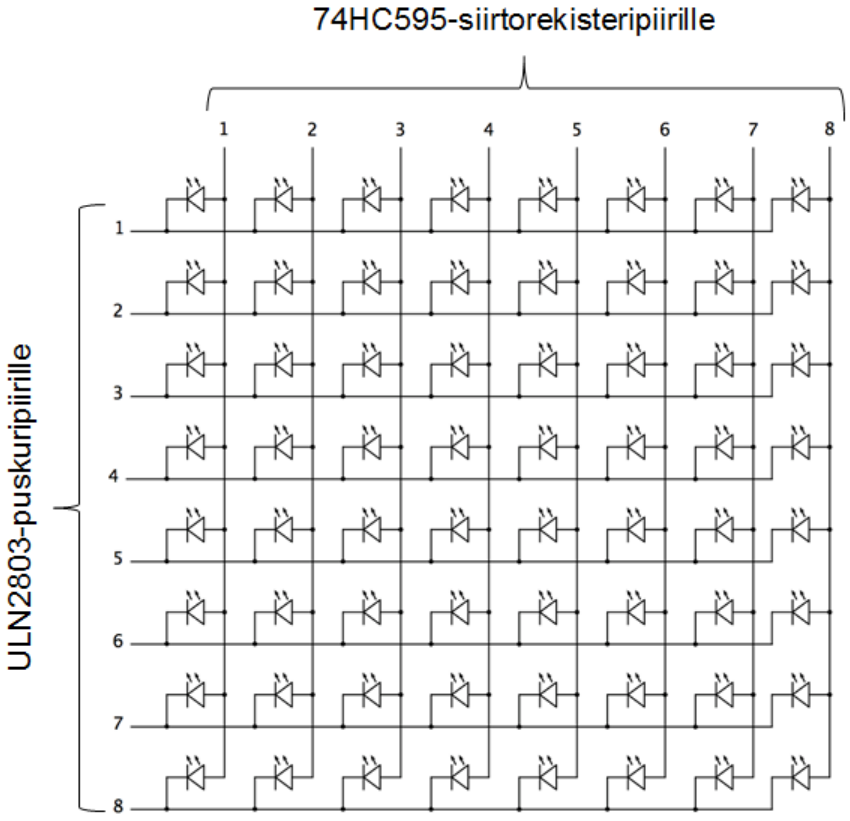
ULN2803-piiri sisältää kahdeksan Darlington-transistoria. Niiden avulla lähtöihin saadaan puskuroitua enemmän virtaa, jolloin lähdöillä voidaan ohjata suoraan esimerkiksi lamppuja tai releitä. Yhdellä lähdöllä on mahdollista ohjata jopa 500 mA:n kuormaa, kun yleensä mikropiirien lähdöillä se on noin 20 mA:n luokkaa. Kuvasta 14 selviää piirin kytkentä sekä pinnijärjestys. (7.)



KUVA 14. ULN2803-piirin kytkentä ja pinnijärjestys (7)

4.3 Ledimatriisin toiminta

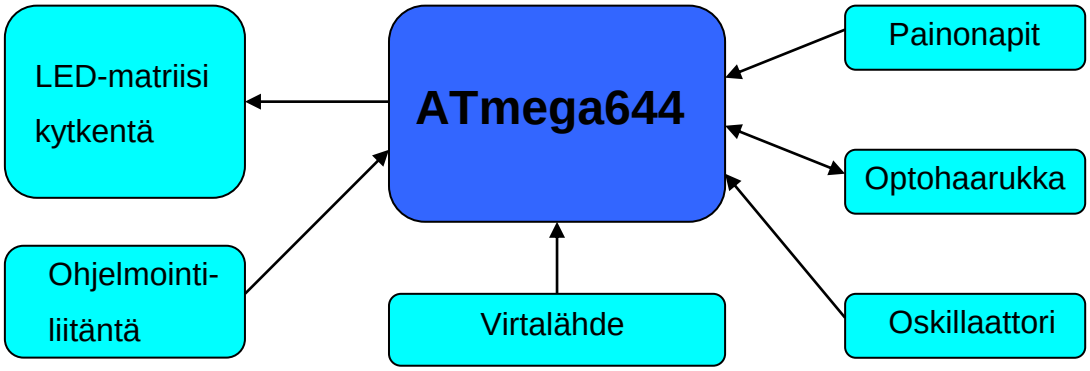
Ledimatriisia ohjataan niin sanotulla skannausperiaatteella. Siinä mikrokontrollerilta lähetetään sarakkeille 8-bittinen data puskuripiirin kautta. Data sisältää kahdeksan arvoa, jotka joko sytyttävät tai sammuttavat yksittäisen ledin. Siirtorekisterin avulla vain yhtä saraketta pidetään aktiivisena kerrallaan. Aktiivisena olevaa saraketta vaihdellaan tarpeeksi nopealla taajuudella, jolloin syntyy vaikutelma, että kaikki matriisin ledit palavat yhtä aikaa. Todellisuudessa ledeistä palaa maksimissaan kahdeksan yhdellä ajanhetkellä eli yhden sarakkeen verran. Ledimatriisin kytkentä selviää kuvasta 15.



KUVA 15. Ledimatriisin kytkentä.

5 KYTKENNÄN SUUNNITTELU JA TOTEUTUS

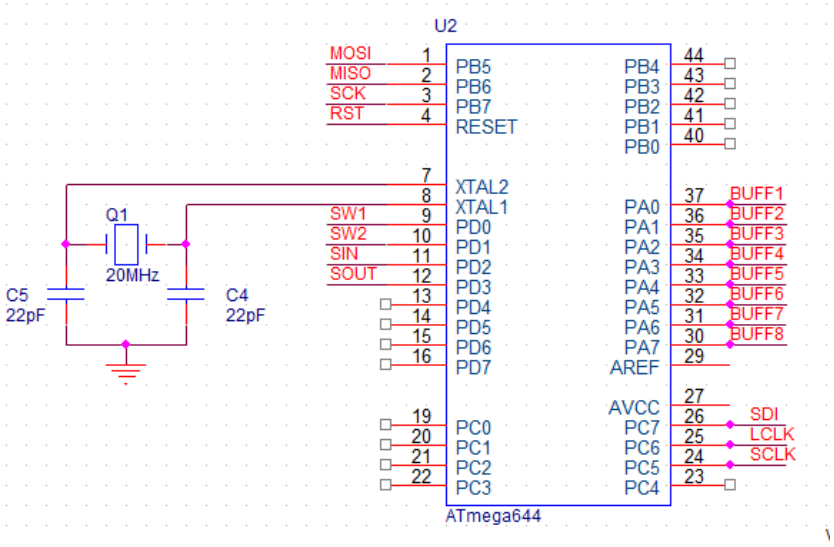
Aluksi piirrettiin lohkokaavio (kuva 16) laitteeseen tulevista osista. Tämän pohjalta alettiin suunnitella kytkentää kaikkine komponentteineen. Kytkentöjen suunnitteluun käytettiin Cadence Orcad -ohjelmiston Capture-osaa. Prototyypin piirilevyt piirrettiin saman ohjelmiston Layout-osalla. Piirilevyjen täydelliset kuvat löytyvät liitteestä 3.



KUVA 16. Lohkokaavio laitteen eri osista.

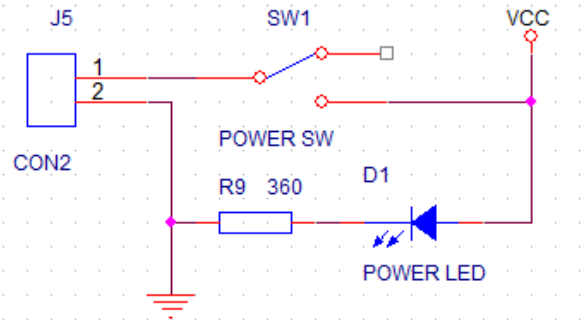
5.1 Ohjauselektroniikka

Mikrokontrollerin valinnassa tärkeänä kriteerinä oli sen kellotaajuus. Sen piti olla riittävän nopea pystyäkseen vilkuttamaan ledejä todella nopeaan tahtiin pyöritettäessä laitetta jopa 1500 kierrosta minuutissa. Mikrokontrollerin sisäisellä kellolla on mahdollista saavuttaa noin 8 MHz:n taajuus. Kytkentään valittiin kuitenkin ulkoinen 20 MHz:n kide vakaamman ja nopeamman toiminnan vuoksi. Kidekytkentään tarvittiin myös 22 pF:n kondensaattorit. Kytkentä on esitetty kuvassa 17.



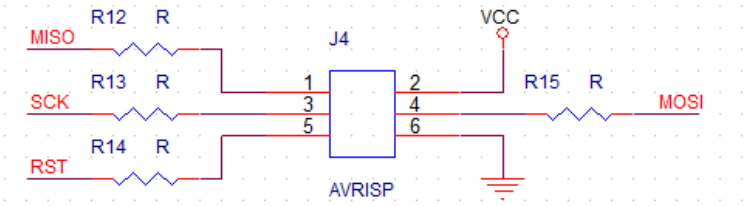
KUVA 17. Mikrokontrolleri ja oskillaattorikytkentä.

Prototyypistä haluttiin mahdollisimman yksinkertainen, joten virtalähteeksi valittiin akku Nokian puhelimesta. Akun malli oli BL-4U ja kapasiteetti 1000 mAh, joten siitä riittää virtaa pitkäksi aikaa. Tällöin vältyttiin helposti rikkoutuvien koskettimien rakentamiselta pyörivään laitteeseen. Kytkentään (kuva 18) lisättiin virtakytkin sekä ledi etuvastuksineen näyttämään, milloin laitteessa on virta päällä.



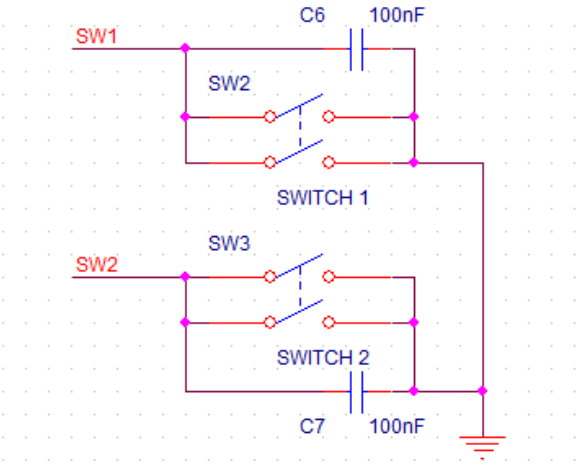
KUVA 18. Virtalähdekytkentä.

Ohjelman lataamista varten lisättiin Atmelin ohjeiden mukainen ohjelmointiliitin, johon AVRISP-ohjelmointilaite voidaan liittää. Kuvassa 19 on esitetty liittimen kytkentä.



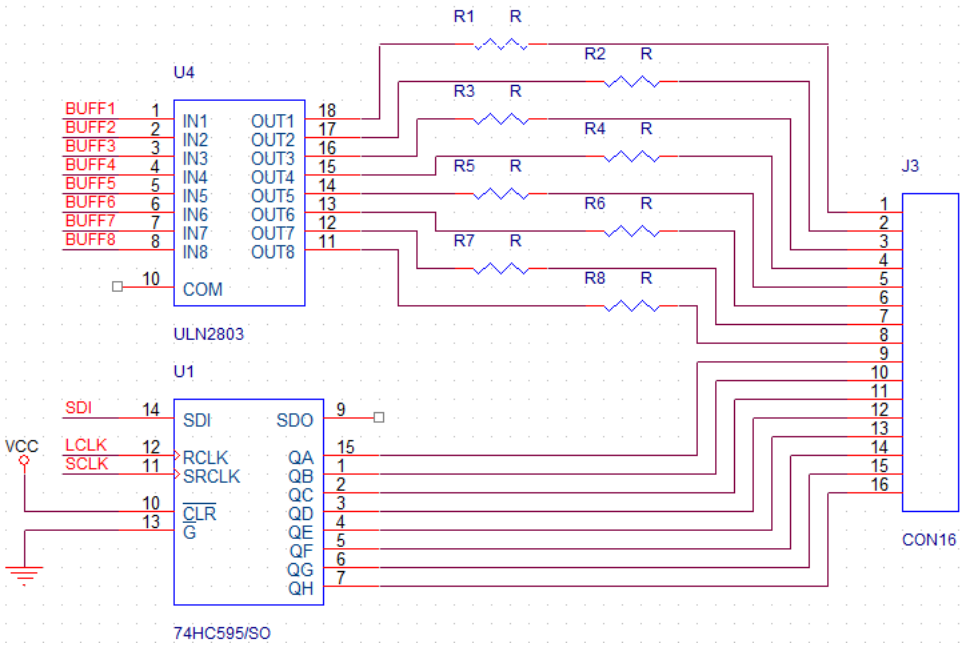
KUVA 19. Kytkentä AVRISP-ohjelmointilaitteelle.

Testikäyttöä varten prototyyppiin lisättiin kaksi painonappia. Niiden kanssa kytkettiin rinnakkain pienet kondensaattorit tasaamaan kytkinvärähtelyä, jolloin sitä ei tarvitse tehdä ohjelmallisesti. Kytkentä on esitetty kuvassa 20.



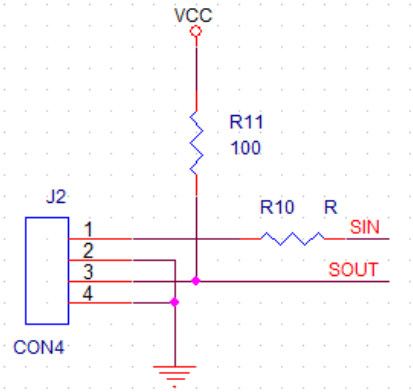
KUVA 20. Painonappien kytkentä.

Ledien ohjauskytkennässä käytettiin ULN2803-puskuripiiriä, josta menee etuvastusten kautta johtimet ledien katodeille. 74HC595-siirtorekisteripiirin lähdöistä menee johtimet ledien anodeille, yhdestä lähdöstä aina yhden sarakkeen ledien anodeille. Ohjauskytkentä on esitetty kuvassa 21.



KUVA 21. Ledien ohjauskytkentä.

Optohaarukka muodostuu infrapunaledistä sekä -vastaanottimesta. Kun lediin johdetaan virtaa ja haarukan välissä ei ole mitään estettä, vastaanotin antaa mikrokontrollerin tuloon PD3 arvon '0'. Kun haarukan väliin laitetaan jotain, arvo vaihtuu '1':een. Optohaarukan kytkentä selviää kuvasta 22.

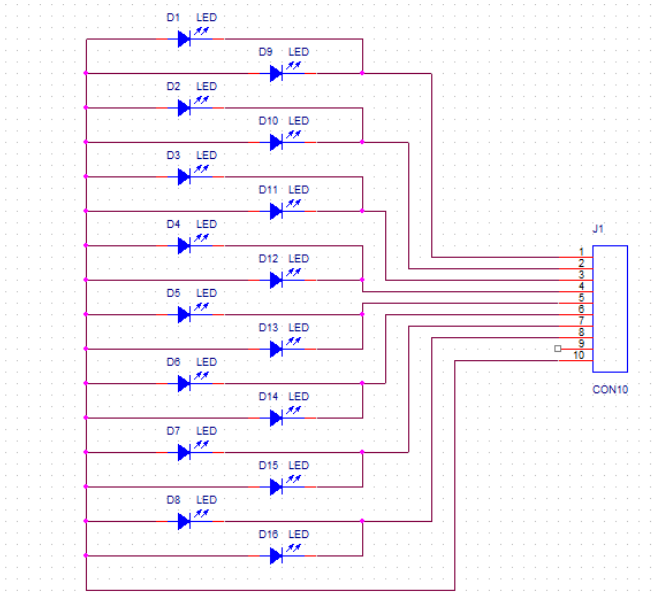


KUVA 22. Optohaarukan kytkentä.

5.2 Leditikku

Leditikkujen kytkentä (kuva 23) oli todella yksinkertaista toteuttaa, koska niihin ei tarvittu muita komponentteja kuin itse ledit ja liitin. Jokaiseen tikkuun tuli 16 lediä, 8 molemmille puolille. Tällä pyrittiin siihen, että kuva näkyisi selkeästi riip-

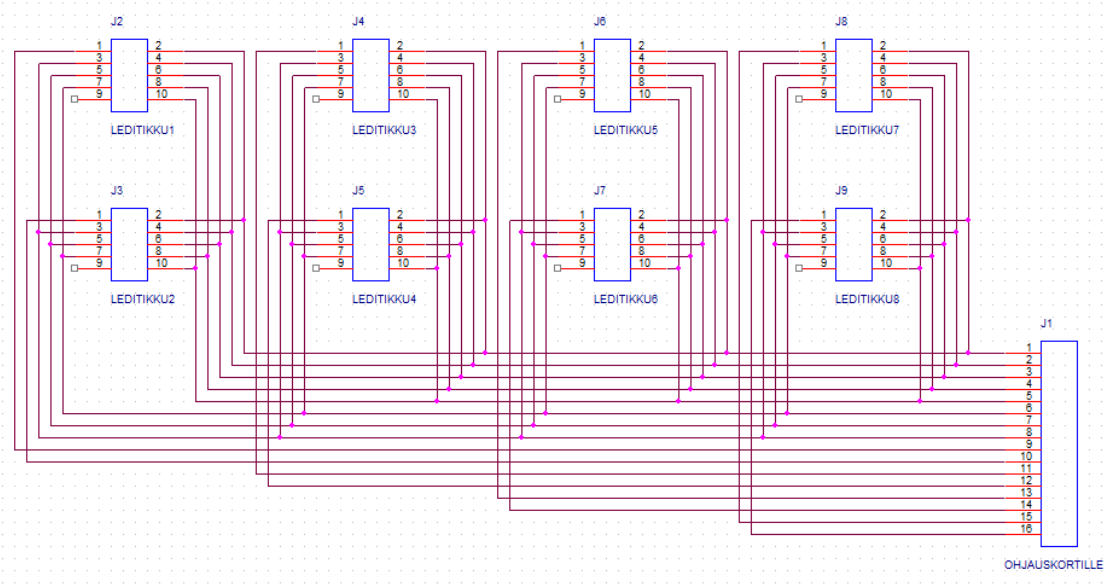
pumatta siitä, mistä päin sitä katsoo. Ledit kytkettiin CA-tyyppisesti (Common Anode), jolloin jokaisen sarakkeen ledien anodit kytketään yhteen.



KUVA 23. Ledien kytkentä.

5.3 Sovitin leditikuille

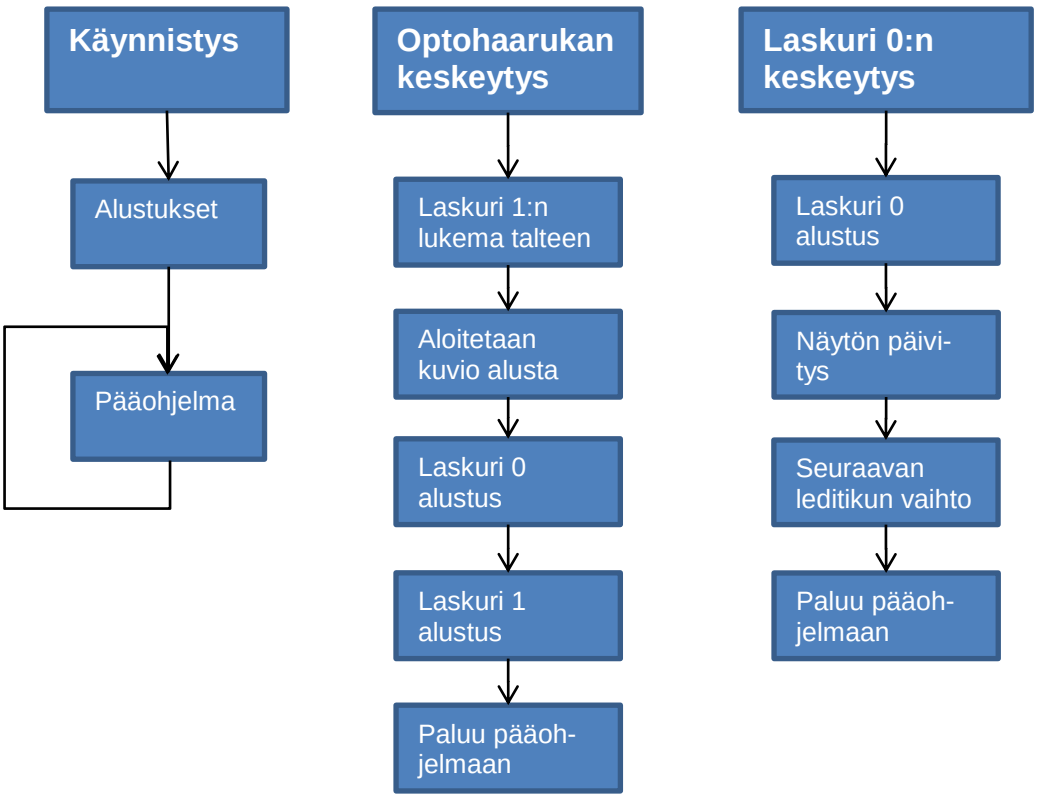
Leditikkujen liittämiseen suoraan ohjauskortille ei löydetty sopivia liittimiä, joten leditikkujen ja ohjauskortin väliin suunniteltiin erillinen sovitin.



KUVA 24. Sovitinkytkentä.

6 OHJELMISTO

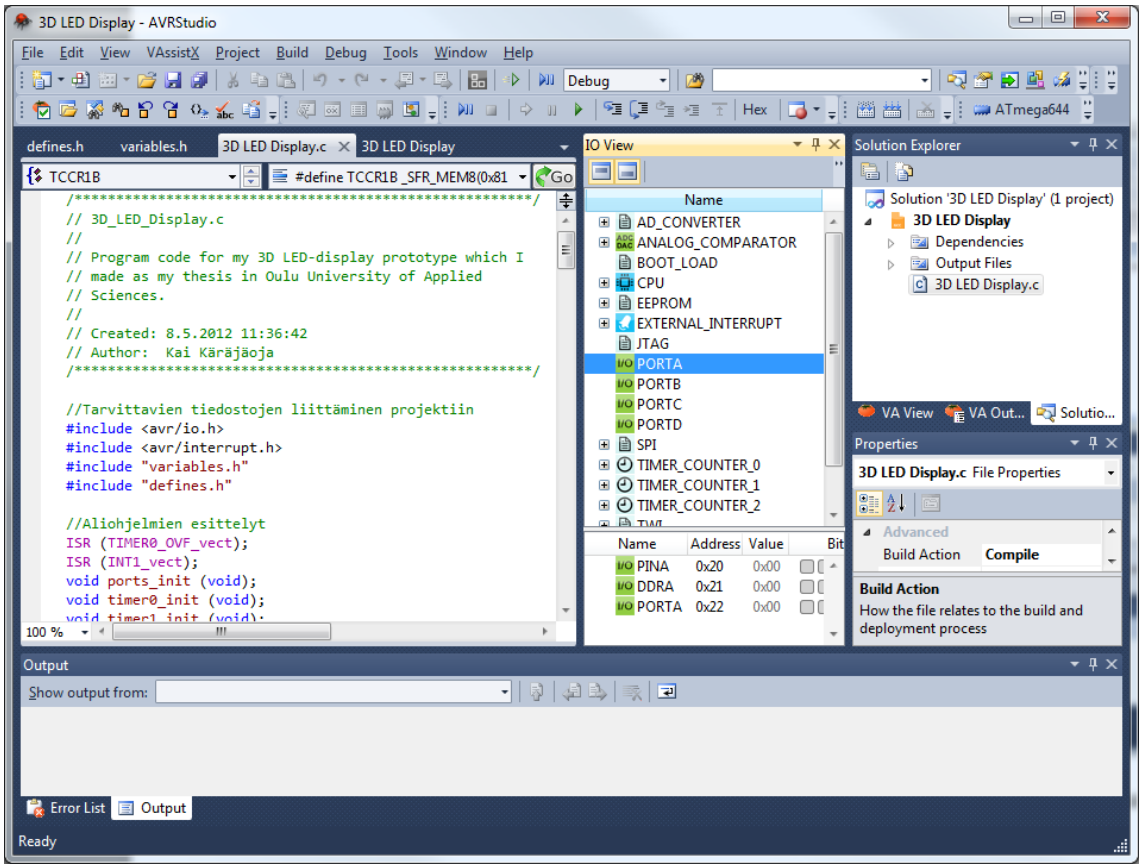
Ohjelman suunnitteleminen mikrokontrolleria varten aloitettiin hahmottamalla erilliset elementit lohkokaavioon (kuva 25), mistä siirryttiin yksityiskohtaisempiin tietoihin. Itse pääohjelman toistosilmukassa ei tehdä mitään, vaan kaikki toteutetaan keskeytysohjelmien avulla.



KUVA 25. Ohjelman karkea lohkokaavio.

6.1 Ohjelmankehitysympäristö

Ohjelmankehitysympäristönä käytettiin Atmelin omaa AVRstudiota (kuva 26), joka on ilmaiseksi ladattavissa yrityksen kotisivuilta. Käytössä ollut versio 5 oli uusin työn tekohetkellä. Ohjelmankehitysympäristö sisältää kaiken tarvittavan ohjelman tekemiseen Atmelin mikrokontrollereille. Siihen kuuluvat koodieditori, koodin kääntäjä, koodin lataus kontrollerille ja debuggeri.



KUVA 26. Ohjelmankehitysympäristön käyttöliittymä.

Lisäksi siihen kuuluu paljon hyödyllisiä ja työtä helpottavia ominaisuuksia kuten esimerkiksi ennustava koodinkirjoittaminen: kun alkaa kirjoittaa, ohjelma antaa ehdotuksia, joista sitten voi valita haluamansa.

6.2 Alustukset

Jokaisen ohjelman alussa suoritetaan vain kerran tehtävät alustukset. Näistä ensimmäisenä koodiin sisällytetään #include-komennolla tarvittavat otsikkotiedostot (kuva 27). Tärkein on io.h, joka kertoo, kuinka mitäkin komentoriviä käsitellään, eli se on kuin kontrollerin käyttöohje. Tiedostossa tehdään kaikille kontrollereille yhteiset määrittelyt, kuten esimerkiksi pino-osoitin ja tilarekisteri. Mallikohtaiset määrittelyt kertovat esimerkiksi, missä osoitteessa kontrollerin I/O-portit sijaitsevat. Interrupt.h-tiedosto taas ohjeistaa, kuinka käsitellä keskeytyksiin liittyviä komentorivejä, ja se on pakollinen käytettäessä kontrollerin keskeytyspalveluita.

```
//Tarvittavien tiedostojen liittäminen projektiin
#include <avr/io.h>
#include <avr/interrupt.h>
#include "variables.h"
#include "defines.h"
```

KUVA 27. Tarvittavien otsikkotiedostojen liittäminen projektiin.

Jälkimmäiset kaksi tiedostoa luotiin projektin edetessä. Variables.h-tiedostossa on esitelty ja alustettu työssä käytetyt muuttujat sekä taulukot. Defines.h-tiedostossa on määritelty työssä käytetyille I/O-liitännöille helppokäyttöisemmät nimet (kuva 28). Koodin kääntövaiheessa tämä tarkoittaa vain, että #define-komennon vasemmanpuoleinen nimi korvataan oikeanpuoleisella nimellä tai koodilla.

```
//I/O-liitännöille selkeämmät nimet
#define SOUT_H  PORTD=PORTD|1<<PD2
#define SOUT_L  PORTD=PORTD&~(1<<PD2)
#define COLUMN  PORTA
```

KUVA 28. Annetaan I/O-liitännöille selkeämmät nimet.

Kuten PORTA:n tapauksessa, koko portin eli kahdeksan lähdön ryhmän tilaa pystytään muuttamaan, mutta tässä työssä oli myös tarve muutella yksittäisten lähtöjen tilaa ja sitä C-kieli ei suoraan tue. Tällöin se saadaan hoidettua maskaamalla. Sillä tarkoitetaan loogisen operaation tekemistä portin sen hetkisten tilojen ja itse muodostettavan maskin välillä. AND- eli JA-operaatiolla saadaan yksittäinen bitti asetettua alatilaan ja OR- eli TAI-operaatiolla päinvastoin. AND-operaation tulokseksi saadaan ’1’ ainoastaan, jos kummankin operoitavan luvun arvo on ’1’. OR-operaation tulos on ’1’, jos toisen tai molemman operoitavan luvun arvo on ’1’. NOT- eli EI-operaation tulokseksi saadaan aina päinvastainen luku.

Kuvassa 29 selvennettiin, kuinka PORTC:n 5-bitti saatiin asetettua ylä- ja alatilaan. OR-operaattorina C-kielessä toimii |-merkki. Sen oikealla puolella muodostetaan käytettävä maski, jossa vain PC5-bitti asetetaan ylätilaan. Sen jälkeen suoritetaan OR-operaatio ja sijoitetaan tulos PORTC:en. Tällöin huomataan, että vain PC5-bitti muuttaa arvoaan. Haluttaessa jokin liitäntä alatilaan

maski muodostetaan kuten edellä, mutta se käännetään NOT-operaatiolla (~-merkki) päinvastaiseksi, minkä jälkeen suoritetaan AND-operaatio.

```
#define SCLK_H  PORTC=PORTC|1<<PC5

        01000010    PORTC
OR      00100000    maski
----------------
        01100010    tulos

#define SCLK_L  PORTC=PORTC&~(1<<PC5)

        01110110    PORTC
AND     11011111    maski
----------------
        01010110    tulos
```

KUVA 29. Esimerkit vain yhden I/O-liitännän tilan muuttamisesta maskaamalla.

Seuraavaksi alustettiin I/O-liitännät laitteiston vaatimalla tavalla. Se tehdään ports_init-aliohjelmassa (kuva 30). PORTA:n liitännöillä ohjataan sarakkeita puskuripiirin välityksellä, joten ne alustettiin lähdöiksi. Tämä tehtiin kohdan 3.2. perusteella asettamalla DDRA-tiedonsuuntarekisterin bitit tilaan ’1’. Heksalukuna se tarkoittaa FFh. PORTA-rekisterin bitit asetettiin tilaan ’0’, jolloin liitännät ovat alatilassa. PORTC:n liitännöillä ohjataan siirtorekisteripiiriä, ja nekin alustettiin lähdöiksi samalla periaatteella. Laitteen painonapit sekä optohaarukka on kytketty PORTD:n liitäntöihin. Liitännät, joihin painonapit sekä optohaarukan lähtöpuoli on kytketty, alustettiin tuloiksi asettamalla DDRD-rekisterin kolme vähiten merkitsevää bittiä '0'-tilaan. Loput liitännöistä alustettiin lähdöiksi. Tulojen ylösvetovastukset alustettiin aktiivisiksi asettamalla PORTD:n vastaavat bitit '1'-tilaan.

```
void ports_init(void)
{
    //Porttien alustukset
    PORTA = 0x00;
    DDRA = 0xFF;

    PORTC = 0x00;
    DDRC = 0xFF;

    PORTD = 0xFF;
    DDRD = 0xF4;
}
```

KUVA 30. Porttien I/O-liitäntöjen alustaminen.

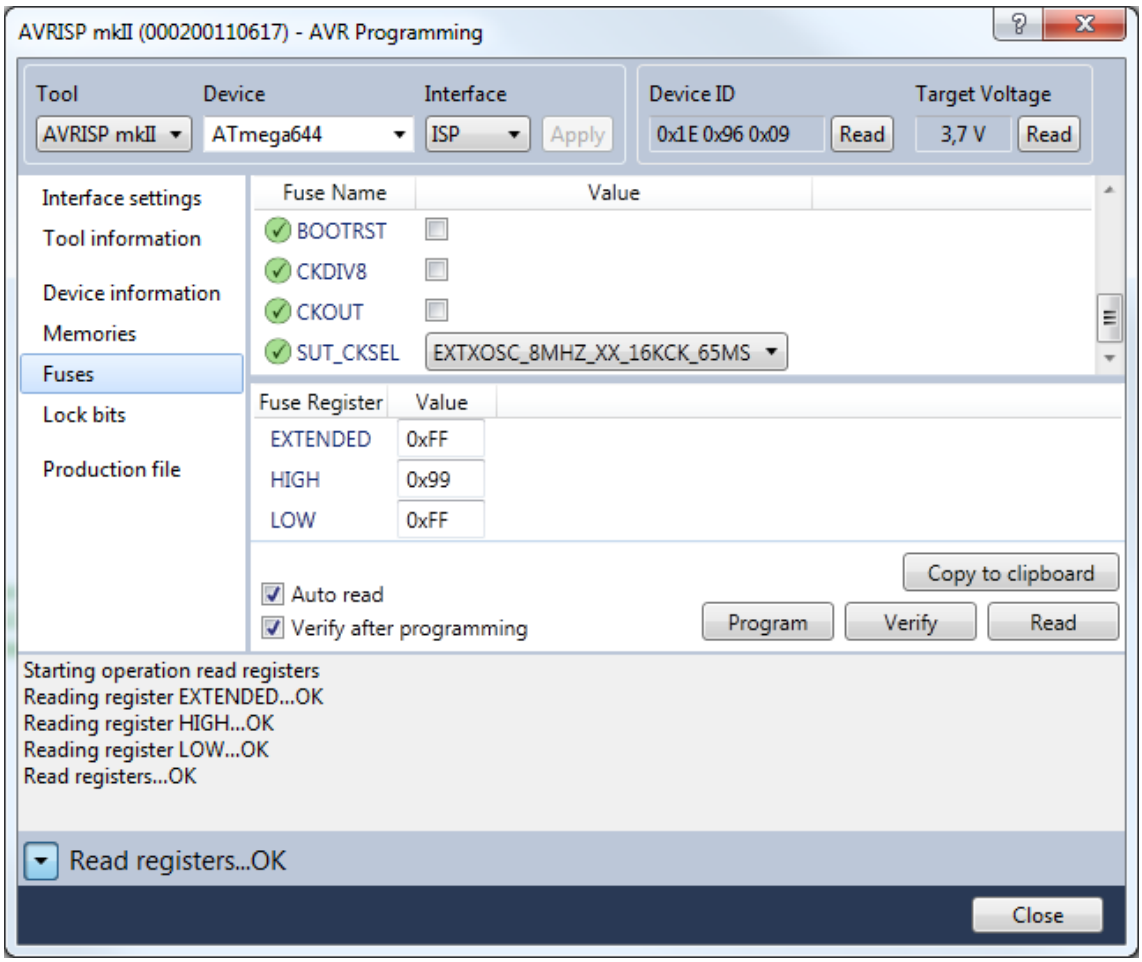
Ports_init-aliohjelmaa kutsuttiin heti pääohjelman alussa. Seuraavaksi asetettiin optohaarukalta tuleva pulssi aiheuttamaan keskeytys nousevalla reunalla. Tämä tehtiin asettamalla EICRA-rekisterin ISC11 ja ISC10 -bitit ylätilaan. Optohaarukan sallittiin aiheuttaa keskeytys asettamalla EIMSK-rekisterin INT1-bitti ylätilaan. Sei()-komennolla sallittiin kaikki keskeytykset. Varsinainen ohjelma käynnistettiin asettamalla optohaarukan ledille menevä liitäntä ylätilaan, jolloin se on aktiivisena ja aiheuttaa keskeytyksen heti, kun jokin kappale menee optohaarukan väliin. Kuvassa 31 näkyy nämä ennen pääohjelman silmukkaa tehtävät alustukset.

```
//Pääohjelma
int main(void)
{
    ports_init();
    EICRA = EICRA | 1<<ISC11;       //asetetaan optohaarukalta tuleva pulssi
    EICRA = EICRA | 1<<ISC10;       //aiheuttamaan keskeytys nousevalla reunalla
    EIMSK = EIMSK | 1<<INT1;        //sallitaan optohaarukan aiheuttaa keskeytys
    sei();                          //sallitaan kaikki keskeytykset

    SOUT_H;                         //optohaarukka päälle

    while(1)
    {
```

KUVA 31. Ennen pääohjelman silmukkaa tehtävät alustustoimet.

Mikrokontrolleri haluttiin toimimaan 20 MHz:n nopeudella, joten työssä täytyi ottaa huomioon myös niin sanotut fuse-bitit. Niitä ei määritellä koodissa vaan ohjelmankehitysympäristön asetuksista, minkä jälkeen ne ohjelmoidaan kontrolleriin erikseen. Taulukoiden 1, 2 ja 3 perusteella CKSEL3..0-bittien arvoiksi ase-

tettiin 0111 ja SUT1..0-bittien arvoiksi 11. Lisäksi CKDIV8-bitti piti asettaa ei-aktiiviseksi. Tällöin kontrolleriin asetettiin hidas nousuaika ja kellon taajuus riippuu ulkoisesta kiteestä, joka tässä tapauksessa on 20 MHz. AVR Studio 5 -ohjelmankehitysympäristössä fuse-bitteihin pääsee käsiksi valitsemalla Tools - AVR Programming, jolloin avautuu kuvan 32 mukainen ikkuna. Asetukset on tehty helpoiksi ja niitä voidaankin muuttaa laittamalla rastin ruutuun tai päinvastoin.



KUVA 32. Fuse-bittien asettaminen AVR Studio 5 -ohjelmassa.

6.3 Laskurien toiminta

Kolmiulotteisen kuvan muodostamiseen käytetään kahta mikrokontrollerin sisältämää laskuria, 16-bittistä sekä 8-bittistä. Selvitetään seuraavaksi, miten ne on tässä työssä ohjelmoitu toimimaan.

Kuvassa 33 näkyy Laskuri 1:n 16-bittinen TCNT1-rekisteri. Se on jaettu kahteen osaan, koska mikrokontrollerin väyläleveys on vain 8-bittiä eikä 16-bittistä tietoa voida käsitellä kerralla. Laskurin alkuarvo tallennetaan tähän rekisteriin ja sen lukema jollakin ajanhetkellä on luettavissa täältä. Laskuri 1:n alustamisessa rekisteriin asetetaan heksaluku 0000h eli laskenta aloitetaan nollasta ylöspäin. Suurin mahdollinen laskettava luku on 65 535 eli heksalukuna FFFFh.

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 33. 16-bittinen TCNT1-rekisteri.

Laskuri käynnistetään asettamalla TCCR1B-rekisterin (kuva 34) Clock Select Bitit eli esijakajan bitit halutulla lailla. Eri vaihtoehdot laskentapulssien lähteeksi löytyvät taulukosta 6. Tässä tapauksessa CS10 ja CS11 bittien arvoiksi asetetaan '1', jolloin laskuri laskee joka 64:nnen kellopulssin.

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 34. TCCR1B-rekisteri.

Toisena laskurina työssä käytetään 8-bittistä Laskuri 0:aa. Sen laskuväli on 0–255 eli heksalukuna 00–FFh. Laskurin alkuarvo alustetaan TCNT0-rekisteriin (kuva 35) niin, että laskurin maksimiluvusta vähennetään Laskuri 1:ltä saatava luku, ja siitä aloitetaan laskemaan ylöspäin. Laskuri 1:ltä saatava luku selvitetään myöhemmin kohdassa 6.5.

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 35. 8-bittinen TCNT0-rekisteri.

Seuraavaksi laskurin TIMSK-rekisteristä (kuva 36) asetetaan TOIE0-bitti arvoon '1', jolloin sallitaan ylivuotokeskeytys. Ylivuodolla tarkoitetaan sitä, kun laskuri laskee maksimiluvun yli. Tällöin suoritetaan keskeytysohjelma. Laskurin käynnistys tapahtuu samoin kuten Laskuri 1:nkin tapauksessa eli TCCR0B-rekisterin (kuva 37) CS00 ja CS01 bitit asetetaan '1':ksi.

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 36. Laskuri 0:n TIMSK-rekisteri.

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

KUVA 37. TCCR0B-rekisteri.

6.4 Kuvion muodostus

Fyysisesti ledejä laitteessa on vain 8 x 8-matriisi eli yhteensä 64 lediä. Jaettaessa yksi kierros 32 lohkoon ledejä muodostuu jo 2048 kappaletta, sanotaan niitä nyt vaikka virtuaaliledeiksi. Yhden kolmiulotteisen kuvan jokaisen yksittäisen virtuaaliledin tila on tallennettu 32 x 64 bitin kokoiseen taulukkoon (kuva 38). Liikkuvaa kuvaa muodostettaessa näitä taulukoita tarvitaan paljon enemmän.

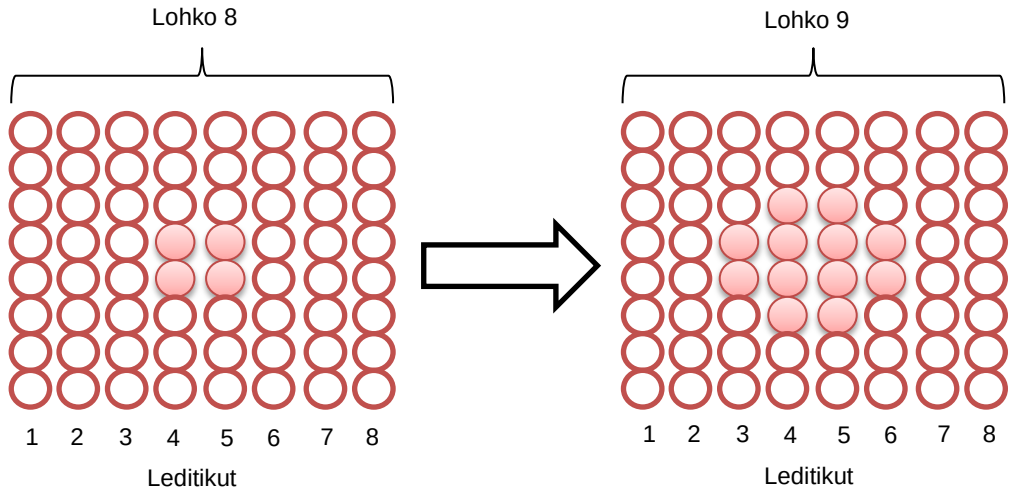
```
//Kuvataulukoiden esittely ja alustus
unsigned char figure0[32][8] = {{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //1. lohko
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //2. lohko
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //.
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //.
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //.
                               {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00},       //8.
                               {0x00, 0x00, 0x18, 0x3C, 0x3C, 0x18, 0x00, 0x00},
                               {0x00, 0x00, 0x18, 0x3C, 0x3C, 0x18, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x18, 0x18, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //12.
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //16.
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //20.
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},       //24.
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
                               {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}};
```

KUVA 38. Yhden kolmiulotteisen kuvan ledien tilat.

Taulukko on muodostettu heksaluvuista, joista jokainen sisältää aina yhden leditikun ledien tilat. Yksi taulukon rivi siis sisältää yhden taulukon ledien tilat. Esimerkiksi heksaluku 0Fh (taulukossa 0x0F) tarkoittaa binäärilukuna 00001111b. ’0’ tarkoittaa, että ledi pysyy pimeänä, ja ’1’ päinvastoin.

Kuvion muodostus aloitetaan ensimmäisen lohkon ensimmäisestä leditikusta. Taulukosta haetaan vastaava heksaluku, joka sijoitetaan COLUMN-muuttujaan. Tällöin se kulkeutuu PORTA:sta puskuripiirin välityksellä ledien katodeille eli sarakepuolelle. Siirtorekisteripiirin tuloon lähetetään ’1’ ja sen kellotuloihin yhdet pulssit, jolloin ensimmäinen leditikku tulee aktiiviseksi. Yhden leditikun piirtämiseen lasketun ajan jälkeen taulukosta haetaan seuraava luku, siirtorekisteripiirin tuloon lähetetään ’0’ ja kellotuloihin pulssit. Tällöin aktiivisena on toinen leditikku. Tästä eteenpäin vain vaihdetaan seuraava luku ja lähetetään pulssit siirtorekisteripiirin kellotuloihin. Kun viimeisen leditikun kohdalla tulee aika vaihtaa seuraavaan, vaihdetaan seuraava lohko (eli seuraava rivi taulukosta) ja aloitetaan leditikut alusta. Viimeisen lohkon viimeisen leditikun jälkeen aloitetaan ko-

ko kuvion muodostus alusta. Noin joka kolmannen kierroksen jälkeen vaihdetaan piirrettävää kuviota, millä saadaan muodostettua liikkuvaa kuvaa. Kuvassa 39 on selvennetty leditikkujen skannaus ja lohkon vaihtotilanne.



KUVA 39. Aktiivisen leditikun skannaaminen ja lohkon vaihto

6.5 Kuvion tahdistus

Ihmisen silmän ominaisuuksien vuoksi kuvion tahdistustaajuudeksi valittiin 25 Hz eli kuvio piirretään 25 kertaa sekunnissa. Tämä tarkoittaa, että yhden kuvion piirtämiseen kuluu aikaa 1 s / 25 = 0,04 s = 40 ms. Kun se jaetaan 32:lla, saadaan yhden lohkon piirtämiseen käytettävä aika, eli 40 ms / 32 = 1,25 ms. Jaetaan se vielä 8:lla, niin saadaan yhden sarakkeen piirtämiseen käytettävä aika, 1,25 ms / 8 = 0,15625 ms = 156,25 µs.

Kuvion tahdistamiseen käytetään laitteen optohaarukkaa sekä kahta mikrokontrollerin sisältämää laskuria. Jokaisella kierroksella optohaarukka antaa pulssin mikrokontrollerin tuloon täysin samalla kohdalla kierrosta, kun jokin kappale menee haarukan lähettimen ja vastaanottimen välistä. Pulssi on ohjelmoitu aiheuttamaan keskeytys, joka aloittaa kuvion piirtämisen alusta. Keskeytysohjelmassa otetaan ylös 16-bittisen laskurin lukema ja alustetaan se uudelleen. Lukema on siis laitteen kierrosaika kellopulsseina. Se jaetaan 256:lla, jotta saadaan yhden sarakkeen piirtämiseen käytettävä aika.

Optohaarukan käynnistämän keskeytysohjelman aikana alustetaan myös 8-bittinen laskuri, joka laskee kellopulsseja edellä mainitun yhden sarakkeen piirtämiseen kuluvan ajan. Tällöin laskuri aiheuttaa keskeytyksen. Keskeytysohjelmassa laskuri alustetaan uudestaan, minkä jälkeen päivitetään näyttöä. Aktiivinen sarake vaihdetaan seuraavaan ja mahdollisesti myös piirrettävänä oleva lohko sekä koko kuvio.

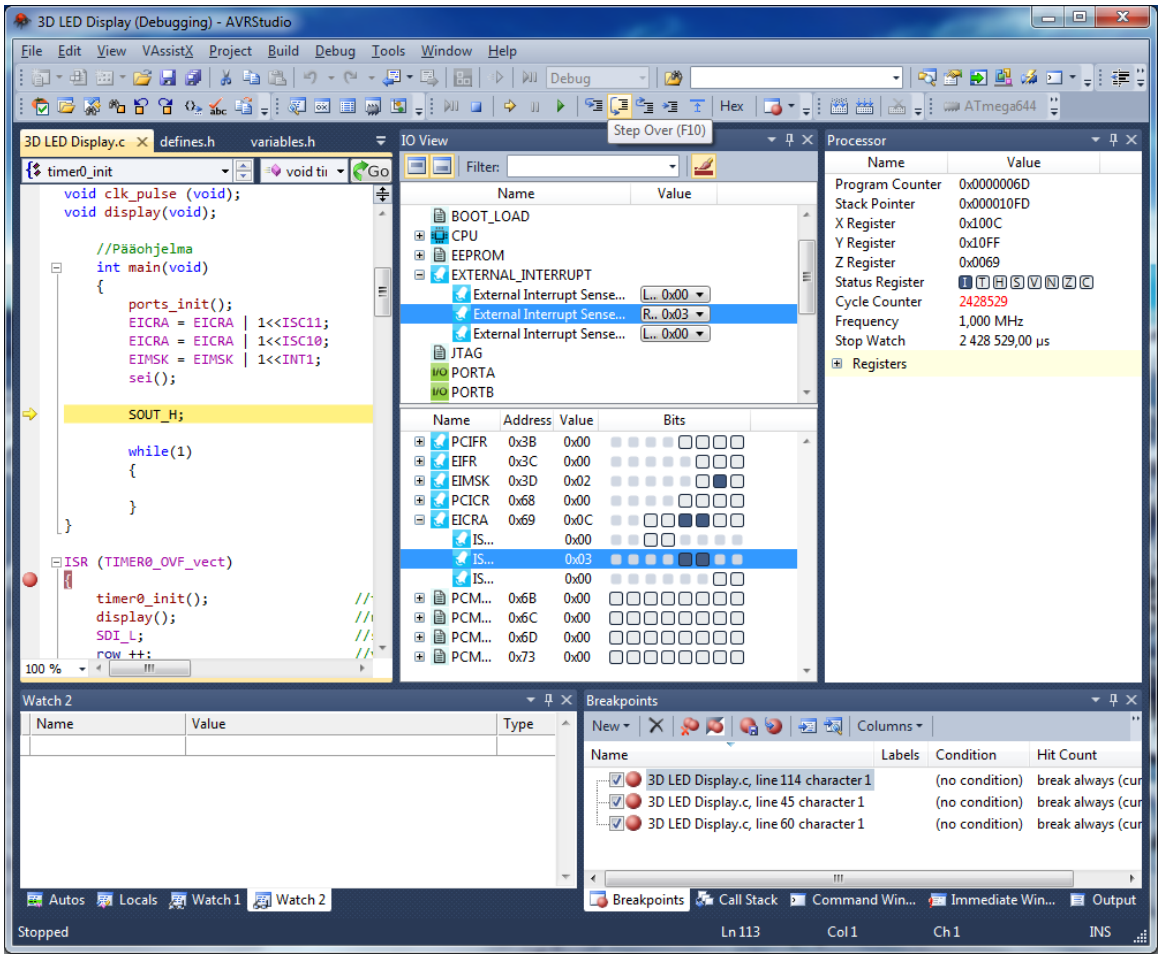
7 TESTAUS

7.1 HW-testaus

Piirilevyjen valmistuttua ensimmäiseksi mitattiin yleismittarilla mahdolliset oikosulut lähekkäin olevissa vierekkäisissä johdinvedoissa. Osa leditikkujen piirilevyistä jouduttiin hylkäämään ja valmistamaan uusia. Syynä olivat piirilevyn kerrosten väliin jääneet kuparisäikeet, jotka aiheuttivat oikosulkuja johdinvetojen välille. Prototyypin kasauksen jälkeen akku liitettiin paikalleen ja kytkettiin virrat päälle. Kaikilta mikropiireiltä mitattiin käyttöjännitteet. Liian alhainen käyttöjännite voisi aiheuttaa ongelmia laitteen toimintaan. Lopuksi vielä testattiin ledien ja niille vievien johdinvetojen toimivuus. Tämä tehtiin yleismittarin dioditesterillä ohjainpiirilevyn puolelta. Tällöin tuli testattua kaikki liittimet ja johdinvedot ohjainpiirilevyltä sovitinpiirilevyn kautta leditikkujen ledeille asti.

7.2 SW-testaus

Ohjelmakoodi kirjoitettiin osio kerrallaan. Ne oli tehtävä tietyssä järjestyksessä, koska osa osioista oli riippuvaisia toisien osioiden toiminnasta. Ohjelmakoodin testaukset suoritettiin käyttämällä ohjelmankehitysympäristön debuggeria. Siinä ohjelmaa ajetaan niin sanotussa virtuaalikontrollerissa. Ohjelmakoodin testaus suoritettiin ajamalla debuggerilla komento kerrallaan ja seuraamalla kontrollerin rekisterien tiloja. Tällöin nähtiin heti, jos rekisterin tila ei muutu, kuten pitäisi. Kuvassa 40 näkyy ohjelmankehitysympäristö debuggaustilassa.



KUVA 40. Ohjelmankehitysympäristön näkymä debuggaustilassa.

Ensimmäisenä oli vuorossa ledimatriisin toimintaan vaikuttavat osiot eli siirtorekisteri- ja puskuripiirin ohjaukset. Seuraavaksi testattiin laskurien ja optohaarukan toiminta.

7.3 Integrointitestaus

Integrointitestauksessa yhdistettiin laitteisto ja ohjelmisto toisiinsa. Tämän jälkeen suoritettiin testi niiden yhteensopivuuden määrittämiseksi. Tässä työssä integrointitestaukset suoritettiin jokaisen uuden ohjelmaosion jälkeen. Tällä vältettiin virheiden kasautuminen, kun kukin osio testattiin toimivaksi ennen seuraavaan siirtymistä.

Käytännössä ensimmäiseksi oli saatava ledimatriisi toimimaan, minkä avulla sitten voitiin saada näkyviä merkkejä muidenkin osioiden toiminnasta. Ensimmäinen ongelma havaittiin, kun kaikki leditikut olivat yhtä aikaa aktiivisina. On-

gelma löytyi siirtorekisteripiirin shift clock –pinnistä, joka ei jaksanut vetää itseään alatilaan pulssien välillä. Ongelma ratkaistiin laittamalla vastus shift clock- ja latch clock -pinnien välille. Sen avulla jännitettä saatiin laskettua hieman alemmaksi.

Opinnäytetyön tekemisen aikana prototyyppiin sopivaa moottoria ei löydetty, minkä vuoksi käytännön testit tehtiin käyttämällä riittävän tehokasta porakonetta. Sen poraussyvyyden rajoitin asetettiin kulkemaan optohaarukan välistä, jolloin piirrettävä kuva saatiin tahdistettua. Ledien huonon valotehon vuoksi testit jouduttiin tekemään hämärässä, jotta kuva nähtiin selkeästi.

8 JATKOKEHITYS

Tässä työssä käytettyyn mikrokontrolleriin olisi helposti mahdollista lisätä jopa nelinkertaisesti ledejä eli 16 x 16-matriisi. Sen nopeus kyllä riittäisi niitä ohjaamaan, mutta tällöin ongelmaksi muodostuisi hyvin äkkiä muistin riittämättömyys. Kuitenkin jos näyttöä ohjattaisiin esimerkiksi suoraan tietokoneelta langattomasti, se voisi olla toimiva ratkaisu. Toisaalta myös ulkoista lisämuistia on mahdollista liittää mikrokontrolleriin. Se onnistuu helposti SPI-liitännän kautta, jolloin se vie vain neljä I/O-liitäntää mikrokontrollerilta. Tällöin taulukoiden alustamiseen muistiin tarvitaan luultavasti erillinen ohjelmointilaite. Yksi huomion arvoinen seikka on myös, että mitä useampia sarakkeita joudutaan skannaamaan, sitä heikommalta ledien valo näyttää. Ledien valoteho pitäisi siis olla riittävän hyvä pystyäkseen erottumaan jopa kirkkaassa päivänvalossa.

Mitä sitten vaaditaan laitteistolta, että pystytään esittämään selkeästi esimerkiksi jokin eläin liikkeessä? Oletetaan ledit niin tehokkaiksi, että niiden valoteho riittää, vaikka ne palaisivatkin vain todella lyhyitä aikoja aina tietyssä kohtaa. Valitaan ledimatriisin kooksi 32 x 32 lediä ja jaetaan kierros 64 lohkoon. Tällöin fyysisiä ledejä laitteessa olisi 1024 ja niin sanottuja virtuaaliledejä huimat 65 536 kappaletta. Piirrettäessä kuva 25 kertaa sekunnissa yhden sarakkeen paloaika kerrallaan olisi vain noin 19,5 mikrosekunnin luokkaa.

Valitaan kontrolleriksi 32-bittinen vaihtoehto, jolloin saadaan lisää suorituskykyä suuremman kellotaajuuden sekä useampien I/O-liitäntöjen ansiosta. Esimerkiksi Atmelin 32-bittisten kontrollereiden jopa 66 MHz:n kellotaajuudella todella nopea sarakkeiden skannaamisen ei pitäisi olla ongelma.

Huomioon on otettava myös virrankulutus. Kerrallaan palaa maksimissaan 32 lediä. Laitteessa on käytettävä hyvän valotehon omaavia ledejä, jolloin yhden ledin virrankulutus voi olla esimerkiksi 40 mA. Tällöin ledien virrankulutus yhdellä ajanhetkellä olisi maksimissaan 1,28 A. Kuitenkin lyhyet paloajat himmentävät ledejä ja ongelman parantamiseksi niille saatetaan syöttää jopa kaksinkertainen määrä virtaa, jolloin virrankulutus voisi olla jopa 2,56 A.

9 YHTEENVETO

Opinnäytetyön tarkoituksena oli tutkia mahdollisuutta rakentaa kolmiulotteinen ledinäyttö ja sen tuottamaa näkymää ihmissilmään. Tavoitteeksi asetettiin toimivan prototyyppi, joka pystyy näyttämään jonkinlaista animaatiota näytöllä.

Työ jaettiin kahteen suurempaan osaan, prototyypin rakentamiseen ja ohjelmiston kirjoittamiseen. Kumpaankin osaan sisältyi tietysti myös tiedonhakua eri lähteistä. Prototyypin rakentaminen aloitettiin sopivien kytkentöjen suunnittelemisella. Piirikaaviot piirrettiin Cadence Orcad -ohjelmiston Capture-osalla ja piirilevyt suunniteltiin saman ohjelmiston Layout-osalla. Prototyypin valmistuttua alettiin kirjoittaa ohjelmaa mikrokontrollerille, joka ohjaa laitteiston toimintaa. Ohjelmankehitysympäristönä käytettiin ilmaiseksi saatavaa Atmelin AVR-studiota.

Lopputulokseksi saatiin tavoitteen täyttävä kolmiulotteista liikkuvaa kuvaa näyttävä prototyyppi. Animaatiossa on kaksi mailaa, jotka vuorotellen lyövät palloa toisilleen. Pallon mennessä ensimmäiseltä mailalta toiselle pallo kirjoittaa ilmaan ”3D” ja tullessaan takaisin pyyhkii sen. Animaatio on toteutettu vain 15:llä eri kuvalla, koska kontrollerin muisti ei riittänyt enempään. Sen vuoksi animaatio nykii hieman. Pyörimisnopeuden vaihteluista huolimatta kuva pysyy samalla kohdalla eikä lähde pyörimään.

Pahin ongelma prototyypissä oli ledien valoteho, joka ei oikein riittänyt päivänvalossa. Testit jouduttiinkin tekemään mieluiten hämärässä tai jopa pimeässä parhaan näkymän saamiseksi. Huonon valotehon osasyyksi epäilen myös, ettei noin 4 voltin käyttöjännite akulta ollut riittävä laitteelle. Tällöin käyttöjännitteen nosto voisi tuoda parannusta tähän ongelmaan.

Itse olen todella tyytyväinen lopputulokseen. Totta kai kehitettävää on paljonkin esimerkiksi kaupallista tuotetta ajatellen, mutta näin ensimmäiseksi prototyypiksi laite oli onnistunut ja se täytti tavoitteet. Yksi haastava asia työssä oli hahmottaa laitteen piirtämä kolmiulotteinen kuva, varsinkin kun jokaisen yksittäisen ledin tila piti syöttää manuaalisesti taulukkoon. AVR-mikrokontrollereita olen hieman tutkinut aiemminkin, mutta tämä työ syvensi tietämystäni paljon ja myös lisäsi kiinnostustani niitä kohtaan yhä enemmän.

LÄHTEET

1. Frame rate. 2012. Saatavissa: http://en.wikipedia.org/wiki/Frame_rate. Hakupäivä 3.7.2012.

2. Brand, Dustin D. 2001. Human eye frames per second. Saatavissa: http://amo.net/NT/02-21-01FPS.html. Hakupäivä 3.7.2012.

3. Koskinen, Jari 2004. Mikrotietokonetekniikka; Sulautetut järjestelmät. Keuruu: Otava.

4. Atmel AVR. 2012. Saatavissa: http://www.atmel.com/products/microcontrollers/avr/default.aspx. Hakupäivä 12.3.2012.

5. ATmega644. 2012. Saatavissa: http://www.atmel.com/Images/doc2593.pdf. Hakupäivä 12.3.2012.

6. 74HC595. 2000. Saatavissa: http://pdf1.alldatasheet.com/datasheet-pdf/view/15644/PHILIPS/74HC595.html. Hakupäivä 13.3.2012.

7. ULN2803. 1996. Saatavissa: http://pdf1.alldatasheet.com/datasheet-pdf/view/12687/ONSEMI/ULN2803.html. Hakupäivä 15.3.2012.

Features of ATmega644/V

• High-performance, Low-power Atmel® AVR® 8-bit Microcontroller

• Advanced RISC Architecture

– 131 Powerful Instructions – Most Single-clock Cycle Execution

– 32 × 8 General Purpose Working Registers

– Fully Static Operation

– Up to 20 MIPS Throughput at 20MHz

• High Endurance Non-volatile Memory segments

– 64 Kbytes of In-System Self-programmable Flash program memory

– 2 Kbytes EEPROM

– 4 Kbytes Internal SRAM

– Write/Erase cyles: 10,000 Flash/100,000 EEPROM(1)(3)

– Data retention: 20 years at 85°C/100 years at 25°C(2)(3)

– Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

– Programming Lock for Software Security

• JTAG (IEEE std. 1149.1 Compliant) Interface

– Boundary-scan Capabilities According to the JTAG Standard

– Extensive On-chip Debug Support

– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

• Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture

Mode

– Real Time Counter with Separate Oscillator

– Six PWM Channels

– 8-channel, 10-bit ADC

Differential mode with selectable gain at 1x, 10x or 200x

– Byte-oriented Two-wire Serial Interface

– One Programmable Serial USART

– Master/Slave SPI Serial Interface

– Programmable Watchdog Timer with Separate On-chip Oscillator

– On-chip Analog Comparator

– Interrupt and Wake-up on Pin Change

• Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated RC Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby

and Extended Standby

• I/O and Packages

– 32 Programmable I/O Lines

– 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

• Speed Grades

– ATmega644V: 0 - 4MHz @ 1.8V - 5.5V, 0 - 10MHz @ 2.7V - 5.5V

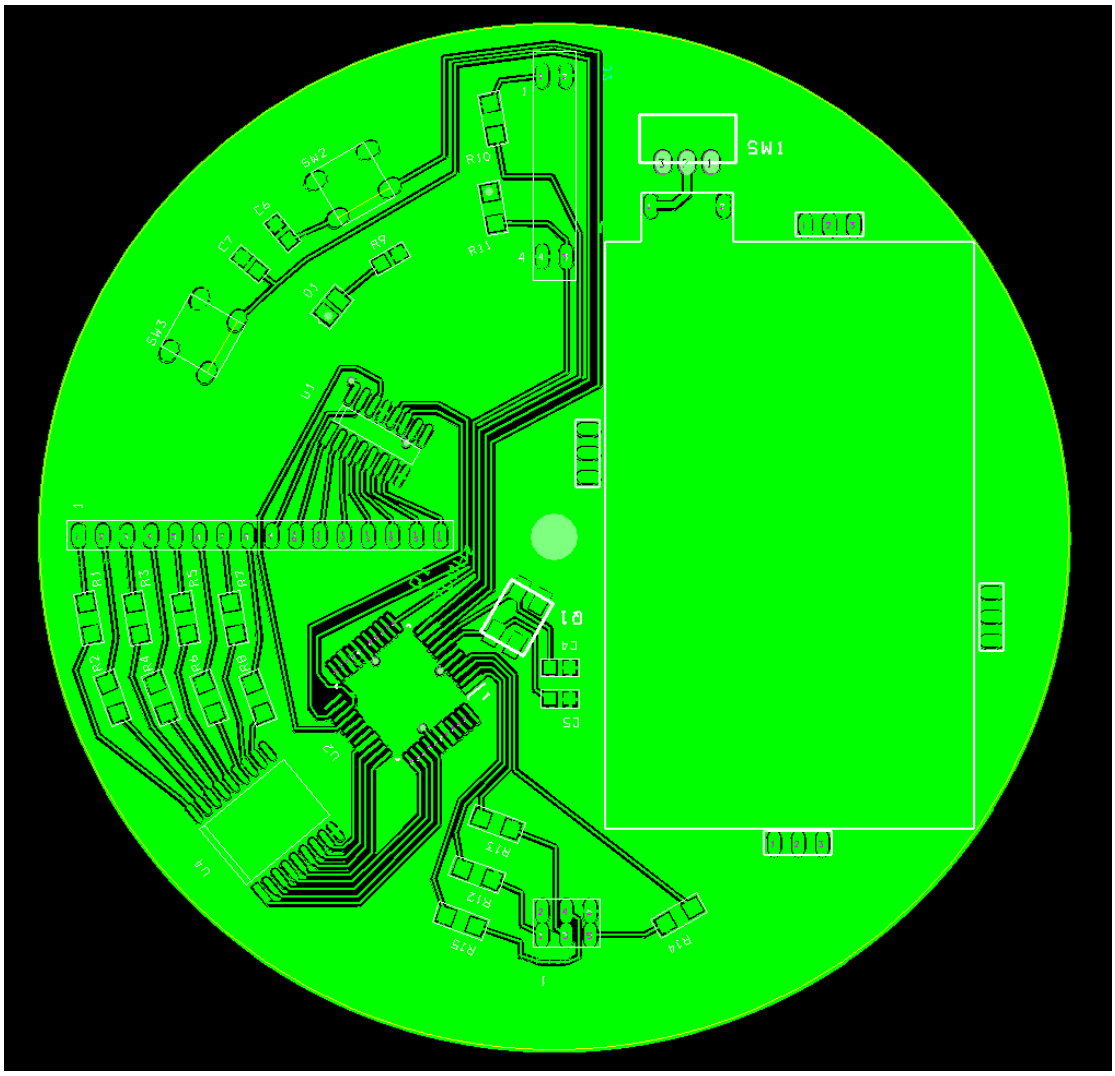
– ATmega644: 0 - 10MHz @ 2.7V - 5.5V, 0 - 20MHz @ 4.5V - 5.5V

• Power Consumption at 1MHz, 3V, 25·C

– Active: 240µA @ 1.8V, 1MHz

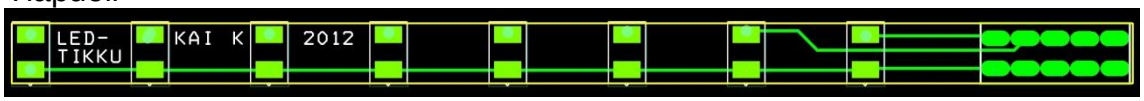
– Power-down Mode: 0.1µA @ 1.8V (5.)

Piirilevy ohjauselektroniikalle

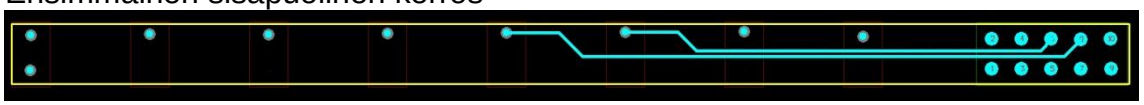


Leditikun piirilevy

Yläpuoli

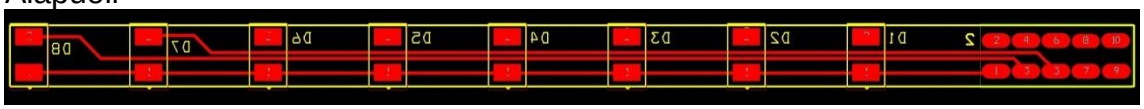


Ensimmäinen sisäpuolinen kerros



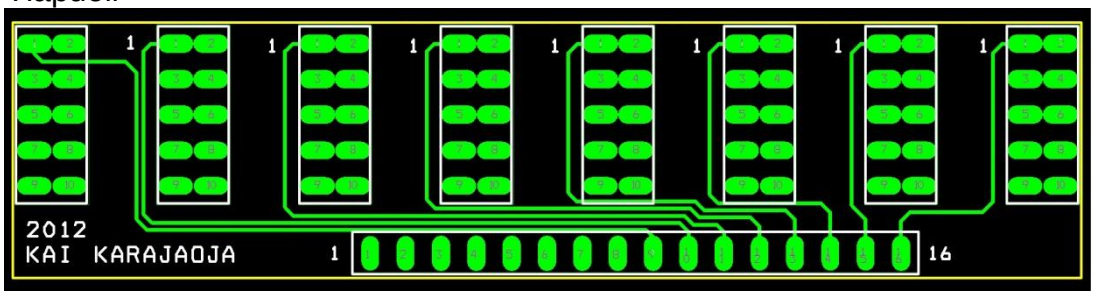
Toinen sisäpuolinen kerros

Alapuoli

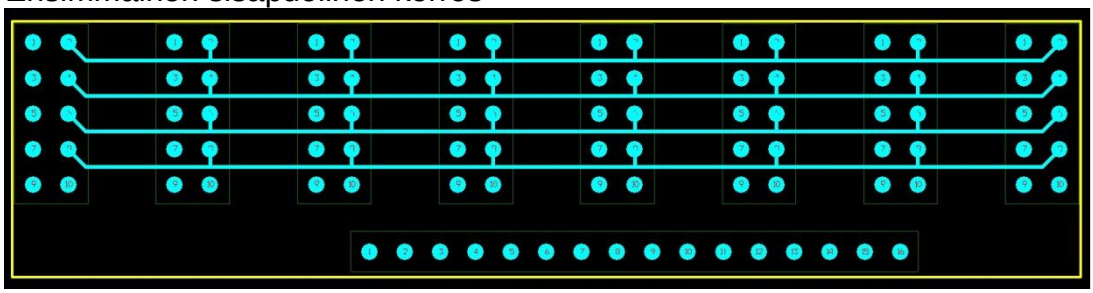


Sovitinpiirilevy

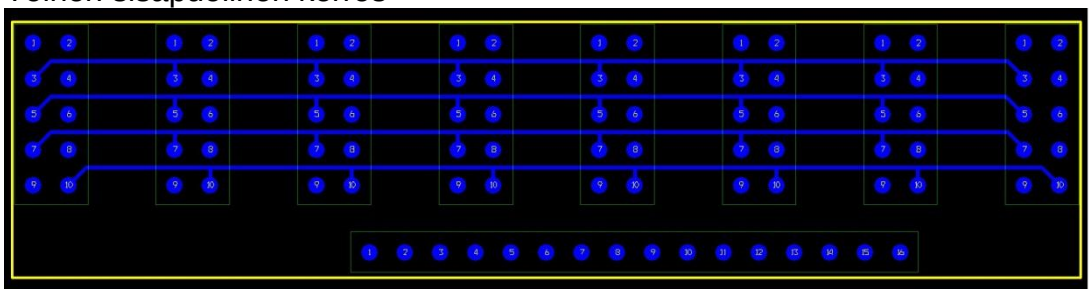
Yläpuoli



Ensimmäinen sisäpuolinen kerros



Toinen sisäpuolinen kerros



Alapuoli