



**LAHDEN AMMATTIKORKEAKOULU**  
*Lahti University of Applied Sciences*

# AUTOMATISOITU JÄÄHDYTYSJÄRJESTELMÄ

LAHDEN  
AMMATTIKORKEAKOULU  
Tekniikan ala  
Tietotekniikan koulutusohjelma  
Tietokone-elektronikka  
Opinnäytetyö  
Kevät 2013  
Juha Haavistola



Lahden ammattikorkeakoulu  
Tietotekniikan koulutusohjelma

HAAVISTOLA, JUHA:

Automatisoitu jäähdytysjärjestelmä

Tietokone-elektroniikan opinnäytetyö, 33 sivua, 3 liitesivua

Kevät 2013

TIIVISTELMÄ

---

Tämän opinnäytetyön tarkoituksena oli suunnitella automaattinen jäähdytysjärjestelmä, joka sopii tietokoneisiin sekä myös muihin järjestelmiin. Työssä perehdytään pulssinleveysmodulaatioon ja lämpöoppiin, koska nämä ovat tärkeitä ominaisuuksia laitteen toiminnallisuuden takia. Jäähdytysjärjestelmä suunnitellaan loppuun saakka ja tehdään prototyyppi, mutta valmista laitetta ei rakenneta, koska tämä ei ole oleellista työn kannalta.

Jäähdytysjärjestelmä koostuu mikrokontrollerista, LCD-näytöstä, lämpöantureista ja MOSFET:sta. Mikrokontrolleri on ohjelmoitu C++-kielellä.

Työn tarkoituksena on tutkia, mitä kaikkea automaattinen tuuletusjärjestelmä vaatii toimiakseen. Tarkoituksena on myös verrata itse tehdyn laitteen ominaisuuksia kaupasta ostetun laitteen ominaisuuksiin ja siihen, mitä kaikkea itse rakennettuun laitteeseen pystytään lisäämään enemmän.

Työn tavoitteet saavutettiin; prototyyppi saatiin rakennettua ja järjestelmä toimii kuten on oletettu. Järjestelmän tuulettimien pyöriminen toimii vakaasti ja kierrosnopeus muuttuu lämpötilan mukaan. Järjestelmä myös tallentaa tietyin väliajoin lämpötilan ja kierrosnopeuden mikrokontrollerin muistiin. Optiona on myös tallentaa tieto serverille, josta tiedon voi lukea etänä.

Asiasanat: pulssinleveysmodulaatio, jäähdytysjärjestelmä, lämpöoppi

Lahti University of Applied Sciences  
Degree Programme in Information Technology

HAAVISTOLA, JUHA:

Automated fan controller

Bachelor's Thesis in Computer Electronics, 33 pages, 3 pages of appendices

Spring 2013

ABSTRACT

---

The objective of this thesis was to design an automated fan controller that can be used with computers and other systems. The fan was based on pulse width modulation and thermal physics, so they were examined first. The objective was to design and make only a prototype, not a working device. The reason for this is there is no need to make a fully working device in this project.

The device consists of a microcontroller unit, LCD-displays as indicators, thermal probes and MOSFETs. The program inside the microcontroller is written in C++ language.

The aim was to examine what the automated fan controller would need to work properly. A comparison was also made between a device bought in a shop and self-made device and what more can be added to a self-made device.

The objectives were met; the prototype was built and it works as expected. The fans rpm is constant and if the temperature changes the fans rpm also changes. The device also saves data to the logs inside the microcontroller. It can also be made to save the log file to a specified server.

Key words: fan controller, pulse width modulation, thermal physics

## SISÄLLYS

1	JOHDANTO	1
2	PULSSINLEVEYSMODULAATIO	2
2.1	PWM:n historia	3
2.2	PWM:n toimintaperiaate	5
2.2.1	Delta-modulointi	7
2.2.2	Delta-sigma-modulaatio	8
2.2.3	PWM-tyypit	9
2.3	Tehon siirto	10
3	LÄMPÖ ELEKTRONIIKAN LAITTEISSA	12
3.1	Komponentin kotelo	12
3.2	Lämmönsiirron analogia	13
4	SÄHKÖLAITTEEN JÄÄHDYTYSTARPEEN MÄÄRITTÄMINEN	15
5	JÄÄHDYTYSJÄRJESTELMÄ	19
5.1	Suunnittelu	19
5.2	Testiympäristön rakentaminen	23
5.3	Testiajosta tehtävät päätelmät	25
5.4	Piirilevysuunnittelu	25
5.5	Ohjelmakoodi	27
6	JOHTOPÄÄTÖKSET	28
7	YHTEENVETO	30
	LÄHTEET	34
	LIITTEET	36

# 1 JOHDANTO

Tämä työ käsittelee sitä, kuinka voidaan itse rakentaa PWM-ohjattu tuuletusjärjestelmä. luvuissa käydään läpi teoriaa joka on oleellista laitteen toiminnan kannalta. Teoriaosuudessa käsitellään PWM-ohjausta fysikaalisesta näkökulmasta, selvitetään mitä se on ja miksi sitä on hyvä käyttää tällaisissa järjestelmissä. Toinen asia, joita tutkitaan ennen kuin itse laitetta ruvetaan suunnittelemaan, on lämpöoppi. Tätä on hyvä tutkia sen takia, koska on tiedettävä miten eri komponentit käyttäytyvät eri lämpötiloissa ja mitä tapahtuu, jos lämpötila ylittää suositukset erilaisissa järjestelmissä.

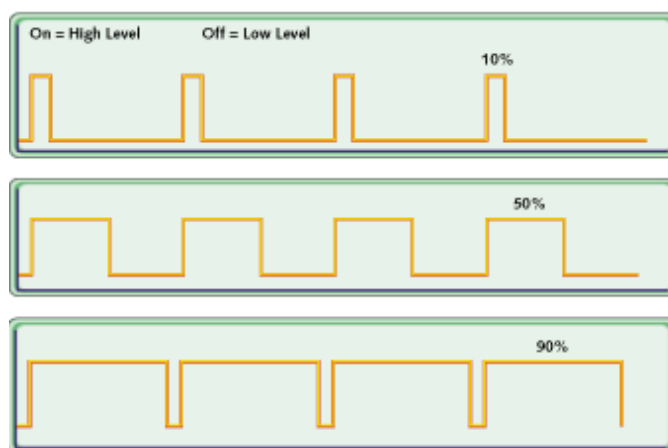
Teorian jälkeen siirrytään itse fyysisen laitteen suunnitteluun ja siihen, mitä kaikkea täytyy ottaa huomioon itse laitteen kannalta. Tähän sisältyy muun muassa piirilevyn suunnittelu, jossa täytyy ottaa huomioon sellaisia asioita kuten elektromagneettinen yhteensopivuus, ja käydään läpi erilaisia suunnittelusääntöjä. Työssä myös tarkastellaan ohjelmakoodia, jolla laite saadaan toimimaan kunnolla, ilman ongelmia.

Tässä työssä tullaan vertaamaan sitä, onko taloudellisesti kannattavaa ostaa kaupasta vastaavanlainen tuote vai rakentaa itse. Vertailussa otetaan huomioon eri komponenttien hinnat, kuten LCD-näyttö, mikrokontrolleri ja transistorit. Huomioon otetaan myös se seikka, mitä kaikkea voidaan lisätä itse tehtyyn laitteeseen verrattuna kaupasta ostettuun laitteeseen.

## 2 PULSSINLEVEYSMODULAATIO

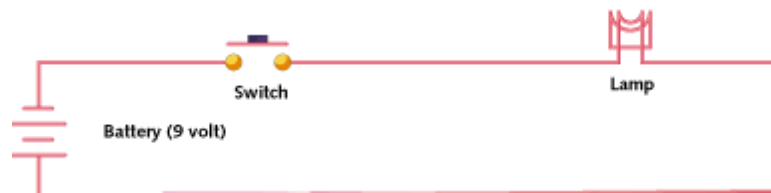
Pulssinleveysmodulaation englanninkielinen termi on pulse width modulation, josta lyhenne PWM tulee. Yksinkertaistettuna PWM-ohjaus on pulssisuhteen muutoksella tehtävää ohjausta, jolloin ohjaus on vuorotellen päällä ja pois tietyllä taajuudella. Jännitteen ja virran keskiarvo, jota syötetään kuormaan, ohjataan kääntämällä lähdettä ja kuormaa päälle ja pois nopeassa tahdissa. Mitä pidemmän aikaa kytkin on päällä, sitä enemmän tehoa syötetään kuormaan. PWM:n yksi suurimmista eduista on tehohäviön pienuus, koska kun kytkin on auki, virtaa ei kulje, ja kun kytkin on kiinni, jännitehäviö kytkimen yli on lähes nolla voltia. Koska teho on jännitteen ja virran tulo ( $P = UI$ ), jompikumpi (jännite tai virta) on aina lähes nolla, joten myös tehohäviö on lähes nolla. (Väyrynen 2013.)

Pulssisuhteella tarkoitetaan sitä, kuinka pitkään signaali on aktiivisessa tilassa (1-tilassa) ja kuinka kauan se on ei-aktiivisessa tilassa (0-tilassa). Tätä pulssisuhdetta kuvataan yleisesti prosenteilla. Englanninkielinen nimitys pulssisuhteelle on duty cycle, tätä nimitystä tullaan käyttämään tässä työssä. Kuviossa 1 on esimerkkinä kuinka PWM:n duty cycle toimii. (Väyrynen 2013.)



KUVIO 1. PWM-signaali eri duty cycle -arvoilla (Barr 2013)

Esimerkkinä PWM-ohjauksesta voidaan pitää seuraavanlaista tilannetta: 9 V:n paristoon on kytketty hehkulamppu, näiden välissä on kytkin. Mikäli kytkin käännetään päälle 50 ms:n ajaksi, saa lamppu 9 V jännitettä. Kun kytkin käännetään pois päältä 50 ms:n ajaksi, lampun jännite nolla voltia. Mikäli tätä toimenpidettä toistetaan 10 kertaa sekunnissa, lamppu syttyy kuin se olisi kytkettynä 4,5 V:n paristoon. Voidaan siis todeta, että tässä tapauksessa duty cycle on 50 % ja modulointitaajuus on 10 Hz. Kytkeä on esitetty kuviossa 2. (Barr 2013.)



KUVIO 2. Yksinkertaistettu PWM-kytkentä (Barr 2013)

## 2.1 PWM:n historia

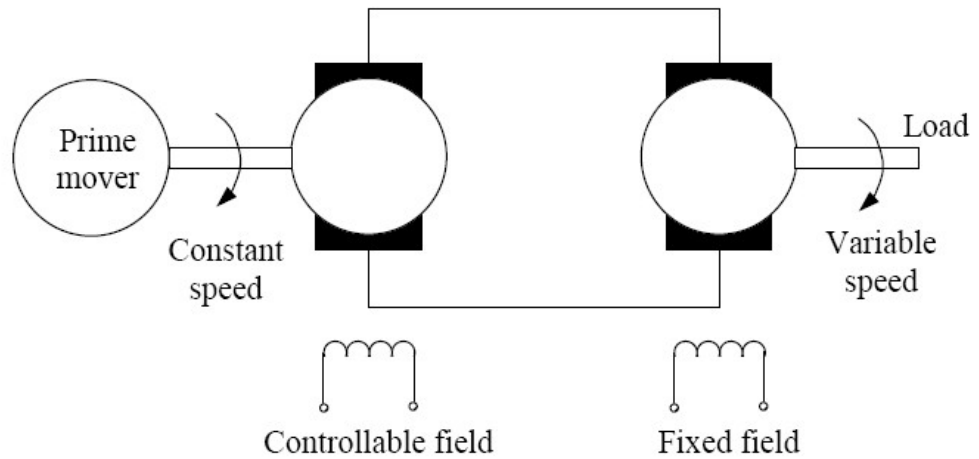
Ennen tehoa käytettiin vain osittain, kuorman kanssa sarjana kytkettiin potentiometri, joka sääti virrankulkua aiheuttaen tosin samalla tehohäviötä. Tämä tehohäviö ilmeni potentiometrin lämpenemisellä. Tämä oli tehoton tapa, mutta siedettävä, koska kokonaisteho oli pieni. Tämä oli yksi monista käytetyistä metodeista tehon säätöön. Osa tämän ajan järjestelmistä on käytössä edelleen kuten automaattitransistoreista koostuva "Australstat". Tätä käytetään valaistukseen. Käytössä oli myös Variac, jota käytettiin vaihtovirran säätöön. Näissä järjestelmissä oli suhteellisen hyvä teho, mutta ne olivat hintavia. Kuviossa 3 on esimerkki Variacista. (Schönung & Stemmler 2010.)





KUVIO 3. Varic AC-virran säätöön. (Harvard Edu 2013)

Kyseisenä aikana on ollut saatavilla joitakin nopeussäätömoottoreita, joilla on ollut siedettävä hyötysuhde, mutta ne olivat paljon monimutkaisempia kuin tasanopeusmoottorit. Ne ovat myös vaatineet kestäviä ulkoisia elektronisia laitteita, kuten tehovastuspankkeja tai pyöriviä muuntimia kuten Ward Leonard drive. Kuviossa 4 esimerkki Ward Leonard drive -toimintaperiaatteesta. (Drakos 2010.)



KUVIO 4. Ward Leonard driven toimintaperiaate. (Drakos 2010.)

Kuitenkin muille moottoreille, kuten tuulettimet, pumput ja robottiservot, oli tarvetta saada kompakti ja halpa ratkaisu, joilla näitä voitaisiin ohjata. Yksi ensimmäisistä toteutuksista, joka käytti PWM:a oli Sinclar X10. Sinclar X10 oli 10 W:n vahvistin, joka saapui markkinoille 1960-luvulla. Samoihin aikoihin PWM:a ruvettiin käyttämään vaihtovirtamoottoreiden ohjaukseen.

## 2.2 PWM:n toimintaperiaate

PWM käyttää kantiaaltoa, jonka pulssin leveyttä moduloidaan aaltomuodon keskiarvon variaatioina. Aaltomuodon keskiarvo saadaan laskettua kaavalla 1:

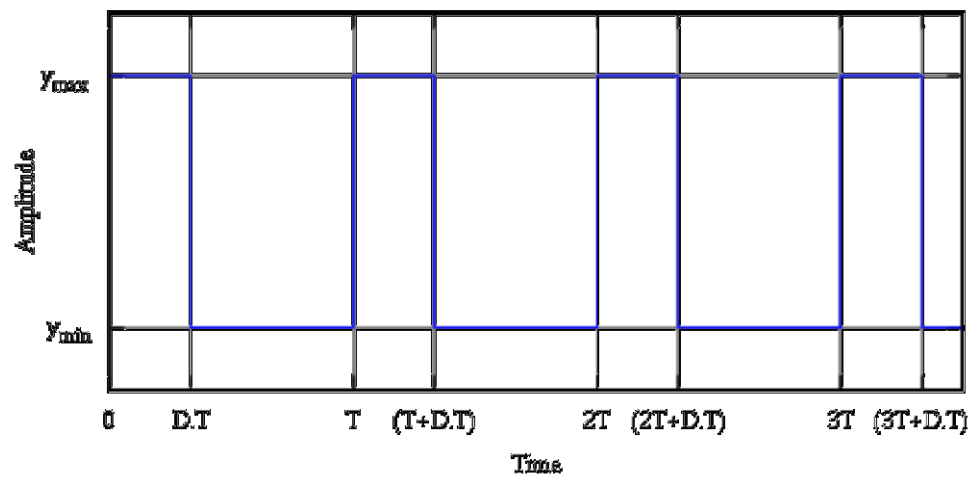
$$\bar{y} = \frac{1}{T} \int_0^T f(t) dt \quad (1)$$

$f(t)$  on pulssiaalto ja sen  $y_{max}$  arvo on  $0 < t < D \cdot T$  ja minimi arvo  $y_{min}$  on  $D \cdot T < t < T$ , yllä olevasta lausekkeesta tulee kaavan 2 mukainen:

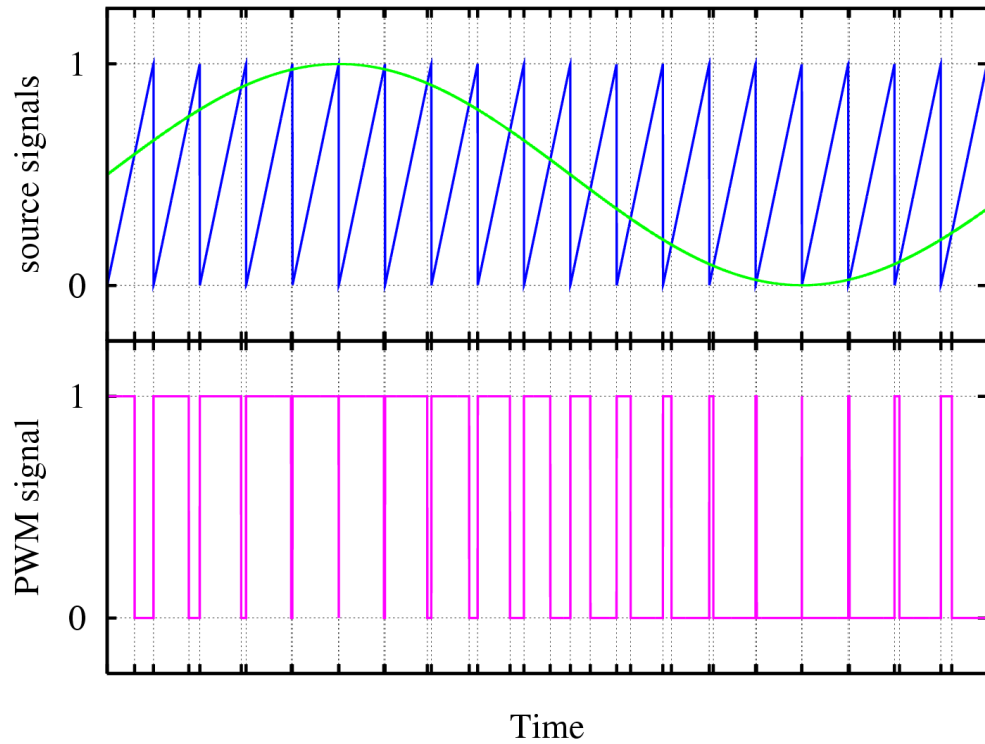
$$\begin{aligned} \bar{y} &= \frac{1}{T} \left( \int_0^{DT} y_{max} dt + \int_{DT}^T y_{min} dt \right) \\ &= \frac{D \cdot T \cdot y_{max} + T (1 - D) y_{min}}{T} \\ &= D \cdot y_{max} + (1 - D) y_{min}. \end{aligned} \quad (2)$$

Tämä kaava voidaan kuitenkin yksinkertaistaa muotoon missä  $y_{\min} = 0$ ;  $y = D \cdot y_{\max}$ . Tämä sen takia, koska signaali  $y$  on suoraan riippuvainen duty cyclestä  $D$ .

Yksinkertaisin keino generoida PWM-signaalia on leikkaus-metodi, johon tarvitaan kolmioaaltoa ja komparaattori. Kun referenssisignaali on suurempi kuin moduloitava signaali, PWM-signaali on aktiivinen, muulloin signaali on nolla. Tämä selittyy paremmin kun tarkastellaan kuvioita 5 ja 6. (Huang, Padmanabhan & Collins 2011.)



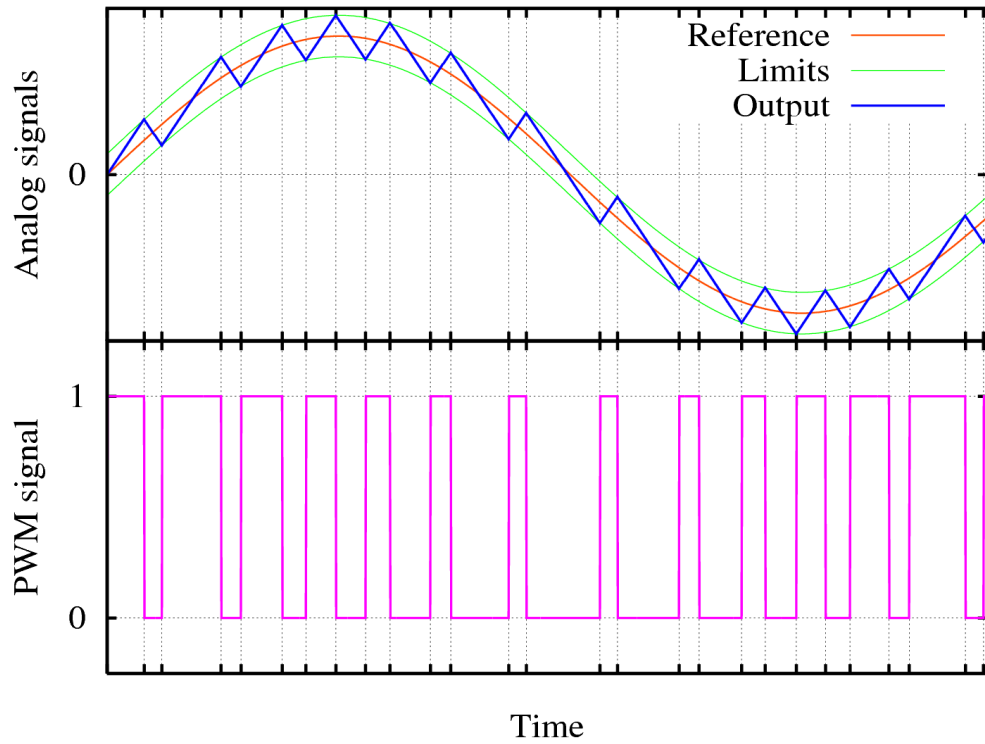
KUVIO 5. Amplitudi ajan suhteen PWM:ssä (Wikipedia 2013)



KUVIO 6. PWM-signaalin muodostuminen. (Wikipedia 2013)

### 2.2.1 Delta-modulointi

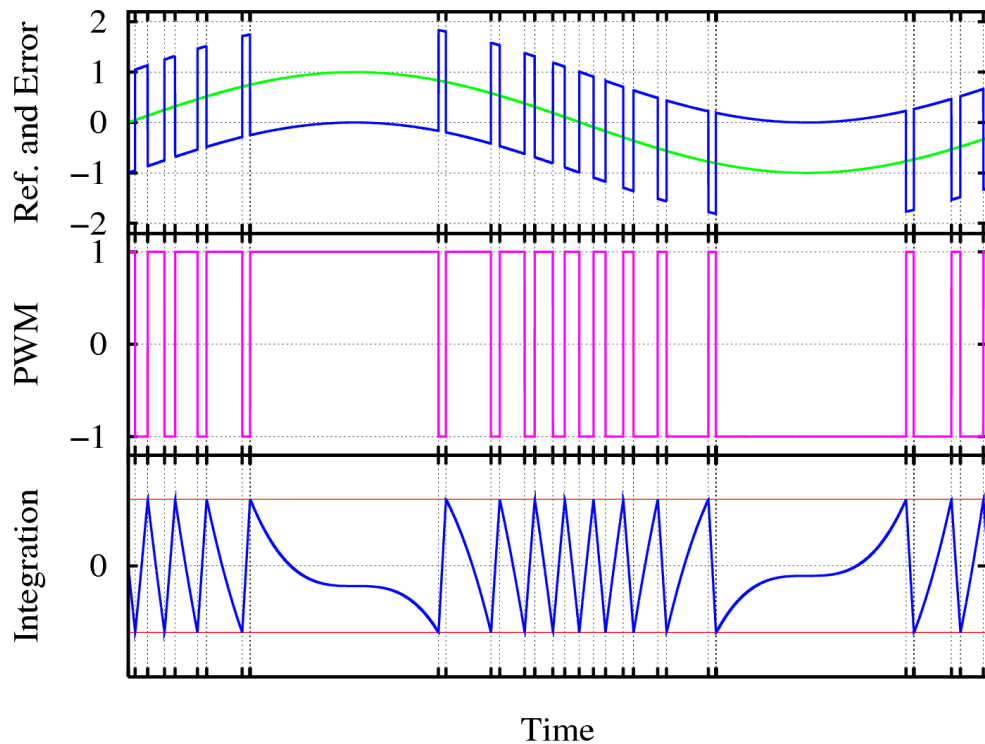
Delta-modulointi PWM-ohjauksessa tapahtuu siten, että lähtösignaali integroidaan ja tulosta verrataan rajoihin. Rajat on määritetty referenssisignaaliilla jota on siirretty vakiolla. Joka kerta kun lähtösignaali ylittää jommankumman rajoista, PWM-signaali vaihtaa tilaansa. Esimerkkinä tästä on kuvio 7. (CML Microsystems Plc. 2012)



KUVIO 7. Delta-moduloinnin periaate. (Wikipedia 2013)

### 2.2.2 Delta-sigma-modulaatio

Delta-sigma-modulaatio on yksi PWM-ohjauksen muoto. Tulosignaali vähennetään referenssisignaalista, jotta saadaan luotua virhe-signaali. Tämä virhe integroidaan, ja kun virheen integraali ylittää rajan, lähtö muuttaa tilaa. Tästä esimerkkinä kuvio 8. (Beis 2008.)



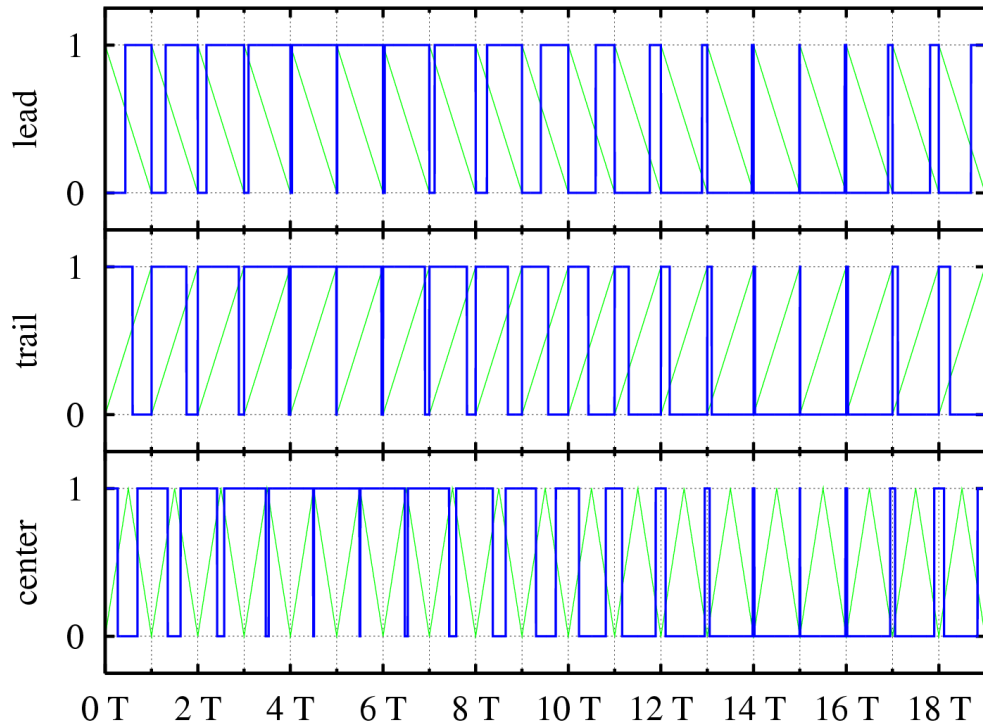
KUVIO 8. Sigma-delta-moduloinnin periaate. (Wikipedia 2013)

### 2.2.3 PWM-tyypit

Kolme erilaista PWM-tyyppiä on mahdollista toteuttaa:

1. Pulssin keskus voi olla kiinteä piste aikaikkunan keskellä, ja pulssin molemmat reunat tiivistyvät tai laajenevat leveyssuunnassa.
2. Pulssin takareuna on paikoillaan ja etureuna laajenee.
3. Pulssin etureuna pysyy paikoillaan ja takareuna laajenee.

Tästä syntyvä spektri, joka on nähtävissä kuviossa 9, huomataan, että kuviot ovat samanlaiset. Sivutaajuus sisältää moduloituneen signaalin ja vaihe moduloituneita kantajia jokaisella pulssin harmonisella taajuudella. Harmonisten ryhmien amplitudi on rajoitettu  $\sin x$  (sini funktiolla) ja kasvatettu äärettömään.



KUVIO 9. PWM:n kolme erilaista tyyppiä (Wikipedia 2013)

### 2.3 Tehon siirto

PWM:a voidaan käyttää kontrolloimaan, miten paljon tehoa siirretään kuormalle, ilman että syntyy suurempia häviöitä, esimerkiksi verrattuna resistiiviseen tapaan. Mahdollisia heikkouksia tälle tavalle on kuitenkin duty cycle, kytkemistaajuus ja kuorman ominaisuudet. Riittävän suuri kytkentätaajuus ja passiivisten komponenttien käyttö auttaa tasoittamaan pulssin rasitusta ja analoginen aaltomuoto saadaan takaisin. (Hirzel 2011.)

Suurtaajuuksinen PWM-teho-ohjausjärjestelmä on helpohko toteuttaa puolijohdekytkimillä. Kuten on jo mainittu, tehoa ei juuri häviä kääntäessä kytkintä on- tai off-asentoon. Täytyy kuitenkin muistaa, että jännite ja virta eivät ole täysin nollassa, joten häviötä tapahtuu kytkennässä. Mutta mikäli tilaa käännetään nopeasti täydestä nolnaan (yleisesti alle 100 nanosekuntia) tehohäviö on paljon pienempi verrattuna tehoon, jota syötetään kuormalle. (Hirzel 2011.)

Modernit puolijohdekytkimet, kuten MOSFET:t tai IGBT:t, ovat erittäin sopivia komponentteja PWM-ohjaukseen. Taajuusmuuntimissa, joita käytetään

vaihtovirtamoottoreissa, voi hyötysuhde olla yli 98 %. Tehokytkinlähteillä hyötysuhde on pienempi, koska lähtöjännite on pieni, usein pienempi kuin 2 V, koska nykyiset mikroprosessorit ovat erittäin energiatehokkaita. Kuitenkin niiden hyötysuhde on 70–80 %. (Hirzel 2011.)

Moninopeustuulettimet, joita käytetään esimerkiksi tietokoneissa, käyttävät PWM:ää, koska se on paljon tehokkaampi tapa verrattuna potentiometrillä toimivaan. Jälkimmäistä ei ole edes järkevää ohjata elektronisesti; se tarvitsisi toimiakseen pienen moottorin. (Huang 2011.)

Valojen himmentimet kotiloissa käyttävät tietynlaista PWM-ohjausta. Himmentimissä on yleensä elektroninen piiri, joka vaimentaa virran kulkua ennalta määrätyn osuuden verran jokaisesta AC-jännitelinjansyklistä. Lampun kirkkauden säädössä on kyse, mikä jännite (tai vaihe) vaihtojännitteen puolijaksolla alkaa päästää virtaa lamppuun. Tässä tapauksessa duty cycle on vaihtojännitesyklin, joka määräytyy taajuuden mukaan (50 Hz tai 60 Hz), ja johtamisajan suhde. (Apex Microtechnology 2013.)

Nämä himmentimet ovat varsin yksinkertaisia, ja niitä käytetään yleensä hitaasti reagoivien valolähteiden kanssa, kuten lamput. Toisenlaisissa valolähteissä, kuten ledeissä, jotka kytkeytyvät nopeasti, PWM-ohjaus voi aiheuttaa vilkkumista. Tämä kuitenkin saadaan poistettua, mikäli nostetaan PWM-ohjauksessa käytettävää taajuutta. Mikäli vilkkuminen tapahtuu tarpeeksi nopeasti, ihmisilmä ei pysty reagoimaan vilkkumiseen, vaan se näyttää siltä, että valo on koko ajan päällä. (Barr, 2013)

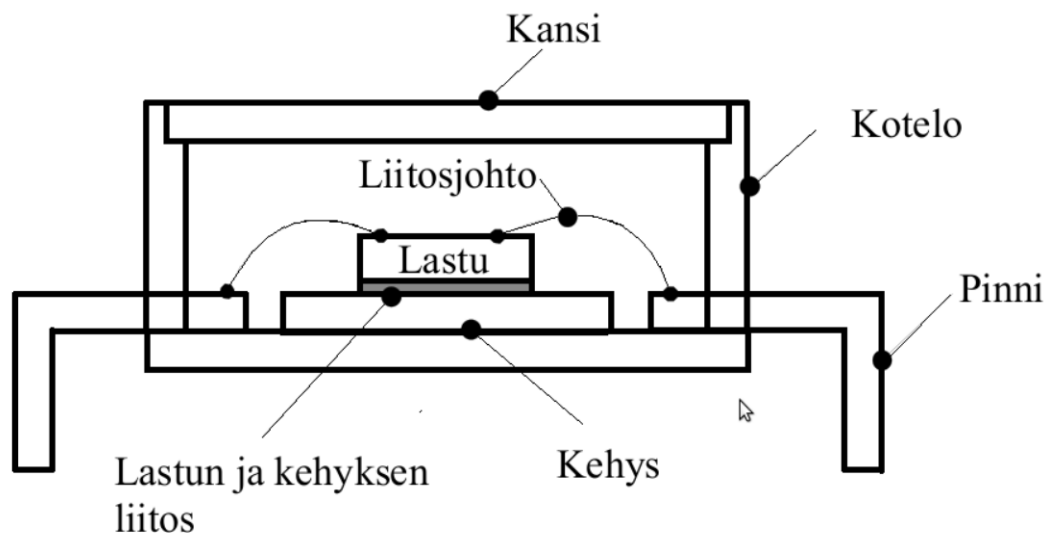


### 3 LÄMPÖ ELEKTRONIIKAN LAITTEISSA

Liitos on kahden erityyppisen puolijohteen välinen kapea vyöhyke. Esimerkiksi transistorissa on kaksi liitosta, mutta diodissa on vain yksi liitos. Lämmönsiirrossa liitos ymmärretään kohtana, jossa elektronit liikkuvat. Tästä voidaan siis päätellä, että liitokset ovat komponenttien kuumimmat paikat. Pii-pohjaisilla puolijohdekomponenteilla on määritetty maksimiarvoksi  $T_{\max} = 125\text{ °C}$ , koska on katsottu että tässä lämpötilassa komponentin toiminta ei ole oleellisesti muuttunut ja komponentti pystyy toimimaan oikein. Kuitenkin  $T_{\max}$  arvoa pienempi toimintalämpötila on suositeltavaa, koska tällöin komponentti ei rasitu liikaa ja tämä takaa komponentille pidemmän iän. (Lappeenranta university of technology 2013.)

#### 3.1 Komponentin kotelo

Kotelon tehtävä on estää komponentin sydämen joutumista suoraan kosketukseen ympäristön kanssa. Piisiru, joka muodostaa komponentin sydämen, on kotelon sisällä. Kotelomateriaali on yleensä muovia, lasia tai keraamia. Puolijohdekomponentin kotelon sisältö on esitettyinä kuviossa 10. (Lappeenranta university of technology 2013.)



KUVIO 10. Puolijohdekomponentin kotelo (Lappeenranta university of technology. 2013)

Piisirua ei voida suoraan liittää kotelon pohjaan, koska esimerkiksi muovilla lämpölaajenemiskerroin on noin 20 kertaa suurempi kuin piillä, joten tästä seuraisi suurta termistä rasitusta. Tähän siis tarvitaan erillinen kehys, jonka lämpölaajenemiskerroin on lähes sama kuin piillä. Kehysmateriaalina käytetään usein kuparia. (Lappeenranta university of technology 2013.)

Termisessä suunnittelussa ensimmäinen asia, johon kannattaa kiinnittää huomiota, on kotelo, koska tämä on ensimmäinen vaihe komponentin tuottaman häviölämmön poistamiseksi. Tämä hukkateho saadaan siirrettyä sirusta koteloon johtumalla, konvektiolla sekä säteilemällä. On kuitenkin hyvä muistaa, että yleensä kotelo suunnitellaan sähkötekniisin perustein, jolloin lämpötekniiset seikat saattavat jäädä vähemmälle huomiolle. (Lappeenranta university of technology 2013.)

### 3.2 Lämmönsiirron analogia

Sähköanalogian avulla voidaan määrittää lämmönsiirto kiinteiden kappaleiden välillä, mutta tähän tarvitaan lämpöresistanssi  $R_{th}$ . Sähköanalogia kertoo, että potentiaaliero  $\Delta U$  vastaa lämpötilaeroa  $\Delta T$ , resistanssi  $R$  lämpöresistanssia  $R_{th}$  sekä sähkövirta  $I$  lämpövirtaa  $q$ , joilloin voidaan soveltaa Ohmin lakia, kuten kaavassa 3 ja 4.

$$\mathbf{R = \Delta U / I} \quad (3)$$

$$\mathbf{R_{th} = \Delta T / q} \quad (4)$$

Kotelon täytteenä on tavallisesti kaasua, joka on huono lämmönjohde. Itse kotelo on usein tehty muovista, joka on myös huono lämmönjohde. Tästä syystä suojakotelon ja piisirun välinen lämpöresistanssi on korkea, joten myös lämpötilaero on suuri kotelon sisällä. Sirun ja kotelon välinen lämpöresistanssi saattaa vaihdella välillä  $10 \text{ }^\circ\text{C/W} - 100 \text{ }^\circ\text{C/W}$ , joka on erittäin suuri.

Lämpöresistanssi vaihtelee suuresti eri sovelluksissa. (Lappeenranta university of technology 2013.)

#### 4 SÄHKÖLAITTEEN JÄÄHDYTYSTARPEEN MÄÄRITTÄMINEN

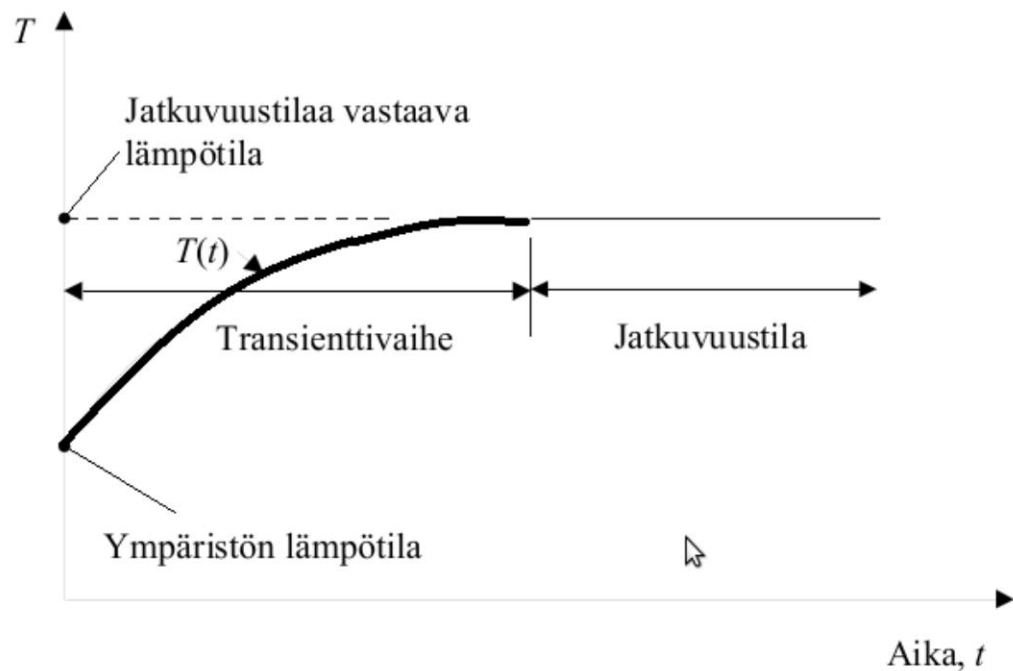
Ensimmäinen tehtävä on laskea systeemissä syntyvä häviöteho eli jäähdytyskuorma. Termodynamiikan ensimmäisessä pääsäännössä sanotaan, että systeemiin tuotavan energian on oltava tasapainossa poistuvan energian kanssa. Voidaan olettaa, että systeemistä lähtevä teho on sen resistansseissa syntyvää lämpöä, joten laitteen häviöteho eli jäähdytyskuorma on yhtä suuri tehonkulutuksen kanssa. Tätä ajatusta voidaan soveltaa myös yksittäisten komponenttien kanssa, mutta täytyy kuitenkin muistaa, että suurimassa osassa sähkölaitteista lähtevä teho on sähköenergiaa (mm. taajuusmuuntaja, tutka). Näissä tapauksissa laitteen lämpökuorma on määritetty otto- ja lähtötehon erotuksena. Lämpökuorman laskenta voidaan myös toteuttaa askel askeleelta, eli määritetään jokaisen systeemin komponentin tehohäviö ja lasketaan nämä yhteen. (Lappeenranta university of technology 2013.)

Kun edellä mainittu on saatu laskettua, on hyvä lisätä laskelmiin varmennusmarginaali. Tämä sen takia, että mikäli systeemiin vaihdetaan osia tai lisätään uusia komponentteja, ei lämpölaskentaa tarvitse heti tehdä uudestaan. Silloin voidaan luottaa siihen, että systeemin jäähdytys on tarpeeksi tehokas. Varmennusmarginaalin kanssa on oltava kuitenkin tarkkana, koska jos systeemin varmennusmarginaali ylimitoitetaan suuresti, se kasvattaa laitteen kokoa ja tuo lisäkustannuksia. (Lappeenranta university of technology 2013.)

Käyttöympäristö tulee ottaa tarkasti huomioon laitetta suunniteltaessa. Huomioon otettavia asioita ovat mm. käyttöympäristön lämpötila ja sen likaisuus.

Jäähdytyksen suunnittelussa tulee myös ottaa huomioon laitteen toimintajakso eli se osa kokonaisajasta, jonka laite on päällä. Mikäli laitteen toimintajakso on alhainen, niin laitteen nimellinen häviöteho on pienempi. Esimerkiksi nimelliseltä häviöteholtaan oleva 5 W:n transistori, jota käytetään vain 2 W:n teholla, jolloin kyseinen transistori on aktiivinen 40 % kokonaisajasta. Mitoituksen kannalta on tärkeää tietää, saavuttaako komponentti toimintajakson aikana tasapainotilaa vastaavan käyttölämpötilan. (Lappeenranta university of technology 2013.)

Sähkölaitteen ollessa poissa päältä on se ympäristönsä kanssa termisessä tasapainotilassa eli laitteen lämpötila on sama kuin ympäristön. Laitteen käynnistyessä sen komponenttien ja samalla koko laitteen lämpötila alkaa nousta. Laitteen lämpötila jatkaa nousua, eli laite on transienttivaiheessa, kunnes saavutetaan jatkuvuustila, eli jäähtytysjärjestelmän pois kuljettama lämpö on yhtä suuri kuin laitteen komponenttien yhteenlaskettu häviöteho. Laitteen lämmönkehitys esitetty kuviossa 11. (Lappeenranta university of technology 2013.)



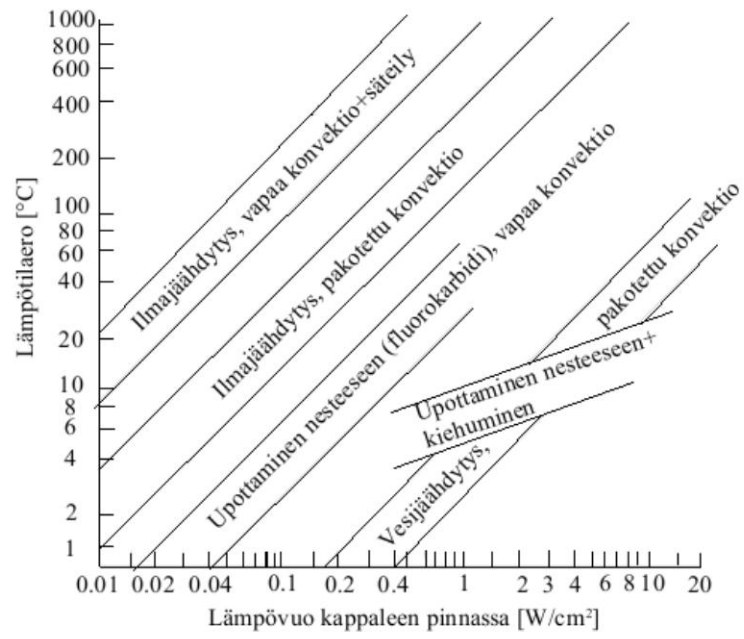
KUVIO 11. Sähkölaitteen lämpötilan kehitys ajan funktiona. (Lappeenranta university of technology 2013.)

Mikäli lämpötila vaihtelee, tämä saattaa aiheuttaa lämpöjännityksiä, jotka puolestaan heikentävät laitteen luotettavuutta. Kokeellisten analyysien avulla on todettu, että yli 20 °C vaihtelut lämpötilassa kasvattavat sähkölaitteiden vikataajuutta noin kahdeksankertaiseksi. Lämpöjännityksestä johtuvat mekaaniset värähtelyt ja lämpöiskut ovat erittäin yleisiä vikojen aiheuttajia ja näin ollen ne täytyy ottaa huomioon mitoituksessa. (Lappeenranta university of technology 2013.)

Sähkö- ja elektroniikkalaitteiden jäähdytysmenetelmät vaihtelevat laajasti, riippuen minkälaisesta sovelluksesta on kyse. Esimerkiksi lentokone-elektroniikassa käytetään pakotettua konvektiota, jolloin jäähdytyksestä vastaa kompressorin tuottama paineilma. Laivoissa ja sukellusveneissä jäähdytys on suoritettu vesijäähdytetyillä lämmönsiirtimillä. (Lappeenranta university of technology 2013.)

Suuritehoisissa mikroaaltosovelluksissa syntyvä häviöteho on suuri, koska näiden energianmuuntohyötysuhde on huono. Tällaisessa sovelluksessa lämpövuoto voi olla jopa  $2000 \text{ W/cm}^2$ . Jotta lämpö saadaan siirrettyä turvallisesti pois, sovellus on yleensä upotettu dielektriseen nesteeseen, joka siirtää lämmön pois kiehumislämmönsiirron avulla. (Lappeenranta university of technology 2013.)

Laite- ja komponenttivalmistajat ilmoittavat datakirjoissaan maksimaalisen lämmönsiirtokyvyn ja suurimman sallitun käyttölämpötilan. Nämä kaksi arvoa auttavat alustavasti valitsemaan sopivan jäähdytysmenetelmän. Kun laitteen tai komponentin häviöteho on tiedossa, voidaan sen lämpövuoto laskea jakamalla häviöteho laitteen tai komponentin lämmönsiirtopinnalla. Ottamalla huomioon laitteen tai komponentin pinnan ja ympäristön välisen lämpötilaeron saadaan laitteen jäähdytykseen soveltuva lämmönsiirtomuoto määritettyä kuvion 12 avulla. Esimerkiksi käytettäessä vapaaseen konvektioon perustuvaa ilmajäähdytystä komponenttiin, jonka lämpövuoto on  $0,5 \text{ W/cm}^2$ , tulisi komponentin pinnan ja sen ympäröivän ilman lämpötilaero olemaan  $500 \text{ }^\circ\text{C}$ , eli vapaa konvektio ei tule toimimaan tässä tilanteessa. Ilmajäähdytys on mahdollista, mikäli voidaan käyttää pakotettua konvektiota eli voidaan käyttää tuuletinta. Täytyy kuitenkin muistaa, että mikäli lämpövuoto on  $1 \text{ W/cm}^2$  tai yli, ei pakotettu konvektio ole enää järkevä vaihtoehto, ellei ole mahdollista käyttää suurta jäähdytyslevyä tuulettimen lisäksi. Suurten lämpövoimien tapauksessa on viisaampaa käyttää pakotettuun konvektioon perustuvaa nestejäähdytystä. (Lappeenranta university of technology 2013.)



KUVIO 12. Eri lämmönsiirtotavoilla saavutetut lämpövuot. (Lappeenranta university of technology 2013)

## 5 JÄÄHDYTYSJÄRJESTELMÄ

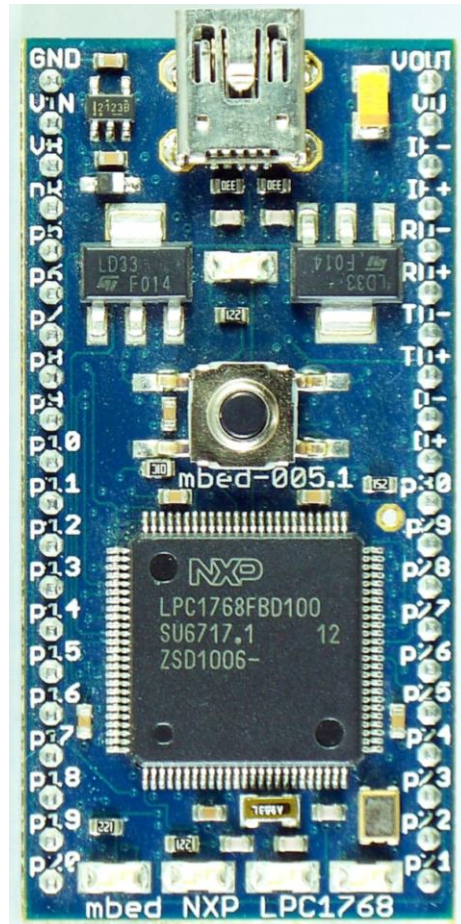
Ajatus rakentaa automatisoitu jäähdytysjärjestelmä syntyi seuraavanlaisen pohdinnan seurauksena: Miten voisin hyödyntää koulussa opittua tietoa ja yhdistää sitä sellaiseen mitä koulussa ei ole tehty. Tässä järjestelmässä pääosassa on PWM ja lämpösuunnittelu. PWM:a ei ole koulussa opetettu juuri ollenkaan, vaikka minusta se on yksi tärkeimmistä järjestelmistä ajatellen moottoreiden ohjausta jne. Toinen tärkeä aihe on lämpösuunnittelu, jota koulussa on opetettu jonkin verran. Koulussa opittua aihetta voi soveltaa käytäntöön, koska ilman lämpösuunnittelua, laite tai järjestelmä ei tule toimimaan kunnolla.

Ideana oli rakentaa jäähdytysjärjestelmä tietokoneisiin, mutta koska tietokoneiden emolevyt pystyvät tekemään saman kuin oma laitteeni, niin ideaa on jatkojalostettu siten, että laite käy myös moniin muihin järjestelmiin. Vaikka tietokoneiden emolevyissä on paikat tuulettimille ja ne voivat myös ohjata tuulettimia, on niiden säätö vaikeaa, sekä tarvitaan erillisiä ohjelmia, jotka näyttävät lämpötilan ja tuulettimen nopeuden. Tekemälläni laitteella kuitenkin tuulettimien ohjaus toimii automaattisesti tai manuaalisesti, ja lämpötila on nähtävissä ilman erillisiä ohjelmia.

### 5.1 Suunnittelu

Järjestelmän toiminnallisuudesta vastaa mikrokontrolleri, mbed NXP LPC1768. Kyseisessä kontrollerissa on ARM Cortex M3 -ydin, jonka kellotaajuus on 90 MHz, ja 512 kB:n flash-muisti ja 64 kB:n RAM-muisti. Kontrollerista löytyy myös erilaisia liitäntöjä, kuten ethernet, CAN, SPI ja muita I/O-liitäntöjä. Kontrolleri on nähtävissä kuviossa 13.





KUVIO 13. Mbed NXP LPC1768

Tällä kontrollerilla hoidetaan siis kaikki toiminnallisuus, jota kytkentään kuuluu, kuten PWM, luku ja kirjoitus näytölle. PWM-ohjaus tapahtuu pinneissä 21 – 26. Muita pinnejä voidaan käyttää esimerkiksi I/O-pinneinä, joilla voidaan ohjata LCD-näyttöä.

Kontrollerin jälkeen täytyi valita tehtävään sopiva LCD-näyttö. Näytön tärkein valintakriteeri oli, että näyttö tukee Hitachin HD44780-standardia. HD44780 helpottaa laitteen ohjelmointia, koska kyseiseen standardiin on valmiina monia eri kirjastoja. Kyseinen standardi on yleisesti käytössä eri valmistajilla, joten on monia eri vaihtoehtoja. Toinen kriteeri, joka vaikutti valintaan, oli näytön hinta. Näyttö ei saisi maksaa liikaa, ajatellen laitteen kokonaiskustannuksia ja tämän myötä sarjatuotantoa. Nämä kaksi kriteeriä huomioiden valitsin järjestelmäni Fordata Electronic LTD:n FDCC1602B-FLYYBW-51LK-näytön. Näyttö on nähtävissä kuviossa 14.



KUVIO 14. Fordata Electronic LTD FDCC1602B-FLYYBW-51LK

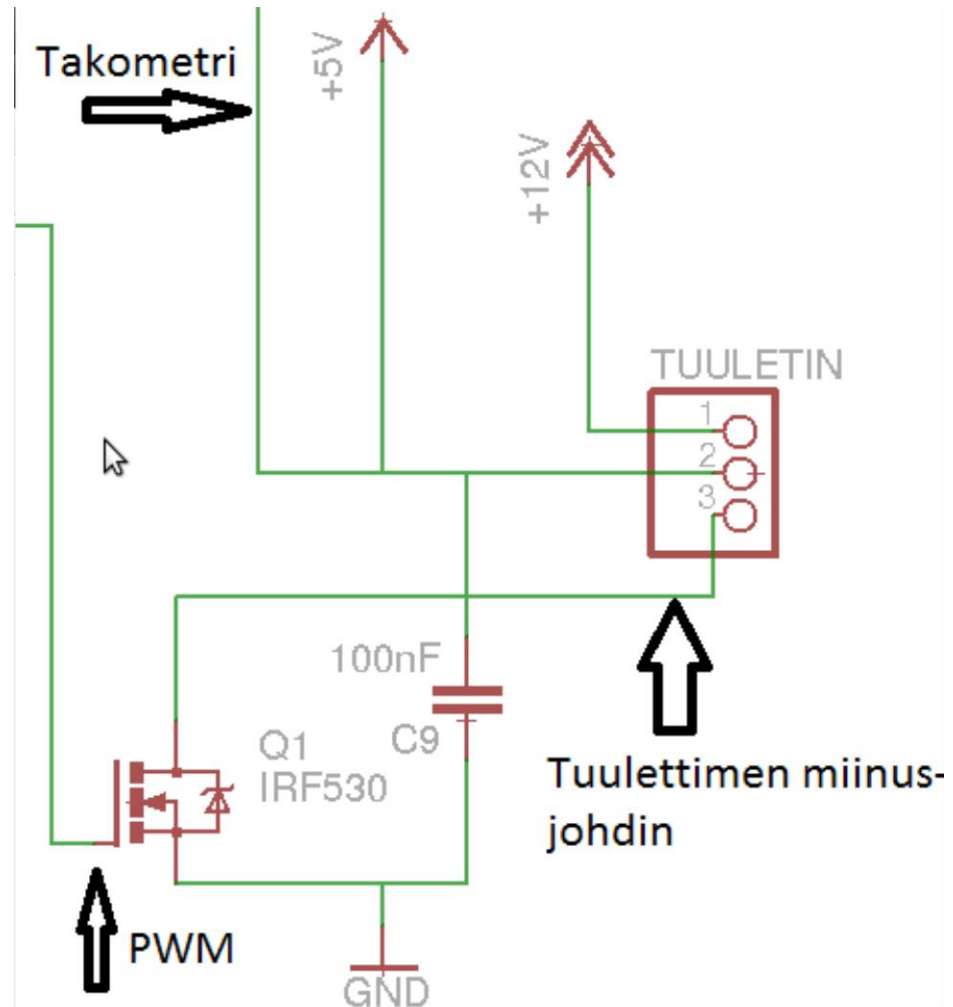
Tuulettimeksi valitsin Nexus D12SL-12. Kyseinen tuuletin on 12 V:n jännitteellä toimiva tuuletin ja se käyttää 0,3 A:n virtaa. Nexus on iso tuulettimien valmistaja ja tunnettu maailmalla. Tuuletin on nähtävissä kuviossa 15.



KUVIO 15. Nexus D12SL-12.

Tuuletin muodosti ensimmäisen ongelman, koska tuuletin tarvitsee 12 V toimiakseen täydellä teholla, ja mbedin PWM-ohjauksen lähtö toimii välillä 0 V – 3,3 V, joten tästä voidaan jo päätellä, että ohjaus ei tule toimimaan suoraan. Mbedin ja tuulettimen väliin tarvitaan jokin, jolla saadaan nostettua jännitettä. Tähän tarkoitukseen sopii esimerkiksi transistori tai MOSFET. Valitsin tähän toteutukseen IRF530 MOSFET:n, koska MOSFET:t ovat toiminnaltaan varmempia ja niiden käyttö on helpompaa kuin JFET:n. IRF530 on N-tyypin MOSFET, joka on erittäin yleisessä käytössä oleva komponentti. Kyseisen MOSFET:n jännitekesto on 100 V ja virrankesto 14 A. Näiltä osin kyseinen komponentti kestää erittäin hyvin tässä käytössä. IRF530:n kynnyksjännite on välillä 2 V – 4 V, eli kanava alkaa aueta, kun MOSFET:lle tulee 2 V, ja kanava on täysin auki, kun jännitettä on 4 V. Tässä on nähtävissä toinen ongelma, koska mbedin PWM-lähtö kykenee tuottamaan maksimissaan vain 3,3 V, joten tämä ei riitä saamaan kanavaa täysin auki, jolloin tuulettimen ohjaus ei toimi kunnolla. Tämäkin ongelma voidaan hoitaa hyvin yksinkertaisella menetelmällä. Muutetaan pinnijärjestys seuraavanlaiseksi: source-elektrodi on vedetty maihin, drain-elektrodista lähtee veto tuulettimen maapiuhaan ja hilaan tulee veto mbediltä. Tätä

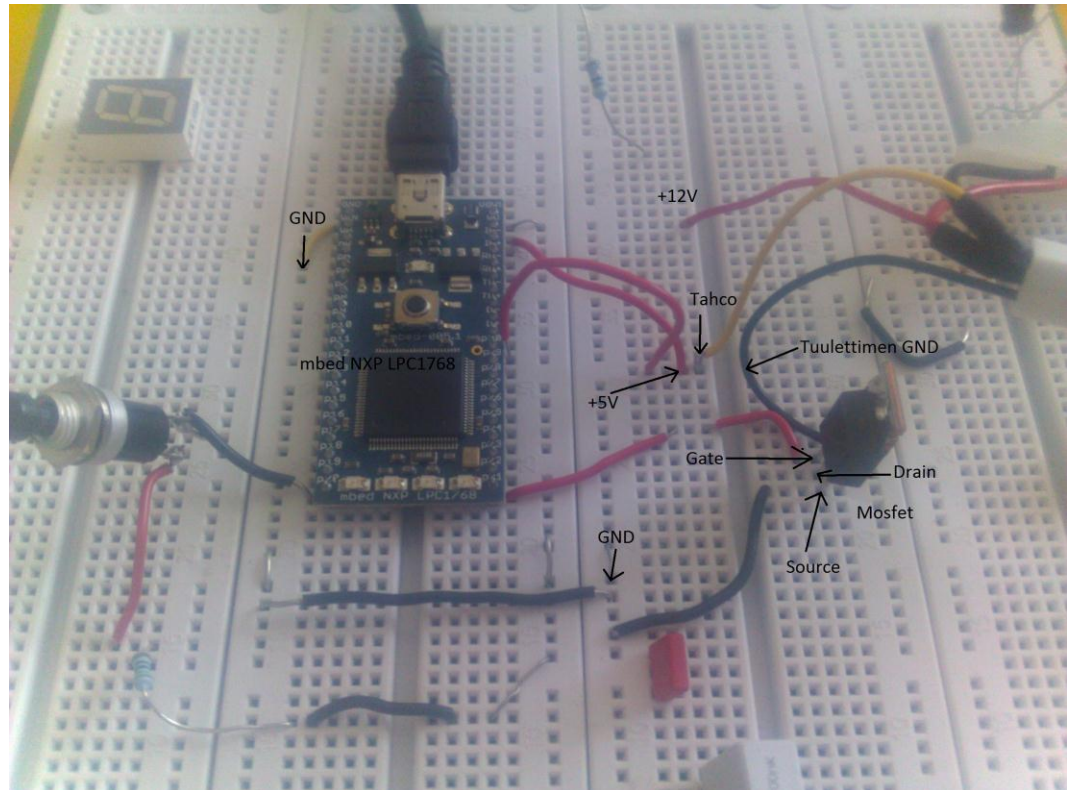
kytkentää selventää kuvio 16. Tämän jälkeen tuulettimen jännitepiuha laitetaan suoraan 12 V:n linjaan, jolloin jännite saadaan tarpeeksi suureksi ja tuuletin toimii mallikkaasti 12 V:lla.



KUVIO 16. Tuulettimen kytkentäkaavio.

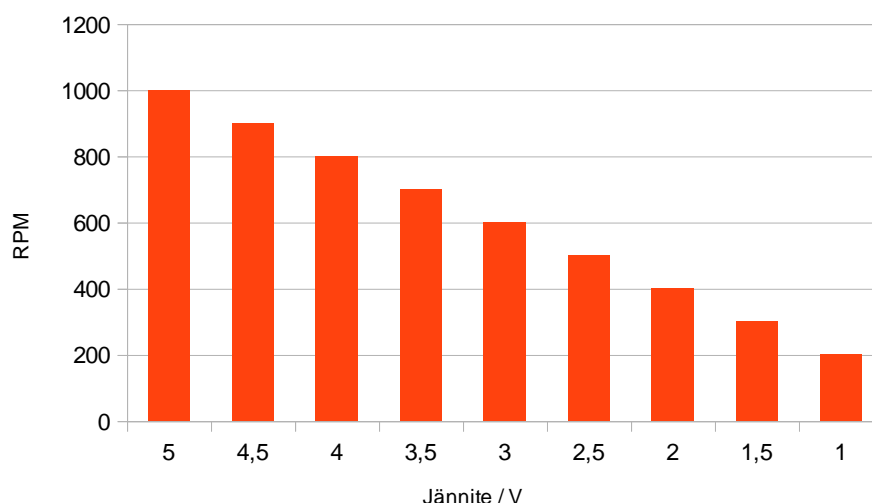
## 5.2 Testiympäristön rakentaminen

Jotta ajateltua teoriaa pääsisi testaamaan käytännössä, on rakennettava testiympäristö. Tämän järjestelmän rakensin koekytkentälevylle, jolla on hyvä tehdä ensimmäinen testiajo. Ensimmäinen testiympäristö on kuviossa 17.



KUVIO 17. Ensimmäinen testiympäristö

Kuten kuviosta voidaan nähdä, kytkentä ei ole kovin monimutkainen. Tärkeimmät asiat, jotka täytyy huomioida, ovat MOSFET:n kytkentä ja takometrin kytkentä. Kuten jo aikaisemmin todettiin, MOSFET kytketään siten, että source-eketrodi menee maihin, tuulettimen maapiuha kytketään MOSFET:n drain-eletrodiin ja PWM-lähtö tulee hilaan. Takometrin kytkennässä tulee ottaa huomioon se, miten takometri toimii. Takometrissä on sama jännite kuin tuulettimella, joten mikäli se kytketään suoraan kontrollerin I/O-nastaan ja tuuletinta ajetaan täydellä teholla, kontrollerin I/O-pinni tulee hajoamaan. Tämän takia takometrin kytkentään on viety +5 V:n linja, jolla estetään se, ettei jännite nouse yli 5 V:n. Tämän avulla voidaan laskea helposti myös kierrosnopeus, koska kun tuuletin toimii täydellä teholla, takometri antaa arvon 5 V. Tästä voidaan siis laskea helposti kierrosnopeus kun tiedetään, että tuulettimen maksimikierrosnopeus on 1000 rpm. Tästä tapahtumasta on havainnollistava kuvio 18. Järjestelmään lisättiin testin jälkeen myös lämpöanturit, joiden kalibrointi ei ollut kaikista helpoin tehtävä. Ensiksi oli saatava sovitettua anturi toimimaan 3,3 V:n logiikan kanssa. Tämä oli kuitenkin helposti toteutettavissa ohjelmallisesti, joten tästä ei muodostunut sen suurempaa ongelmaa.



KUVIO 18. Kierrosnopeus jännitteen suhteen

### 5.3 Testiajosta tehtävät päätelmät

Järjestelmän testeissä ei havaittu mitään ongelmaa. Mbedin ansiosta PWM-tehonsäätö oli helppoa, koska PWM:ää pystyttiin säätämään prosenttiluvuilla, joka kuvasti tämän tehoa välillä 0,0 V – 3,3 V. Testiä suoritettiin yhtäjaksoisesti noin viiden tunnin ajan, vaihtaen nopeutta minuutin välein. Järjestelmälle suoritettiin myös yön yli kestävä koe, jossa tuuletin säädettiin pyörimään 600 kierrokseen minuutissa. Järjestelmä selvisi näistä testeistä hyvin. Itse sensoreiden testaus oli hieman vaikeampaa, koska luotettavaa vertauskohtaa ei löytynyt testeihin. Myös lämpötilan nousun ja laskun tuottaminen oli hankalaa. Koska tähän ei löytynyt luotettavaa tapaa, joudutaan oletamaan, että anturin antama lämpöarvo on enemmän suuntaa-antava kuin tarkka arvo.

Testien tulokset ovat kuitenkin positiiviset, joten tästä ensimmäisestä vaiheesta voidaan siirtyä seuraavaan vaiheeseen, jossa tarkoituksena on siirtää kytkentä koekytkentälevyltä piirilevylle.

### 5.4 Piirilevysuunnittelu

Piirilevysuunnitteluun on olemassa monia erilaisia ohjelmia. Itse valitsin Eaglen. Eagle ( Easily Applicable Graphical Layout Editor) on varsin yleinen

suunnitteluohjelma, joka on CadSoftin valmistama. Eaglesta löytyy kahta erillistä versiota, maksullinen ja maksuton. Maksuttoman ja maksullisen ohjelman erona on se, että maksuttoman version piirilevyn maksimikoko, joka voidaan tehdä, on kymmen kertaa kymmenen senttimetriä. Eaglessä on myös hyvää se, että ohjelmaan voi ladata ilmaisia kirjastoja erilaisista komponenteista, joten niitä ei tarvitse itse tehdä.

Kytkenän lohkokaaavion tekeminen on yksinkertaista. Ensiksi on etsittävä oikeat komponentit, joita tarvitaan, tämän jälkeen on tehtävä tarvittavat kytkennät. Tässä suurena apuna toimivat komponenttien datakirjat, joista löytyy kaikki tarvittava tieto komponentteihin liittyen. Kytkenän lohkokaavio on liitteessä 1.

Itse piirilevyn suunnittelussa täytyi ottaa enemmän asioita huomioon, kuten elektromagneettinen yhteensopivuus. Emc (electromagnetic compatibility) tarkoittaa sähkömagneettista yhteensopivuutta, jolla pyritään saamaan laite toimimaan luotettavasti sen luonnollisessa toimintaympäristössä. Hyvään piirilevyn suunnitteluun on monia erilaisia muistisääntöjä, joita voidaan noudattaa piirilevyn tekemisessä. Piirilevy on liitteessä 2. Muutamia hyviä esimerkkejä asioista, joita täytyy ottaa huomioon piirilevyä tehtäessä. Suotonkondensaattorit mahdollisimman lähelle suodatettavaa kohdetta, koska jos suotonkondensaattorit ovat liian kaukana kohteesta, suodosta ei ole mitään apua. Tässä myös täytyy huomioida se, että johtimien ja jalkojen impedanssi heikentää suodatusta, ja myös tästä syystä suotonkondensaattorit täytyy laittaa tarpeeksi lähelle. Johtimien tulee olla paksuja ja lyhyitä, tällä saadaan minimoitua induktanssi ja resistanssi. Suotonkondensaattorit täytyy sijoittaa symmetrisesti virtalähteen ympärille, koska näin saadaan rippelijännite vaikuttamaan jokaiseen suotonkondensaattoriin yhtä paljon.

Maadoituksessa tulee ottaa huomioon seuraavia asioita: Hyvä tapa käyttää maadoitusta on tehdä yksipistemaadoitus. Mailla on myös omat signaalinsa, joten väärin kytkettyinä nämä aiheuttavat häiriötä. Maahäiriöt voivat säteillä avoimista piirilevyn reunoista. Maadoitettu faraday-suoja on tehokas tapa ehkäistä kapasitiivista kytkeytymistä.

Linjojen tulee olla oikean kokoisia, koska liian kapeassa vedossa kulkeva virta voi aiheuttaa jännitehäviöitä vetojen päissä, jolloin voi syntyä RFI-antenni. Linjoista tulee tehdä mahdollisimman suorat, ilman suuria mutkia tai käännöksiä, koska jos kytkennässä on paljon käännöksiä, tämä saattaa luoda kelan kytkentään. Mikäli linjat ovat liian lähellä toisiaan, tämä voi aiheuttaa kapasitiivista kytkeytymistä. Kelat tulee sijoittaa oikein, ei lähelle kriittisiä signaaleja, koska kela voi aiheuttaa induktiivista kytkeytymistä.

Huomioitavaa on myös se, että kannattaa tehdä moni-kerroslevyjä, koska näin saadaan luotua omat kerrokset linjoille, kuten oma jännite- ja maakerros. Näin saadaan myös kriittiset signaalit tehtyä omalle kerrokselleen, jolloin eliminoidaan häiriöiden siirtyminen kriittisiin signaaleihin.

## 5.5 Ohjelmakoodi

Mbed-järjestelmässä on täysi C++-tuki, eli ohjelmoinnin voi tehdä joko C:llä tai C++:lla. Päädyin tekemään koodin C++:lla, koska tästä kielestä minulla on enemmän kokemusta. Mbedissä on myös online-kääntäjä, jolloin kaikki toiminta tapahtuu pilvessä, eli erillisiä ohjelmia tai kääntäjiä ei tarvita. Mbedillä on myös laaja käyttäjäkunta, joka tarkoittaa sitä, että siihen löytyy paljon erilaisia kirjastoja joita pystyy käyttämään hyväkseen. Tästä hyvänä esimerkkinä on näytönohjaus, johon löytyy valmiit kirjastot. Koko koodi on liitteenä 3.



## 6 JOHTOPÄÄTÖKSET

Projektin alussa minulla oli epäilyksenä, olisiko tämä tarpeeksi haastava, mutta epäilykseni osottautuivat vääriksi projektin alun jälkeen. Vaikka itselläni oli jo entuudestaan tietoa siitä, miten PWM toimii, kuitenkin tämä projekti auttoi minua ymmärtämään PWM:n monimuotoisuuden. PWM on suhteellisen uusi keksintö, mutta se on tuonut erittäin paljon hyötyä, varsinkin moottoreiden ohjaukseen. Vaikka PWM:ää käytetään lähes pelkästään moottoreiden ohjaukseen, sitä voidaan käyttää myös moneen muuhun asiaan.

Jo entuudestaan tiesin lämpösuunnittelun ja lämmönsiirron olevan tärkeässä roolissa laitteen toiminnan kannalta, mutta tieto, jota sain tästä projektista, vain vahvisti käsitystä aiheesta. Ilman kunnon lämmönsiirtoa tai lämpösuunnittelua mikään laite ei tule toimimaan kunnolla.

Itse laitteen teossa kohtasin erilaisia ongelmia, kuten kuinka ohjata tuulettimia. Ensimmäisessä rakennelmassa oli tiettyjä vikoja, jotka jäivät selvittämättä. Esimerkkinä tästä se, että mikäli tuulettimen takometri jätetään kytkemättä, tuuletin ei pyöri täydellä nopeudella. Mutta jos tuulettimen takometri kytketään johonkin, esimerkiksi + 5 V, tällöin tuuletin pyöri normaalisti. Lämpöantureiden kanssa oli myös ongelmana se, että kyseisistä antureista ei ole saatavissa mitään datakirjaa, joten niiden toiminta oli selvitettävä kokeilemalla erilaisia ratkaisuja.

Kaupoista löytyy myös samanlaisia laitteita, joiden hinnat ovat 15 – 75 € välillä. Halvimmissa ratkaisuissa on pelkästään potentiometrillä tapahtuva säätö, eli niissä ei ole älyä juuri lainkaan. Yli 35 € maksavissa laitteissa löytyy näyttöpaneeli, jossa näkyy tuulettimien nopeudet ja joissain versiossa myös lämpötilat. Mutta näissä laitteissa ei ole aktiivista lämpötilan tarkkailua ja sen mukaan tapahtuvaa lämpötilan säätöä. Yli 50 € maksavissa laitteissa on jo oma äly, jolla voidaan säätää tuulettimien nopeutta automaattisesti tai manuaalisesti. Automaattinen ohjaus käyttää samaa periaatetta kuin tekemässäni järjestelmässä, eli tuulettimia säädetään lämpötilan mukaan. Tästä esimerkkinä on Zalman MFC2 Multi Fan and Temp Monitor and Control Drive Bay Device kuviossa 18. Kyseisen laitteen hinta on noin 55 €.



KUVIO 18. Zalman MFC2 Multi Fan and Temp Monitor (Oc modshop 2008)

Laitteeni hintavin osa on mikrokontrolleri, NXP ARM Cortex-M3. Tämän hinta on noin 10 – 20 € ostopaikasta riippuen. Seuraavaksi kallein laite on näyttö, jonka hinta on noin 10–15 €. MOSFET:ien ja muiden oheiskomponenttien yhteishinnaksi tulee noin 10 €. Tästä voidaan päätellä, että rakentamalla laitteen itse, komponenttien hinnaksi tulee 30 – 45 €. Mikäli halutaan tehdä jonkinlainen kotelointi laitteelle, niin lopulliseksi hinnaksi tulee noin 45 – 60 €, joten hinnallisesti ei ole suurta eroa, ostaako kyseisen laitteen kaupasta vai rakentaako sen itse. Suurimmassa osassa kaupasta löytyvistä laitteista löytyy ohajus neljälle tuulettimelle. Mikäli tarve on suurempi, joutuu ostamaan toisenkin laitteen, kun taas mikäli laitteen tekee itse, on paljon helpompaa lisätä laitteeseen paikkoja tuulettimille. Tarkastelun jälkeen on huomioitavaa, ettei missään kaupoista löytyvistä laitteista löydy datan keräystä lokiin. Tämäkin voidaan lisätä itse tehtyyn laitteeseen, mikäli vain kontrollerissa löytyy tuki tähän, kuten NXP ARM Cortex-M3:sta. Omasta mielestäni itse tehty laite on toimivampi ja kannattavampi tehdä kuin ostaa kaupasta laite, mikäli vain tekijältä löytyy mielenkiintoa aiheeseen, tarvittavat välineet laitteen tekemistä varten ja riittävää tietämystä aiheesta.

## 7 YHTEENVETO

Yksinkertaistettuna PWM-ohjaus on pulssisuhteen muutoksella tehtävää ohjausta, eli kytkimen tila vaihtuu tietyllä taajuudella. Koska teho on jännitteen ja virran tulo ( $P = UI$ ), jännite tai virta on aina lähes nolla, joten myös tehohäviö on lähes nolla.

Ennen PWM-ohjausta, tehoa käytettiin vain osittain. Tehoa säädettiin potentiometreillä, joka säätö virrankulkua. Tämä oli kuitenkin tehoton tapa, koska tällä tavalla syntyi paljon tehohäviötä.

PWM:lla kontrolloidaan sitä, kuinka paljon tehoa siirtyy kuormalle, ilman että syntyy suurempia häviöitä. Tässä tavassa on myös omat heikkoudet kuten duty cycle, kytkemistaajuus ja kuorman ominaisuudet. Pulssin rasiudesta voidaan vähentää, mikäli on tarpeeksi suuri kytkentätaajuus. Passiivisten komponenttien käyttö auttaa analogisen aaltomuodon takaisin saamisessa.

Liitokseksi kutsutaan kahden erityyppisen puolijohteen välistä kapeaa vyöhykettä. Esimerkiksi transistorissa on kaksi liitosta, mutta diodissa on vain yksi liitos. Lämmönsiirron näkökulmasta liitos on kohta, jossa tapahtuu lämpöä. Voidaan siis olettaa, että liitoksissa on suurin lämpö. Pii-pohjaisilla puolijohdekomponenteilla maksimaalinen käyttölämpötila ( $T_{\max}$ ) on 125 °C.  $T_{\max}$  arvoa pienempi toimintalämpötila on suositeltavaa, koska komponentti ei rasitu liikaa ja tämä takaa komponentille pidemmän iän.

Komponentin kotelo estää komponentin ydintä joutumasta suoraan kosketukseen ympäristön kanssa. Piisiru muodostaa komponentin ytimen. Kotelomateriaalina käytetään yleensä muovia, lasia tai keraamia.

Piisirua ei voida suoraan liittää kotelon pohjaan. Esimerkiksi muovin lämpölaajenemiskerroin on noin 20 kertaa suurempi kuin piin. Tähän tarvitaan erillinen kehys, jonka lämpölaajenemiskerroin on lähes sama kuin piillä, esimerkiksi kupari.

Ensimmäiseksi lasketaan systeemissä syntyvä häviöteho eli jäähdytyskuorma. Kuten termodynamiikan ensimmäinen pääsääntö sanoo, tuotavan energian on

oltava tasapainotilassa poistuvan energian kanssa . Tämän perusteella voidaan olettaa, että resistansseissa syntyvä lämpö on systeemin lähtevä teho. Laitteen häviöteho eli jäähdytyskuorma on yhtä suuri tehonkulutuksen kanssa. Lämpökuorma voidaan toteuttaa askel askeleelta, eli määritetään jokaisen systeemin komponentin tehohäviö ja laskea nämä yhteen.

Sähkölaite on termisessä tasapainotilassa eli laitteen lämpötila on sama kuin ympäristö, kun laite on poissa päältä. Kun laite käynnistetään, komponentit ja koko laitteen lämpötila alkaa nousta. Transienttivaiheessa lämpötila jatkaa nousuaan, kunnes päästään jatkuvuustilaan, eli häviöteho ja jäähdytysjärjestelmän pois kuljettava lämpö ovat yhtä suuria.

Lämpötilan vaihtelu voi aiheuttaa lämpöjännityksiä, jotka heikentävät laitteen toimivuutta. Kokeelliset analyysit ovat todistaneet, että mikäli lämpötila vaihtelee 20 °C, kasvaa sähkölaitteiden vikataajuus noin kahdeksankertaiseksi. Mitoituksessa täytyy ottaa huomioon lämpöjännityksestä johtuvat mekaaniset värähtelyt ja lämpöiskut, koska nämä ovat erittäin yleisiä vikojen aiheuttajia.

Maksimaalinen lämmönsiirtokyky ja suurin sallittu käyttölämpötila on ilmoitettu laite- ja komponenttivalmistajien datakirjoissa. Nämä arvot auttavat valitsemaan sopivan jäähdytysmenetelmän. Kun laitteen tai komponentin häviöteho tiedetään, voidaan lämpövuoto laskea jakamalla häviöteho laitteen tai komponentin lämmönsiirtopinnalla. Lämmön siirtomuoto saadaan määritettyä, kun otetaan huomioon vielä laitteen tai komponentin pinnan ja ympäristön välinen lämpötilaero.

Ajatus rakentaa automatisoitu jäähdytysjärjestelmä syntyi omasta aloitteestani. Miten voisin hyödyntää koulussa opittua tietoa ja yhdistää sitä sellaiseen, mitä koulussa ei ole tehty.

Ideana oli rakentaa jäähdytysjärjestelmä tietokoneisiin, mutta se soveltuu myös moniin muihin järjestelmiin. Vaikka tietokoneiden emolevyissä on paikat tuulettimille ja ne voivat myös ohjata tuulettimia, mutta tähän tarvitaan erillisiä ohjelmia, jotka näyttävät lämpötilan ja tuulettimen nopeuden. Tekemälläni

laitteella tuulettimien ohjaus toimii automaattisesti tai manuaalisesti ja lämpötila on nähtävissä ilman erillisiä ohjelmia reaaliajassa.

Mikrokontrolleri, mbed NXP LPC1768, vastaa järjestelmän toiminnallisuudesta. Kyseisessä kontrollerissa on ARM Cortex -M3 -ydin, jonka kellotaajuus on 90 MHz, ja 512 kB:n flash-muisti ja 64 kB:n RAM-muisti. Kontrollerista löytyy myös erilaisia liitäntöjä, kuten ethernet, CAN, SPI ja muita I/O-liitäntöjä.

Kontrollerin jälkeen oli valittava sopiva LCD-näyttö. Näytön tärkein valintakriteeri oli, että näyttö tukee Hitachin HD44780-standardia. HD44780:n avulla näytön ohjaaminen helpottuu, koska tähän on valmiita kirjastoja. Toinen kriteeri, joka vaikutti valintaan, oli näytön hinta. Näytön hinta ei saisi olla liian suuri, koska muuten laitetta ei voida tehdä sarjatuotantona.

Tuulettimesta muodosti ensimmäisen ongelma. Tuuletin tarvitsee 12 V toimiakseen täydellä teholla, mutta mbedin PWM-ohjauksen lähtö toimii välillä 0 V – 3,3 V, joten tästä voidaan päätellä, että ohjaus ei tule toimimaan suoraan. Mikrokontrollerin ja tuulettimen väliin tarvitaan MOSFET.

Järjestelmän testeissä ei havaittu mitään ongelmaa. PWM-tehonsäätö oli helppoa, koska PWM:a pystyttiin säätämään prosenttiluvuilla, mikä kuvasti tämän tehoa välillä 0,0 V – 3,3 V. Yhdessä testissä tuulettimen kierroksia vaihdettiin minuutin välein. Tätä testiä jatkui yhteensä viisi tuntia. Toisessa testissä järjestelmä jätettiin yöksi päälle, jossa tuuletin säädettiin 600 kierrokseen minuutissa. Järjestelmä selvisi näistä testeistä hyvin.

Piirilevysuunnittelussa käytin CadSoftin Eaglea. Eagle ( Easily Applicable Graphical Layout Editor) on varsin yleinen suunnitteluohjelma. Eaglesta löytyy kahta erillistä versiota, maksullinen ja maksuton.

Itse piirilevyn suunnittelussa täytyi ottaa enemmän asioita huomioon, kuten elektromagneettinen yhteensopivuus. Hyvään piirilevy suunnitteluun on monia erilaisia muistisääntöjä, joita voidaan noudattaa piirilevyn tekemisessä.

Mbed-järjestelmässä on täysi C++-tuki, eli ohjelmoinnin voi tehdä joko C:llä tai C++:lla. Koska mbedissä on online-kääntäjä, ei erillisiä ohjelmia tai kääntäjiä tarvita, vaan kaikki tapahtuu pilvessä.

## LÄHTEET

- Apex Microtechnology. 2013. PWM [viitattu 25.3.2013]. Saatavissa: <http://www.apexanalog.com/wp-content/uploads/2012/08/PwmBasics.pdf>
- Barr, M. 2013. Introduction to Pulse Width Modulation (PWM) [viitattu 25.3.2013]. Saatavissa: <http://www.barrgroup.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation>
- Beis, U. 2008. An Introduction to Delta Sigma Converters [viitattu 25.3.2013]. Saatavissa: <http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html>
- CML Microsystems Plc. 2012. Adaptive Delta Modulation for Short-range Digital Voice [viitattu 25.3.2013]. Saatavissa: <http://www.cmlmicro.com/Press/briefs/index.asp?/Press/briefs/adm.htm>
- Drakos, N. & Moore R. 2013. AC Ward Leonard drive systems [viitattu 25.3.2013]. Saatavissa: [http://zhongqc.staff.shef.ac.uk/ACDrive\\_web/ACDrive\\_web.html](http://zhongqc.staff.shef.ac.uk/ACDrive_web/ACDrive_web.html)
- Harvard.edu. 2013. Welcome to the Variable Autotransformer Page! [viitattu 25.3.2013]. Saatavissa: <http://www.nmr.mgh.harvard.edu/~reese/VariacPage/>
- Hirzel T. 2011. PWM [viitattu 25.3.2013]. Saatavissa: <http://arduino.cc/en/Tutorial/PWM>
- Huang, J. Padmanabhan. K. & Collins, O. M. 2011. The sampling theorem with constant amplitude variable width pulses. IEEE transactions on Circuits and Systems, vol. 58. [viitattu 25.3.2013]. Saatavissa: [http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5699377&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs\\_all.jsp%3Farnumber%3D5699377](http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5699377&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D5699377)
- Kazmierkowski, M. P., Krishnan, R. & Blaabjerg, F. 2013. Control in Power Electronics: Selected Problems [viitattu 25.3.2013] Saatavissa: [http://books.google.com/books?id=6\\_dmMHEyvrkC&pg=PA373&dq=%22space+vector+modulation%22+intitle:%22Control+in+Power+Electronics%22&lr=&as\\_brr=0&as\\_pt=ALLTYPES&ei=CBWOSdCVDJO2ygTvxuiXBg](http://books.google.com/books?id=6_dmMHEyvrkC&pg=PA373&dq=%22space+vector+modulation%22+intitle:%22Control+in+Power+Electronics%22&lr=&as_brr=0&as_pt=ALLTYPES&ei=CBWOSdCVDJO2ygTvxuiXBg)

Lappeenranta university of technology. 2013. Elektroniikkalaitteiden koostumus. [viitattu 25.3.2013]. Saatavissa:

[https://noppa.lut.fi/noppa/opintojakso/bl20a0100/luennot/luento\\_4.pdf](https://noppa.lut.fi/noppa/opintojakso/bl20a0100/luennot/luento_4.pdf)

Oc modshop. 2013. Zalman ZM-MFC2 Multi Fan Controller [viitattu 25.3.2013].

Saatavissa: <http://www.ocmodshop.com/zalman-zm-mfc2-multi-fan-controller/>

Peterchev, A. 2013. Digital Pulse-Width Modulation Control in Power Electronic Circuits: Theory and Applications [viitattu 25.3.2013]. Saatavissa:

[http://power.eecs.berkeley.edu/publications/theses/Peterchev\\_Dissertation\\_06-05.pdf](http://power.eecs.berkeley.edu/publications/theses/Peterchev_Dissertation_06-05.pdf)

Schönung, A. & Stemmler, H. 2010. Geregelter Drehstrom-Umkehrantrieb mit gesteuertem Umrichter nach dem Unterschwingungsverfahren. BBC Mitteilungen (Brown Boveri et Cie) 51 (8/9): 555–577.

Väyrynen, M. 2013. Moottorin ohjaus PWM:llä [viitattu 25.3.2013]. Saatavissa:

[http://www.hutasu.net/Mikrokontrollerit/MCU\\_launchpad\\_osa15.htm](http://www.hutasu.net/Mikrokontrollerit/MCU_launchpad_osa15.htm)

Wikipedia. 2013. Pulse-width modulation [viitattu 25.3.2013] Saatavissa:

[http://en.wikipedia.org/wiki/Pulse-width\\_modulation](http://en.wikipedia.org/wiki/Pulse-width_modulation)







### LIITE 3. Ohjelmakoodi

```
#include "mbed.h"
```

```
#include "TextLCD.h"
```

```
TextLCD lcd(p15, p16, p17, p18, p19, p20); // rs, e, d4-d7
```

```
DigitalOut myled1(LED1);
```

```
DigitalOut myled2(LED2);
```

```
DigitalOut myled3(LED3);
```

```
DigitalOut myled4(LED4);
```

```
DigitalIn kytkin(p20); // tässä kytkin
```

```
DigitalIn sensor1 (p10); //lämpöanturit
```

```
DigitalIn sensor2 (p11);
```

```
DigitalIn sensor3 (p12);
```

```
DigitalIn sensor4 (p13);
```

```
DigitalIn tacho1 (p25);
```

```
DigitalIn tacho2 (p26);
```

```
DigitalIn tacho3 (p27);
```

```
DigitalIn tacho4 (p28);
```

```
PwmOut fan1(p21); // tuulettimet
```

```
PwmOut fan2(p22);
```

```
PwmOut fan3(p23);
```

```
PwmOut fan4(p24);
```

```
void tuuletin_sensorit1(void)
```

```
{
```

```
int i;
```

```
int x;
```

```
float luku1=0;
```

```
int luku2=0;
```

```
float ka1;
```

```
int rpm;
```

```
float summa1=0;
```

```
int summa2=0;
```

```
for(i=0; i<100; i++) { // tässä keskiarvolaskuri
```

```
luku1 = sensor1 / 0.0033; // luku jaetaan jotta saadaan vastaamaan asteita (aste =  
10mv)
```

```
summa1 = summa1 + luku1;
```

```
wait(0.01); // odotetaan jotta saadaan häiriötä pois
```

```
}
```

```
ka1 = summa1 / 100; // tässä tärkein osa, että saadaan keskiarvo kunnolla, jaetaan  
100 koska i = 100
```

```
if(ka1>20) {
```

```
fan1=0.5;
```

```
}
```

```
if(ka1>35) {
```

```
fan1=0.6;
```

```
}
```

```
if (ka1>45) {
```

```
fan1=0.8;
```

```
}
```

```
if(ka1>60) {
```

```
fan1=1.0;
```

```
}
```

```
for(x=0; x<100; x++) { // uusi kerkiarvolaskuri, tällä kertaa kyseessä rpm
```

```
luku2 = tacho1 / 0.0005; // luku jaetaan jotta saadaan vastaamaan kierroksia (1rpm  
= 5mv)
```

```
summa2 = summa2 + luku2;
```

```
wait(0.01); // häiriön poisto
```

```
}
```

```
rpm = summa2 / 100; // tässä rpm
```

```
lcd.printf("Lampotila: %.2f C\n\r",ka1); // tässä taas näytetään lämpötila näytöllä
```

```
lcd.printf(" %i rpm",rpm); // tässä näkyy rpm näytöllä
```

```
}
```

```
void tuuletin_sensorit2(void)
```

```
{
```

```
int i;
```

```
int x;
```

```
float luku1=0;
```

```
int luku2=0;
```

```
float ka1;
```

```
int rpm;
```

```
float summa1=0;
```

```
int summa2=0;
```

```
for(i=0; i<100; i++) { // tässä keskiarvolaskuri
```

```
luku1 = sensor1 / 0.0033; // luku jaetaan jotta saadaan vastaamaan asteita (aste =  
10mv)
```

```
summa1 = summa1 + luku1;
```

```
wait(0.01); // odotetaan jotta saadaan häiriötä pois
```

```
}
```

```
ka1 = summa1 / 100; // tässä tärkein osa, että saadaan keskiarvo kunnolla, jaetaan  
100 koska i = 100
```

```
if(ka1>20) {
```

```
fan1=0.5;
```

```
}
```

```
if(ka1>35) {
```

```
fan1=0.6;
```

```
}
```

```
if (ka1>45) {
```

```
fan1=0.8;
```

```
}
```

```
if(ka1>60) {
```

```
fan1=1.0;
```

```
}
```

```
for(x=0; x<100; x++) { // uusi kerkiarvolaskuri, tällä kertaa kyseessä rpm
```

```
luku2 = tacho1 / 0.0005; // luku jaetaan jotta saadaan vastaamaan kierroksia (1rpm  
= 5mv)
```

```
summa2 = summa2 + luku2;
```

```
wait(0.01); // häiriön poisto
```

```
}
```

```
rpm = summa2 / 100; // tässä rpm
```

```
lcd.printf("Lampotila: %.2f C\n\r",ka1); // tässä taas näytetään lämpötila näytöllä
```

```
lcd.printf(" %i rpm",rpm); // tässä näkyy rpm näytöllä
```

```
}
```

```
void tuuletin_sensorit3(void)
```

```
{
```

```
int i;
```



```
int x;

float luku1=0;

int luku2=0;

float ka1;

int rpm;

float summa1=0;

int summa2=0;

for(i=0; i<100; i++) { // tässä keskiarvolaskuri

luku1 = sensor1 / 0.0033; // luku jaetaan jotta saadaan vastaamaan asteita (aste =
10mv)

summa1 = summa1 + luku1;

wait(0.01); // odotetaan jotta saadaan häiriötä pois

}

ka1 = summa1 / 100; // tässä tärkein osa, että saadaan keskiarvo kunnolla, jaetaan
100 koska i = 100

if(ka1>20) {

fan1=0.5;

}

if(ka1>35) {
```

```
fan1=0.6;
```

```
}
```

```
if (ka1>45) {
```

```
fan1=0.8;
```

```
}
```

```
if(ka1>60) {
```

```
fan1=1.0;
```

```
}
```

```
for(x=0; x<100; x++) { // uusi kerkiarvolaskuri, tällä kertaa kyseessä rpm
```

```
luku2 = tacho1 / 0.0005; // luku jaetaan jotta saadaan vastaamaan kierroksia (1rpm  
= 5mv)
```

```
summa2 = summa2 + luku2;
```

```
wait(0.01); // häiriön poisto
```

```
}
```

```
rpm = summa2 / 100; // tässä rpm
```

```
lcd.printf("Lampotila: %.2f C\n\r",ka1); // tässä taas näytetään lämpötila näytöllä
```

```
lcd.printf(" %i rpm",rpm); // tässä näkyy rpm näytöllä
```

```
}
```

```
void tuuletin_sensorit4(void)
```

```
{
```

```
int i;
```

```
int x;
```

```
float luku1=0;
```

```
int luku2=0;
```

```
float ka1;
```

```
int rpm;
```

```
float summa1=0;
```

```
int summa2=0;
```

```
for(i=0; i<100; i++) { // tässä keskiarvolaskuri
```

```
luku1 = sensor1 / 0.0033; // luku jaetaan jotta saadaan vastaamaan asteita (aste =  
10mv)
```

```
summa1 = summa1 + luku1;
```

```
wait(0.01); // odotetaan jotta saadaan häiriötä pois
```

```
}
```

```
ka1 = summa1 / 100; // tässä tärkein osa, että saadaan keskiarvo kunnolla, jaetaan  
100 koska i = 100
```

```
if(ka1>20) {
```

```
fan1=0.5;
```

```
}
```

```
if(ka1>35) {
```

```
fan1=0.6;
```

```
}
```

```
if (ka1>45) {
```

```
fan1=0.8;
```

```
}
```

```
if(ka1>60) {
```

```
fan1=1.0;
```

```
}
```

```
for(x=0; x<100; x++) { // uusi kerkiarvolaskuri, tällä kertaa kyseessä rpm
```

```
luku2 = tacho1 / 0.0005; // luku jaetaan jotta saadaan vastaamaan kierroksia (1rpm  
= 5mv)
```

```
summa2 = summa2 + luku2;
```

```
wait(0.01); // häiriön poisto
```

```
}
```

```
rpm = summa2 / 100; // tässä rpm
```

```
lcd.printf("Lampotila: %.2f C\n\r",ka1); // tässä taas näytetään lämpötila näytöllä
```

```
lcd.printf(" %i rpm",rpm); // tässä näkyy rpm näytöllä
```

```
}
```

```
int main()
```

```
{
```

```
int laskuri;
```

```
while(1) {
```

```
for(laskuri=1; laskuri<=4;) {
```

```
if(laskuri==1) {
```

```
tuuletin_sensorit1();
```

```
wait(0.5);
```

```
if(kytkin==1) {
```

```
laskuri++;
```

```
}  
  
}  
  
if(laskuri==2) {  
  
    tuuletin_sensorit2();  
  
    wait(0.5);  
  
    if(kytkin==1) {  
  
        laskuri++;  
  
    }  
  
}  
  
if(laskuri==3) {  
  
    tuuletin_sensorit2();  
  
    wait(0.5);  
  
    if(kytkin==1) {  
  
        laskuri++;  
  
    }  
  
}  
  
if(laskuri==4) {  
  
    tuuletin_sensorit2();  
  
    wait(0.5);  
  
    if(kytkin==1) {
```

```
laskuri++;
```

```
}
```

```
}
```

```
}
```

```
}
```

```
}
```