# Autonomt styrsystem för RGB-lysdioder med systemplattformen Arduino

Joachim Lindqvist

Examensarbete
Elektroteknik
2010

Joachim Lindqvist

Joachim Lindqvist

EXAMENSARBETE

Arcada

| | |
|---|---|
| Utbildningsprogram: | Elektroteknik |
| Identifikationsnummer: | 2919 |
| Författare: | Joachim Lindqvist |
| Arbetets namn: | Autonomt styrsystem för RGB-lysdioder med systemplattformen Arduino |
| Handledare (Arcada): | Rene Herrmann |
| Uppdragsgivare: | Arcada |

Sammandrag:

Detta arbete behandlar byggandet av en styrenhet för RGB-LEDar, dvs. lysdioder vars färg kan justeras genom att reglera tre parametrar; rött, grönt och blått. Styrenheten bygger på systemplattformen Arduino och programmeras genom ett enkelt program-varugränssnitt varpå den kan fungera självständigt. Detta är till nytta i applikationer där det är omöjligt att ha en dator konstant kopplad till styrenheten. Processen består av fyra delar; inmatning, överföring, lagring och användning av styrdata. I arbetet behandlas två styrmetoder: direkt styrning där styrenheten själv reglerar strömmen och indirekt styrning där data sänds ut till understyrenheter. I den direkta styrningen används pulsbreddsmodulation och i den indirekta styrningen DMX-protokollet. Programvarulösningar föredras framför externa komponenter i detta arbete eftersom meningen är att undersöka systemplattformens styrkor och svagheter. Till fråge-ställningarna hör: går det att bygga ett sådant system med en Arduino, hur långt kommer man utan hjälp av externa komponenter och är det något som inte går att genomföra ens med externa komponenter? Vissa funktioner, däribland DMX-styrning kunde inte implementeras eftersom konflikter uppstod i programkoden. I diskussionen presenteras dock förslag på hur dessa problem kunde lösas samt hur systemet kunde utvecklas ytterligare. Meningen är dessutom att ge läsaren en allmän bild av olika metoder för belysningsstyrning, även sådana som inte fysiskt implementeras i detta projekt.



| | |
|---|---|
| Nyckelord: | Arduino, automation, DMX, LED, RGB, PWM |
| Sidantal: | 67+20 |
| Språk: | svenska |
| Datum för godkännande: | 10.5 2010 |

| DEGREE THESIS | |
|---|---|
| Arcada | |
| | |
| Degree Programme: | Electrical Engineering |
| | |
| Identification number: | 2919 |
| Author: | Joachim Lindqvist |
| Title: | An independent control system for RGB-LEDs based on the system platform Arduino |
| Supervisor (Arcada): | Rene Herrmann |
| | |
| Commissioned by: | Arcada |
| | |

Abstract:

This thesis concerns the construction of a control unit for RGB-LEDs, i.e. light emitting diodes whose color can be chosen by adjusting three parameters; red, green and blue. It is built upon the system platform Arduino and programmed through a software interface after which it can function independently. This feature is useful in cases where it is impossible to have a computer permanently connected to the control unit. The control process can be divided into four parts; input, transmission, storing and usage of control data. Two control methods are used; direct control where the unit itself regulates the current and indirect control where data is sent out to sub control units. The direct control is achieved with pulse width modulation and the indirect with the DMX protocol. As one of the purposes is to examine the strengths and weaknesses of the system platform, software solutions are preferred whenever possible. Central questions include: is it possible to build a system like this on an Arduino, at which point is it necessary to add external components and whether some feature is impossible to create even with external components. Some features, including DMX could not be implemented as they resulted in conflicts in the program. Possible solutions to these issues as well as ideas for further development will be presented in the discussion part. An additional purpose of this thesis is to give the reader a general picture of various methods of lighting control including those not physically implemented in this project.



| Keywords: | Arduino, automation, DMX, LED, RGB, PWM |
|---|---|
| Number of pages: | 67+20 |
| Language: | Swedish |
| Date of acceptance: | 10.5 2010 |

| | |
|---|---|
| OPINNÄYTE | |
| Arcada | |
| | |
| Koulutusohjelma: | Sähkötekniikka |
| | |
| Tunnistenumero: | 2919 |
| Tekijä: | Joachim Lindqvist |
| Työn nimi: | Itsenäinen Arduino-alustaan pohjautuva ohjausjärjestelmä RGB-valodiodeille |
| Työn ohjaaja (Arcada): | Rene Herrmann |
| | |
| Toimeksiantaja: | Arcada |
| | |


Tiivistelmä:

Tämä opinnäytetyö käsittelee RGB-LEDien ohjauslaitteen rakennusvaiheita. RGB-LED on valodiodi jonka väri on säädettävissä kolmen parametrin; punaisen, vihreän ja sinisen avulla. Arduino-alustalle rakennettu laite ohjelmoidaan ohjelmistokäyttöliittymän avulla jonka jälkeen se voi toimia itsenäisenä. Tästä on hyötyä tilanteissa jossa on mahdotonta pitää ohjainlaite pysyvästi kytkettynä tietokoneeseen. Ohjausprosessi koostuu neljästä osasta; tietojen syötöstä, siirrosta, tallennuksesta ja käytöstä. Työssä käsitellään kahta ohjausmenetelmää: suora ohjaus jossa laite itse säätää valodiodeille syötettävän virran ja epäsuora ohjaus jossa dataa lähetetään valodiodeja ohjaaville aliohjausyksiköille. Suora ohjaus toteutetaan pulssileveysmodulaatiolla ja epäsuora ohjaus DMX-protokollalla. Työssä suositaan ohjelmistoratkaisuja koska tarkoituksena on kartoittaa alustan omat vahvuudet ja heikkoudet. Keskeisiin kysymyksiin kuuluu: onko tällaisen Arduino-pohjaisen ohjauslaitteen rakentaminen mahdollista, missä vaiheessa joudutaan turvautumaan ulkoisiin komponentteihin sekä onko joitakin toimintoja mahdotonta toteuttaa vaikka käytössä olisi myös ulkoisia komponentteja?. Kokeilussa ilmeni että joitakin toimintoja, mukaanlukien DMX-ohjausta ei voitu toteuttaa niiden ohjelmistossa synnyttämien ristiriitojen takia. Keskustelussa esitetään mahdollisia ratkaisuja näihin ongelmiin sekä ajatuksia siitä, miten järjestelmää voitaisiin kehittää edelleen. Tarkoitus on myös antaa lukijalle yleinen kuva erilaisista valonohjaustekniikoista, myös sellaisista joita ei tässä työssä ole sovellettu.



| | |
|---|---|
| Avainsanat: | Arduino, automaatio, DMX, LED, RGB, PWM |
| Sivumäärä: | 67+20 |
| Kieli: | ruotsi |
| Hyväksymispäivämäärä: | 10.5 2010 |

# INNEHÅLL

## Figurer

## Tabeller

# 1 INTRODUKTION

## 1.1 Bakgrund

Styrning av belysning är något som alla gör dagligen utan att tänka på det. Både inomhus och utomhus finns olika sorters belysningsarmaturer som slås på och av flera gånger per dag. Ett annat styrelement som ibland är installerat är dimmern som möjliggör inställning av ljusstyrkan hos lampan. Vid tillfällen då man är intresserad av att styra färgeffekter används olika slags automatiserade system. Dessa finns i många olika varianter riktade till olika behov; från system som är avsedda för att kontrollera några lysdioder i en stämningslampa till anläggningar bemannade av särskilda ljustekniker för att kontrollera tusentals lysdioder eller andra lampor vid en konsert eller liknande tillställning.

Lysdioden har redan länge varit en etablerad komponent främst som signallampa inom elektroniken. Då ljuseffekten hos denna har förbättrats har den fått ett bredare användningsområde. Styrning av konventionella belysningsarmaturer kräver vanligen kraftelektronik, medan lysdiodens relativt låga arbetsspänning gör att den kan styras med vanlig elektronik. RGB-lysdioder där färgen kan ändras genom spänningsvariation gör det möjligt att generera färgeffekter – vilket förutsätter ett styrsystem.

Då jag gjorde praktik på ett företag inom ljusreklamsbranschen fick jag bekanta mig med några RGB-lysdioder kontrollerade av en enkel mikrokontrollerbaserad krets som skickats till företaget som varuprov. Lysdioderna ändrade färg i en viss ordning och med en viss hastighet som inte kunde ändras. Det fanns möjlighet att med några extra delar ansluta kontrollern till ett DMX-nätverk varpå lysdioderna kunde styras. Jag funderade om någon på fullt allvar menade att man skall lägga ut tresiffriga eurobelopp på utrustning för att kontrollera tre spänningsnivåer upp till 12 V.  Fanns det inte ett enklare sätt? Detta såg ut som ett intressant ämne för ett examensarbete men vid en närmare titt såg det ut som

om designen av ett sådant system skulle vara ett alltför omfattande arbete, och idén begravdes för nästan ett halvt år. Då jag stötte på Arduino-projektet och såg att plattformen fanns till salu i Finland funderade jag över vad man skulle kunna göra med den för att testa dess egenskaper och detta problem verkade lämpligt. Arduino-projektet stötte jag för övrigt på då jag försökte hitta information om programvara för elektronikplanering och mikrokontrollerprogrammering på Macintosh-datorer. Det fanns nästan ingen saklig och strukturerad information om ämnet; den största delen av informationen fanns i små bitar på bloggar, diskussionssidor och personliga hemsidor. Den information som fanns tillgänglig kunde snarare klassificeras som rykten och jag kunde inte hitta ett enda examensarbete där detta ämnesområde ens skulle ha tagits upp. Därför kommer arbetet att innehålla en introduktion till detta ämne.

## 1.2 Mål och metod

Målet är att bygga en prototyp av en styrkrets som kan programmeras genom ett grafiskt användargränssnitt och därefter fungera självständigt dvs. utan datorstyrning så att den kan sättas in i applikationer där det inte är praktiskt eller möjligt att ha den permanent kopplad till en dator. Styrkretsen följer samma filosofi som plattformen den baserar sig på; användaren skall inte tvingas studera i timtal för att kunna göra något enkelt, medan det lätt skall gå att anpassa koden vid behov. Det innebär att programvarulösningar används då det går, vilket ger en större flexibilitet. De två huvudverktygen som används inom detta arbete, en dator med operativsystemet Mac OS X och Arduino-plattformen, är inte direkt förknippade med elektroteknik eller övriga ingenjörsvetenskaper utan snarare med media. Detta var ett medvetet val av två orsaker. Först och främst riktar den färdiga produkten sig just till personer verksamma inom detta område. Den andra orsaken är att det råder en brist på litteratur där dessa plattformer behandlas ur ett ingenjörsperspektiv. Meningen är att arbetet även skall fungera som en allmän introduktion för seriöst ingenjörsarbete på dessa plattformer.

## 1.3 Avgränsningar

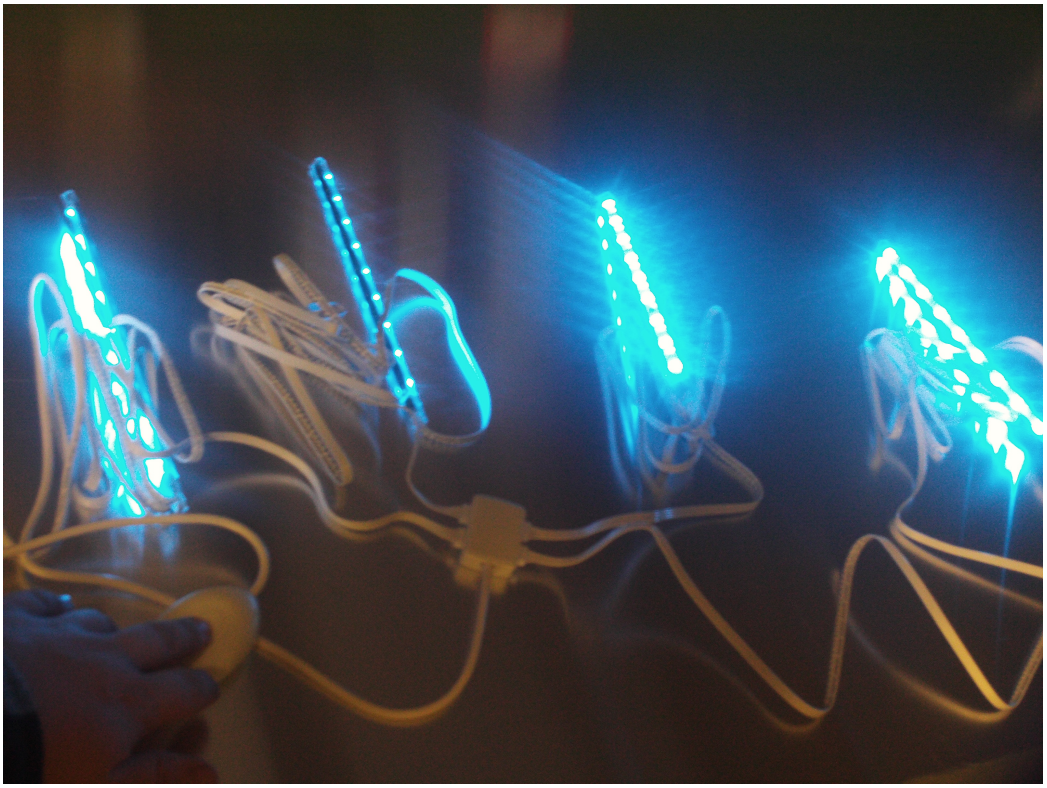
Då man bygger ett system av denna typ påminner arbetsprocessen om programvaruutveckling i allmänhet. Det är lätt att komma på nya egenskaper och detaljer som man vill implementera vilket leder till att arbetets omfattning växer i all oändlighet. I arbetet finns fyra centrala delar: dataöverföring mellan dator och styrenhet sparande och läsande av data i styrenhetens permanenta minne, behandling av data och slutligen styrning av lysdioder enligt de parametrar som användaren matat in. Då dessa delar fungerar kan arbetet anses färdigt. Även om programmeringsmiljön erbjuder möjligheten att använda mikrokontrollerns lågnivåfunktioner kommer dessa inte användas eftersom detta arbete strävar till att undersöka egenskaperna hos systemplattformen som sådan.

Arbetet begränsar sig till styrning av lysdioder dels för att de i allt större utsträckning håller på att ersätta glödlampor och andra belysningsarmaturer, dels för att de fungerar med spänningar av samma storleksklass som styrelektroniken. Färgstyrning av lysdioder kan genomföras direkt med en varierad spänning direkt från styrkretsen eller med nätverksstyrning där själva lysdioden styrs av nätverksanslutna mottagarkretsar. I detta arbete implementeras en vanlig teknik från vardera gruppen: pulsbreddsmodulation och DMX. DMX står för Digital Multiplexed och innebär att data skickas ut per kabel till mottagare som styr lysdioderna enligt de instruktioner de fått från huvudstyrenheten.

# 2 PROBLEMSTÄLLNING

## 2.1 Effekter för nybörjare

Som tidigare nämndes kan man grovt dela in styrsystem för lysdioder i enkla och avancerade. Problemet är att det finns en ganska bred klyfta mellan dessa två grupper.



*Figur 1. IKEA Dioder, en typisk enkel styrenhet.*

I sin enklaste form är det som går under beteckningen "LED-teknik" inte särskilt avancerat. Små byggsatser bestående av några dioder, en transformator med AC-DC likriktare (eftersom lysdiodena drivs med 12V likström) och installations-material har redan funnits på marknaden några år. Dessa finns även som RGB(färg)varianter. Ett typiskt exempel är en enhet som säljs i IKEA som går under namnet "Dioder" (se figur 1). Denna består av några lysdioder, en stickkontakt med integrerad inverter, en styrenhet där man per knapptryck kan byta färg eller byta mellan olika funktioner. Det finns tre funktioner; konstant

färg, snabbt färgbyte samt dimning genom hela färgspektret. Lysdioderna är kopplade till styrenheten genom en delare.

Byggsatsen levereras med installationsmaterial som skuvar och diverse plastbitar och en mycket stor vikt har lagts på att skapa en så elsäker produkt som möjligt. Användaren ges endast ett begränsat antal färger som visas i en viss ordning och inte ens hastigheten går att välja fritt. Eftersom alla lysdiodsstänger är kopplade till samma styrenhet lyser alla i samma färg.

På marknaden finns även elektroniska byggsatser för styrning av lysdioder där användaren kan reglera färgen manuellt. Denna typ av system brukar vanligen ändå endast kunna kontrollera en enda grupp dvs. alla lysdioder lyser i samma färg. Vissa tillverkare som Velleman erbjuder en fjärrkontrollerad lyxversion (Velleman nv 2010).

## 2.2 Professionella anläggningar och installationer

Yrkespersoner inom scenbranschen använder sig av automatiserade system för styrning av ett stort antal belysningsarmaturer. Dessa anläggningar är vanligen stora, stationära, dyra (ibland skräddarsydda för platsen) och opereras av en eller flera särskilt utbildade ljustekniker och elektriker. Ljusscener planeras, byggs upp och planeras. Hela processen går knappt ens att jämföra med belysningsstyrning i hemförhållanden. I denna typ av installationer handhar ljustyrningen även intensitet, rörelser och övriga effekter. (jfr. Viinamäki 2005)

Exempel på en professionell styranläggning är produkterna i Vector-serien som produceras av företaget Compulite (Compulite 2006). Computerlite Vector RED som bl.a. är utrustad med fem skärmar, kan kontrollera 16 DMX-universum dvs. 8192 enskilda lampor eller lysdioder, ha 200 olika program som löper samtidigt och nästan 10 000 olika makron och kölistor. Priset för denna anläggning är över 40 000 amerikanska dollar, därtill kommer kablar och DMX-kompatibla

belysningsarmaturer. Eftersom det tar tid att lära sig att använda denna typ av anläggningar riktar de sig enbart till yrkespersoner inom mediabranschen. Även utomhus finns komplicerade styrsystem. I taket på den täckta gågatan Fremont Street i Las Vegas, USA har man monterat ungefär 12,5 miljoner lysdioder (se figur 2) med hjälp av vilka man kan visa rörliga bilder i verkligt stor skala (Fremont Street Experience 2007). En tidigare installation bestod av ungefär två miljoner glödlampor som styrdes av 121 fullskaliga datorer (Simpson 2003:467).


*Figur 2. Takinstallation på Fremont Street i Las Vegas, USA – ett extremt exempel på belysningsstyrning.*

Även om styrsystemen är mer komplicerade skiljer sig inte själva lamporna särskilt mycket från de som används annanstans. Inom scenbranschen har man använt konventionella glöd- och halogenlampor men liksom inom annan slags belysning håller lysdioden på att ersätta dessa (Salzberg & Kupferman 2009-2010).

## 2.3 Mellanapplikationer

En användare som vill styra t.ex. 12 enskilda belysningsarmaturer, vilket ofta är fallet inom t.ex. inredningsarkitektur stöter snart på ett problem. De små byggsatserna räcker inte till medan stora anläggningar från konserthallar inte är praktiska att använda. De mellanapplikationer som finns är vanligen någon form av USB-DMX-gränssnitt; dvs. det är fråga om en kontrollenhet som sänder ut seriell data till mottagare kopplade till belysningsarmaturerna. Denna typ av system är ändå relativt dyra eftersom en DMX-kompatibel ledlampa kostar minst hundra euro. Dessutom är dessa system begränsade till ett visst antal förprogrammerade effekter och/eller kräver konstant koppling till en dator. Man kan säga att dessa produkter kombinerar de sämsta sidorna från de föregående kategorierna.

Generellt sett är det enda som behövs ett minne och något som kan producera tre styrsignaler per lysdiodsgrupp. Det går att designa ett eget mikrokontroller-baserat system från grunden – förutsatt att man har tillräckliga kunskaper i programmering och elektroteknik. Detta har inte alla, varför man många gånger bygger denna typ av system i team där artister och ingenjörer samarbetar. Ett sådant samarbete kan vara fördelaktigt om tekniken skall implementeras i en massproducerad produkt som discolampor eller om det är fråga om en större installation på en allmän plats. För en småskalig enskild installation skulle det bästa alternativet vara något som är lika lätt att använda som dessa enkla "IKEA-byggsatser", som fungerar bara genom att sätta stickkontakten i eluttaget men ändå har mer egenskaper.

Det ideala för dessa fall är en lösning där slutanvändaren bara kopplar in lysdioderna och matar in färgsekvenser genom ett användargränssnitt varefter dessa spelas upp. Systemet som skall kunna styra många enskilda lysdiodsgrupper samtidigt skall ge användaren så stor frihet som möjligt att bestämma färg- och tidsvariabler och det skall dessutom kunna fungera självständigt dvs. utan dator. Systemet kommer inte att fokusera på scen-

teknikens behov. Tyngdpunkten ligger istället på att skapa ett kompakt och enkelt system som är lätt att gömma undan. Den centrala frågeställningen är: går detta att bygga med plattformen Arduino eller inte?

# 3 TEKNISK BAKGRUND

## 3.1 Arbete med elektronik på Macintosh-datorer

Datorer spelar en central roll i arbete med elektroteknik. Inom området behöver man vanligen program för att planera och rita kretsar, utföra datorstyrda mätningar och simulationer, samt sammanställa resultaten i rapporter och diagram. Eftersom Macintosh-datorer har en relativt liten marknadsandel och en relativt liten andel av alla datoranvändare behöver denna typ av program existerar det endast ett fåtal sådana för Macintosh-operativsystemet. Operativsystemet associeras oftast med editering av grafik och media samt "vanligt" datorarbete.

Då det är fråga om planering och simulation finns det ett visst antal vanliga program som Eagle, LabView, SciLab och Spice tillgängliga som sådana. Programmen uppdateras ändå senare och inte lika ofta som för de andra operativsystemen. Eftersom Mac OS X baserar sig på operativsystemet UNIX går det att med vissa förändringar kompilera program som ursprungligen är skrivna för Linux för Macintosh (McKerrow 2007:2, Djajadiningrat 2004).

Vill man koppla ett yttre objekt som t.ex. en mikrokontroller till en Mac kommer man fort att stöta på problem. Dessa bör stöda USB eller alternativt Firewire, eftersom Mac aldrig använt sig av den typ av serieportar som Windows-datorer har, dessutom bör den ha en driver för Macintosh. Ett alternativ är att använda en serieportsadapter som även den kräver en driver.

På grund av dessa svårigheter finns det ingen orsak att föredra Macintosh framför de andra operativsystemen då det gäller arbete med elektronik. Då det gäller anslutning till perifer utrustning är det nästan ett arbete i sig att få konfigurationen att fungera överhuvudtaget. Är man intresserad av att verkligen göra någonting och inte bara att experimentera med konfigurationen lönar det sig att programmera på en högre nivå. Mac stöder plattformer som Basic Stamp

och den Java-baserade TINI (McKerrow 2007:7-9). Även dessa kopplas med hjälp av en seriekabel och kräver därför en serieadapter.

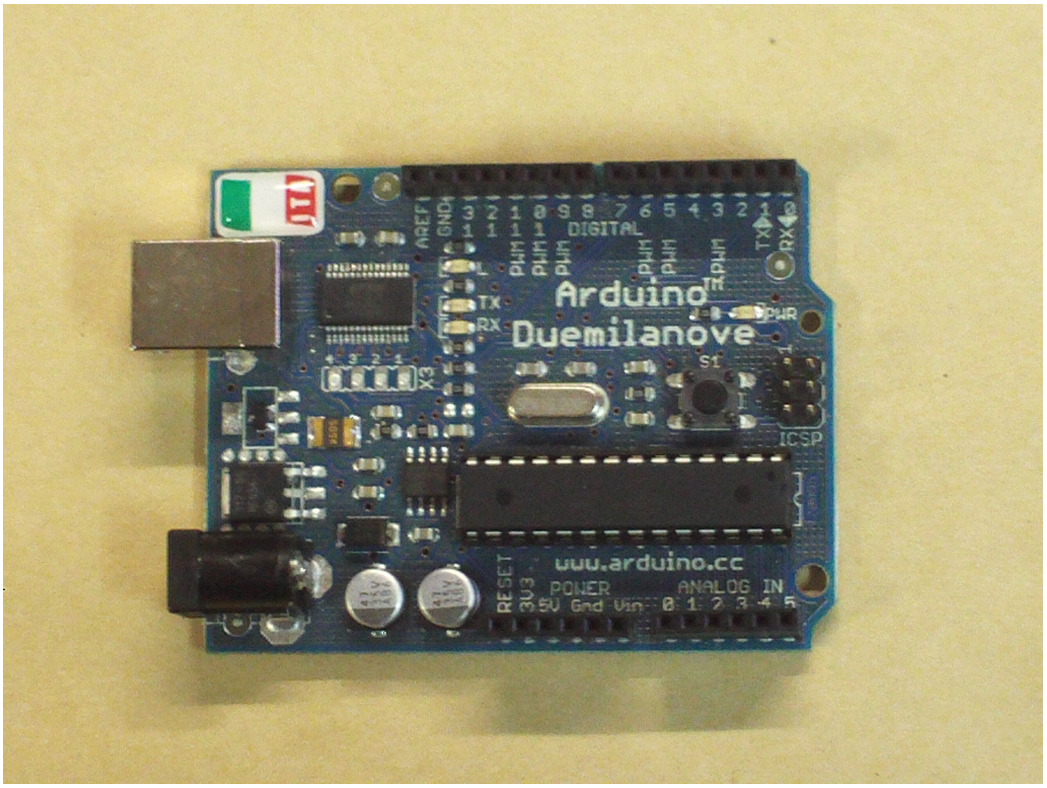
Om man vill ha en enkel "plug-and-play"-lösning är alternativet någon av plattformerna i Arduino-familjen. Dessa kopplas till datorn med en USB-kabel och kräver endast ett program: användargränssnittet.

## 3.2 Arduino

### 3.2.1 Arduino Duemilanove

Arduino presenteras på hemsidan som en öppen försöksplattform för elektronik (Arduino 2010a). En Arduino består av en mikrokontroller tillverkad av AVR Atmel som är förprogrammerad med en sk. bootloader som tar hand om lågnivåuppgifter samt bl.a. en yttre oscillator och en USB-port. Detta gör att programmeringen påminner mer om PC-programmering än mikrokontroller-programmering. Då det är fråga om en öppen design har det utvecklats många olika "Arduino-kompatibla plattformer" vars namn ofta slutar med "-duino". En del av dem baserar sig på PICs eller ARMs mikrokontrollrar (jfr. Wikipedia 2010a).

Eftersom det dessutom finns totalt elva Arduino-plattformer som har olika egenskaper och som styrs av olika mikrokontrollrar men ändå programmeras med samma gränssnitt är det felaktigt att, så som Gibb (2010) gjort, kalla Arduino "mikrokontroller"! Arduino är en helhet som består av både programvara och hårdvara. Som rubriken anger, kommer detta arbete att koncentrera sig på vad man kan göra med Arduino-plattformen, inte med mikrokontrollern ATmega328. Även om programmeringsmiljön stöder C-kommandon som inte hör till Arduinos dokumentation och mikrokontrollerns assemblerkod kommer dessa inte att användas.




*Figur 3. Arduino Duemilanove, systemplattformen som användes i arbetet.*

Plattformen som används inom detta arbete är Arduino Duemilanove (se figur 3). Modellnamnet betyder tvåtusennio på italienska och betecknar året för publiceringen av denna modell. Arduino utvecklades först vid Interaction Design Institute Ivrea i staden Ivrea. Namnet Arduino kommer från Arduino av Ivrea som utropade sig kung av Italien år 1002 (Clarke 2009, Wikipedia 2010b). Duemilanove är en grundmodell som räcker till de flesta applikationer. De andra plattformerna används om man behöver mer minne, fler in- och utgångar eller en fysiskt mindre styrkrets. Egenskaperna är sammanfattade i nedanstående tabell.

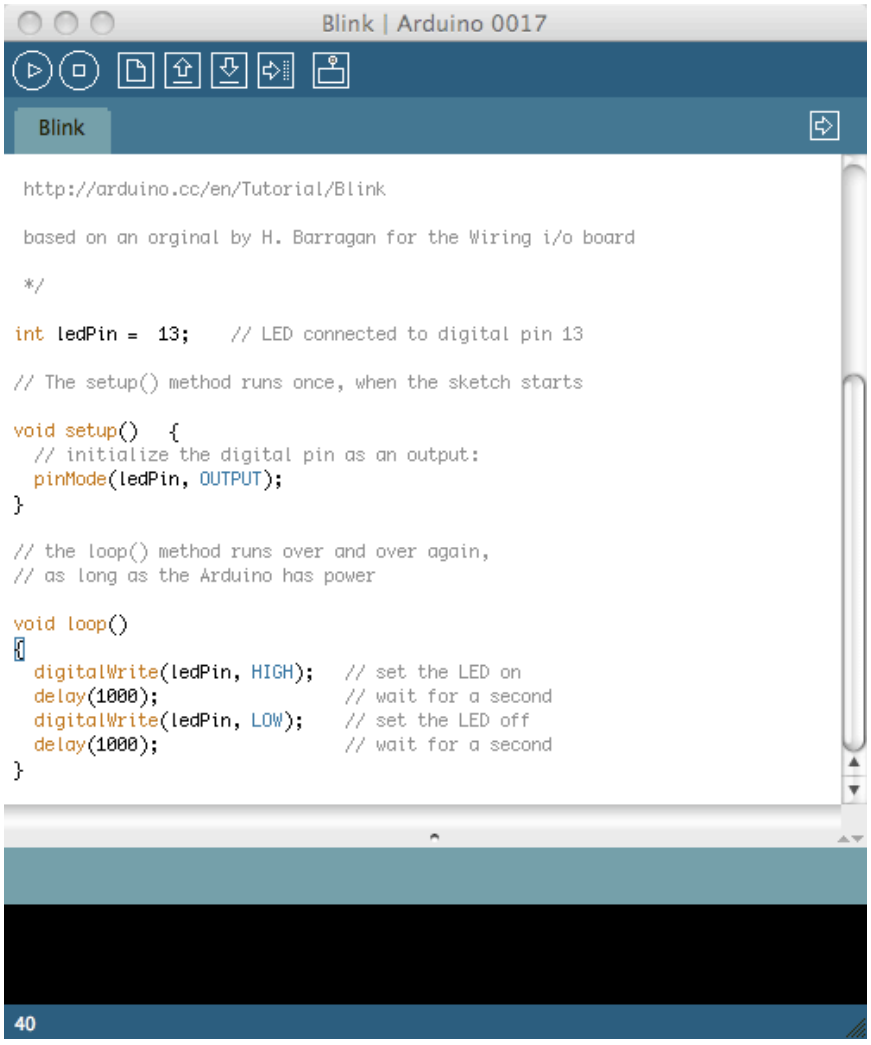
*Tabell 1: Tekniska specifikationer för Arduino Duemilanove (Arduino 2010b).*

| | |
|---|---|
| Mikrokontroller | ATmega328 |
| Arbetsspänning | 5 V |
| Rekommenderad inspänning | 7-12 V |
| Tillåten inspänning | 6-20 V |
| Digitala in- och utgångar | 14 st (6 st PWM-utgångar) |
| Analoga ingångar | 6 st |
| Ström igenom in- och utgångarna | 40 mA |
| Ström igenom 3,3 V - utgången | 50 mA |
| Flash-minne | 32 KB |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Klockfrekvens | 16 MHz |

### 3.2.2 Användning i praktiken

Jämfört med annan programmerbar elektronik är Arduino mycket enkel att använda. Kretsen kopplas till datorn med en USB-kabel, programmeringen sköts med ett enda gränssnitt som också innehåller en terminal med vilken man kan skicka och ta emot data från Arduinon. Arduino utvecklades för personer som inte nödvändigtvis har ingenjörsbakgrund; personer som på ett enkelt sätt vill uppfinna eller skapa enkla saker där man har nytta av en mikrokontroller (Clarke 2009, Gibb 2010). Även om Arduino inte är bra för signalbehandling, frekvenssyntes och andra tidskritiska tillämpningar kan den med framgång användas då man vill bygga t.ex. ett programmerbart styrsystem, en robot eller ett mätinstrument.

Plattformen baserar sig på programmeringsspråket Processing och användar-gränssnittet ser likadant ut (jfr figur 4, figur 5). Båda följer samma princip; tröskeln för nybörjare skall vara låg, medan det inte skall finnas något "tak". För Arduinos del betyder detta att man inte behöver skriva många rader kod för att t.ex. tända och släcka en lysdiod, medan man med tid och tålamod kan bygga i princip hur komplicerade saker som helst. Tom Igoe, en av utvecklarna beskriver det som en "glaslåda": "Arduino embodies what I call “glass box encapsulation”. That means that you don't have to look at the lower level code that comprises the libraries if you don't want to, but you can if you choose". (Clarke 2009)

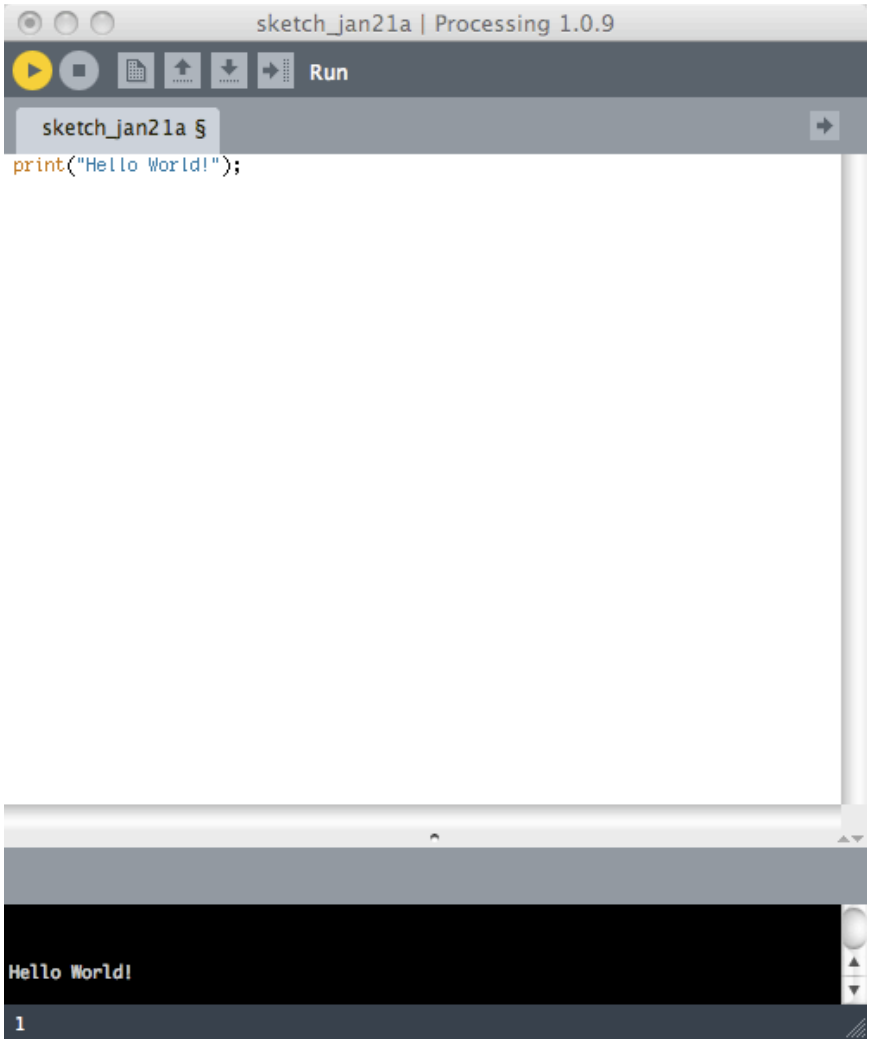



*Figur 4. Arduinos användargränssnitt med ett exempelprogram för en blinkande lysdiod.*

Då Arduinon som sådan inte räcker till måste man utöka den med extra komponenter som t.ex. sensorer, IC-kretsar, lampor, högtalare eller uttag. Ett praktiskt sätt är att placera kretsen med de extra komponenterna ovanpå Arduino-kretsen. Arduino Duemilanove levereras med en serie pinnar som man kan löda fast i ett kretskort. Efter det kan man trycka fast kretsen på Arduinon på samma vis som ett DIP-förpackat chip trycks fast i en sockel på ett kretskort. Kretsen sitter stadigt och inga lösa ledningar behövs. Denna konstruktion kallas "shield", eftersom den ser ut som en sköld som skyddar Arduinon. Mindre modeller som Arduino LilyPad är designade för att lödas fast som sådana på ett kretskort.

### 3.2.3 Arduino tillsammans med annan programvara

Som tidigare nämnts innefattar Arduinos användargränssnitt en terminal med vilken plattformen kan kommunicera med datorn som den är kopplad till. Denna terminal är användbar för testning, men syftet är att utveckla ett användar-gränssnitt där användaren inte tvingas skriva in värden enskilt. Det som behövs är alltså ett användarvänligt program som direkt kan kommunicera med Arduinon.




*Figur 5. Programmeringsmiljön Processing med exempelkod för "Hello World" - programmet.*

Ett vanligt val för detta är programmeringsspråket och -miljön Processing (se figur 5). Språket utvecklades först vid medialaboratoriet vid Massachusetts Institute of Technology och senare som ett enskilt projekt under ledning av Ben Fry och Casey Reas (Processing 2010). Processing utvecklades ursprungligen som ett medietekniskt verktyg men i praktiken går det att lösa vilka programmeringsuppgifter som helst med det. I detta fall innebär det att användaren skall mata in information om färg, tid och då det är fråga om armaturer som är kopplade i ett nätverk, adress. Programmets uppgift är att överföra informationen till Arduino-plattformen i en kompakt form. Processing kan skapa fullt fungerande fristående program för både Macintosh, Windows och Linux. Detta innebär att användaren inte egentligen ser eller behöver känna

till något av det som händer "under ytan", och att oerfarna användare inte av misstag kommer åt att modifiera koden.

### 3.2.4 Minnestyper

Arduino Duemilanove använder sig av tre minnestyper; Flash, SRAM och EEPROM. Den förstnämnda är störst och används om man har en stor mängd data som programmet skall använda under körning. I ett exempel som följer med ett programbibliotek för snabb åtkomst av Flash-minnet sparas stora mängder text och värden i detta minne som sedan printas ut. Det största problemet är att detta minne inte kan skrivas i då programmet körs, därmed har man inte direkt någon nytta av det i denna applikation. SRAM, i praktiken programminnet, används för att spara variabler då processen körs. Det är fråga om ett flyktigt minne vilket betyder att all data försvinner då strömtillförseln till kretsen bryts.

Det till storleken minsta minnet, EEPROM, är det enda som ger möjligheten att spara värden som kommer in genom en serielänk och att använda dessa efter att strömmen har brutits. Storleken på minnet som erbjuds är 1 KB, dvs. 1024 byte, som räcker till för detta projekt även om det kunde ha varit större. Behövs ett större minne kan man koppla en extern EEPROM till Arduinon varpå dessa kommunicerar med varandra med Serial Peripheral Interface (SPI) eller i2c-protokollet. För extremt stora minnesbehov har det utvecklats en shield för Secure Digital-kort, dvs. de rektangulära korten som används i de flesta digitalkameror och som vanligen har en kapacitet på flera gigabyte.

Slutligen bör nämnas en begränsning som främst berör applikationer där ny data ofta skrivs till mikrokontrollerns permanenta minnen; det finns en övre gräns för hur många gånger det är möjligt. Flash-minnet, där även de program man skriver hamnar kan skrivas 10 000 gånger och varje minnesplats i EEPROM kan skrivas över 100 000 gånger. (Atmel 2009:1)

## 3.3 Lysdioden

### 3.3.1 Allmänna egenskaper

Lysdioden är en halvledardiod som emitterar ljus då ström leds från anod till katod. Lysdioden har vissa egenskaper som skiljer den från vanliga glödlampor; eftersom den är en slags diod beter den sig också som en sådan. Då en liten spänning sätts över en lysdiod leder den ingen ström, men vid ett visst tröskelvärde börjar den leda. Ökar man ytterligare på spänningen kommer man till en punkt då den leder nästan oändligt mycket ström. Då man vill koppla en enskild lysdiod till en spänningskälla bör man ha ett skydds-motstånd seriekopplat med lysdioden för att förhindra att den förstörs av en alltför stor ström.

Enligt Haug (2010) skall en matningskrets för lysdioder vara just en strömreglerare. Då strömmen som en funktion av spänningen visas i ett diagram för lysdioder av många tillverkare finns samma beteende hos samtliga; de stiger långsamt till och med en viss punkt varefter strömmen stiger nästan lodrätt. Problemet är att denna punkt ligger vid något olika spänning för var och en typ, och vid ett experiment utfört av Haug påvisades strömmen vid en viss spänning vara sju gånger större hos en lysdiodstyp jämfört med en annan. Då däremot strömmen regleras till en konstant nivå finns det en liten skillnad i spänningen över de olika lysdioderna men den är så liten att den inte i praktiken märks. (Haug 2010)

Då en elektrisk ström leds genom lysdioden reagerar de laddningsbärande partiklarna med varandra och sänder ut fotoner, dvs. synligt ljus. Detta ljus har en smalare bandbredd än andra ljuskällor vilket betyder att det finns liten variation i ljusets våglängd. Våglängden dvs. ljusets färg beror på vilka material lysdioden är uppbyggd av. Förutom grundmaterialet finns det även små mängder orenheter som är avsiktligt tillsatta för att skapa sk. aktiva skikt, som ger upphov

till att fotoner dvs. ljus emitteras. (Jeppson 1984:141-142, Siemens Aktiengesellschaft 1990:93)

*Tabell 2: Sammansättning av och egenskaper hos några vanliga lysdioder (Siemens Aktiengesellschaft 1990:94).*

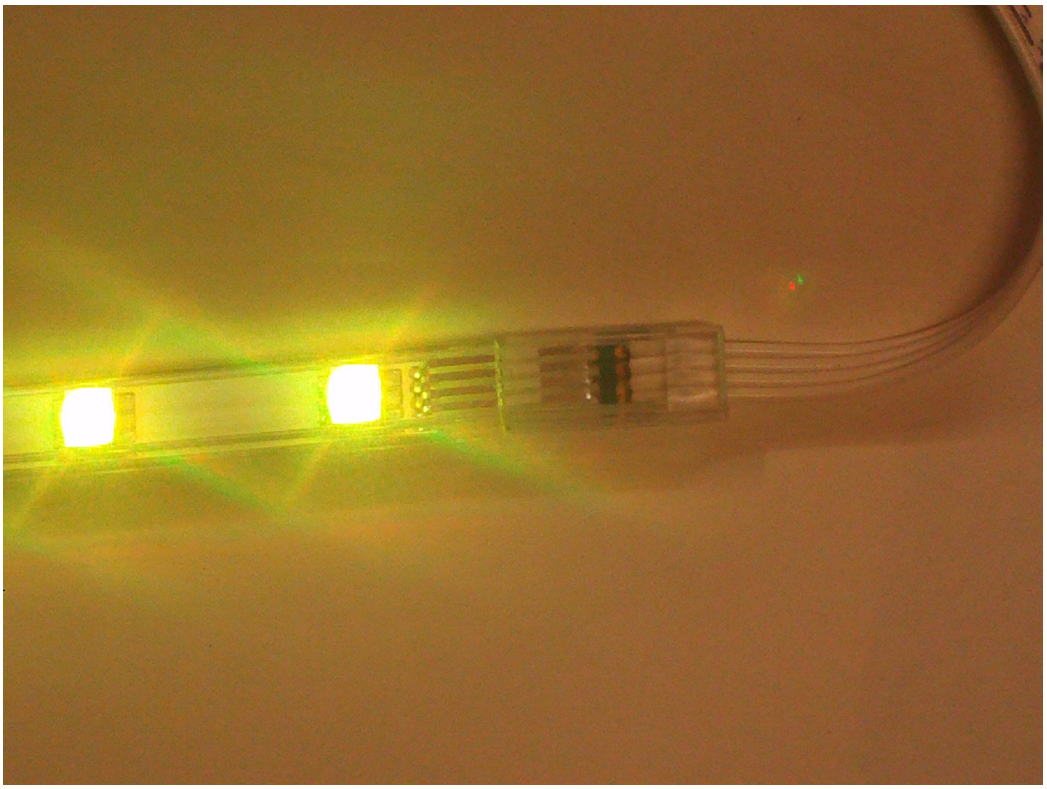
| Färg | Våglängd (nm) | Material | Bandbredd (eV) | Aktivt skikt |
|---|---|---|---|---|
| Infraröd | 950 | GaAs | 14 | GaAs:Si |
| Infraröd | 800-900 | GaAs | 14 | GaAlAs |
| Röd | 700 | GaP | 23 | GaPZn |
| Standardröd | 660 | GaAs | 14 | $GaAs_{0,6}P_{0,4}$ |
| Superröd | 635 | GaP | 23 | $GaAs_{0.35}P_{0,65}$: N (TSN) |
| Gul | 590 | GaP | 23 | $GaAs_{0.15}P_{0,85}$: N (TSN) |
| Grön | 565 | GaP | 23 | GaP:N |
| Blå | 480 | SiC | 28 | SiC |

Komponenten skiljer sig från konventionella belysningsarmaturer som t.ex. glödlampor även genom sin lägre driftsspänning och lägre energiförbrukning. Detta har gjort komponenten till ett alternativ i belysningssammanhang då man vill spara energi. Då lysdioder används som alstrare av belysning använder man inte enskilda dioder utan diodlampor eller plattor som består av flera dioder. I dessa finns vanligen skyddsmotståndet inkluderat i chipet, så denna typ av lysdioder kan anslutas direkt till spänningskällan som om den vore en glödlampa. Lysdioder där färgen kan styras är oftast av denna typ.

### 3.3.2 RGB-lysdioden

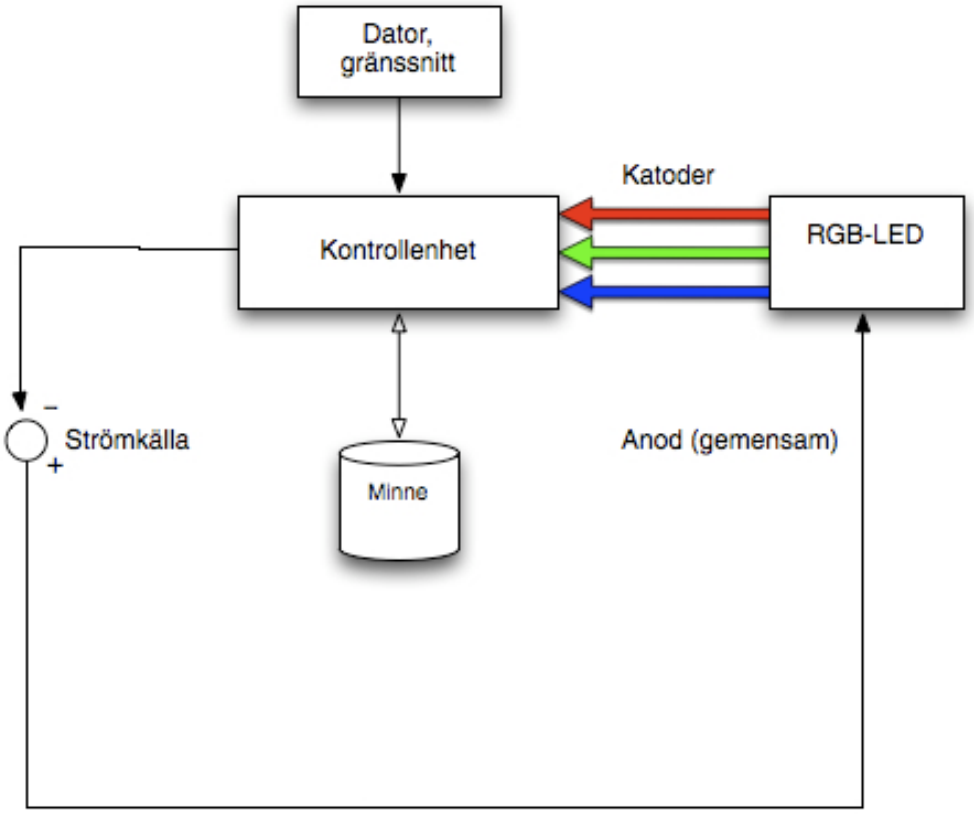
Lysdioden är inte huvudsakligen en alstrare av belysning. Ett större användningsområde är signallampor på elektronisk utrustning, ljusreklamer och

inredningsarmaturer där komponentens huvudsakliga uppgift inte är att producera vitt läsljus. I denna typ av applikationer är man ibland intresserad av att byta färg. Detta kan genomföras med enfärgade lysdioder som kopplas på och av eller med RGB-lysdioder. RGB-lysdioder kan producera ljus av nästan vilken färg som helst.



*Figur 6. Matningen för en RGB-lysdiod – en anod och tre katoder.*

RGB är en färgmodell som används förutom att definiera färg hos lysdioder också till att bl.a. definiera färg i datorer. Färger delas upp i tre komponenter, rött grönt och blått, och varje komponent antar i datorn ett värde mellan noll och 255 (eller då talen representeras med hexadecimaler, FF). Detta innebär att man genom att kombinera dessa färgkomponenter kan producera över 16 miljoner olika färger. RGB är en additiv färgmodell, vilket innebär att man börjar från svart (inget ljus), och närmar sig vitt då man blandar in mer och mer av färgkomponenterna. Vitt ljus är även i verkligheten en blandning av alla våglängders ljus.




*Figur 7. Blockschema över direkt färgreglering av en RGB-lysdiod.*

Lysdioder där färgen kan varieras är vanligen av "gemensam anod"-typ, dvs. dioden har en anod och tre katoder, en för varje färgkomponent. Genom att variera strömmen genom katoderna reglerar man färgkomponenterna och därmed lysdiodens färg (se figur 6, figur 7). Vitt representeras av att alla komponenters värde är helt på (255,255,255) medan svart representeras av att alla komponenter är ställda på noll. Kombinering av rött, grönt och blått ljus är för övrigt en metod för att skapa vitt led-ljus, som dock ställer vissa krav på styrningen eftersom de olika komponenterna har olika känslighet för ström och temperatur (Subramanian et al. 2002).

### 3.3.3 Användningsområden för RGB-lysdioder

RGB-lysdioder används där många olika färger skall fås ur samma lysdioder, vilket i praktiken alltid involverar någon typ av styrsystem. Liksom vanliga lysdioder används RGB-lysdioder både inomhus och utomhus. Användningsområden sträcker sig från restauranger, barer och heminredning till installationer vid tillställningar, reklamskyltar och upplysta fasader. Vanligen önskar man gömma undan styrsystemet, eftersom svarta lådor och kablar inte är det som man vill visa åskådaren. Är det fråga om en installation som befinner sig utomhus vill man ofta ha styrsystemet någon annanstans där det inte kan påverkas av temperaturförändningar, fukt och liknande. Utomhusinstallationer är för övrigt vanligare i USA än i Europa eftersom reglerna är mer strikta här (Simpson 2003:465). I Finland regleras utomhusinstallationer av ordningslagens 6§ 1 mom (Finlex 2010).

## 3.4 Styrning av lysdioder

Ljusintensiteten hos en lysdiod varieras genom att variera strömmen. Den enklaste metoden är att använda ett vridbart motstånd. Tekniskt sett är den enda skillnaden mellan dimning och färgstyrning att den senare innebär dimning av många färger samtidigt, vilket innebär att begreppen kan användas synonymt (Howell 2002:1). Man måste ändå koppla ett skyddsmotstånd i serie med dessa eftersom båda kan förstöras av en alltför stor ström.

I sammanhanget kan nämnas att förändring av intensiteten hos RGB-lysdioder inte är den enda metoden att skapa färgeffekter med lysdioder. Man kan även ha lysdioder av olika färg placerade nära varandra och tända och släcka dessa. I en sådan installation krävs endast binär styrning dvs. lysdioderna kan vara antingen helt på eller helt avstängda. En sådan installation kunde produceras med mycket kortare kod, och eftersom alla digitala uttag kan användas kan man styra många

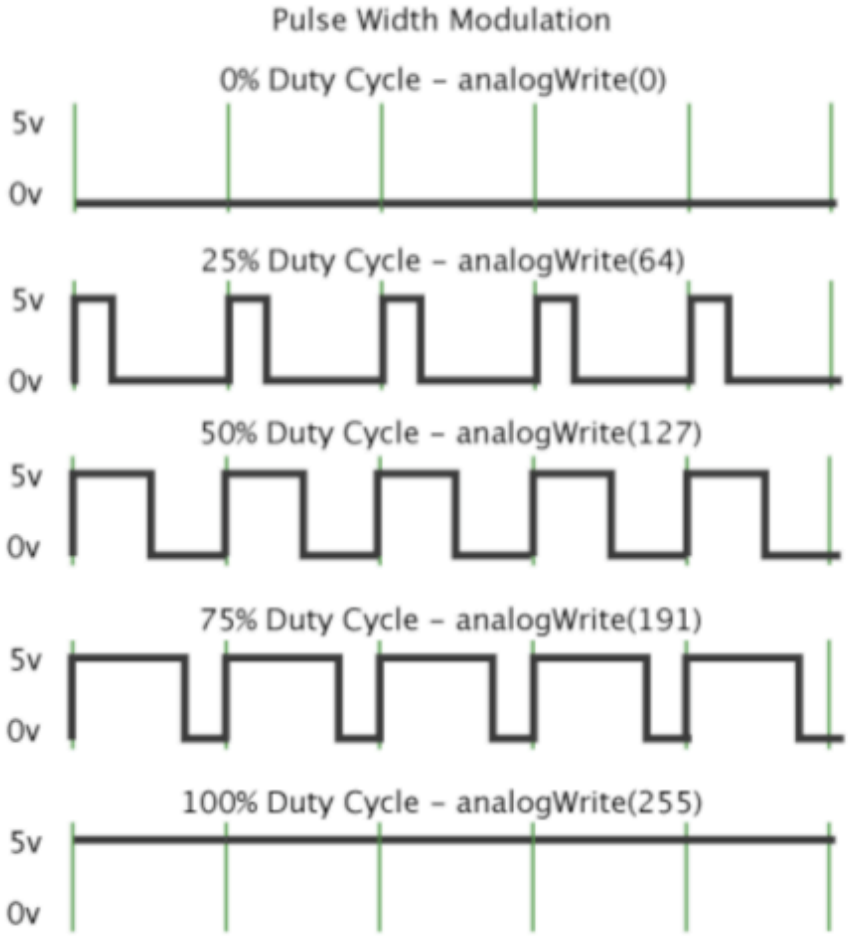
fler lysdiodsgrupper. Till nackdelarna hör att färgurvalet är begränsat till de lysdioder man har och att ljusstyrkan endast kan vara 100% eller 0%.

Styrmetoderna för lysdioder har här delats upp i två huvudgrupper: direkt och indirekt styrning. Direkt styrning innebär att det reglerande elementet seriekopplas med lysdioden och strömkällan medan indirekt innebär att data skickas ut till externa styrelement som i sin tur styr lysdioderna.

### 3.4.1 PWM

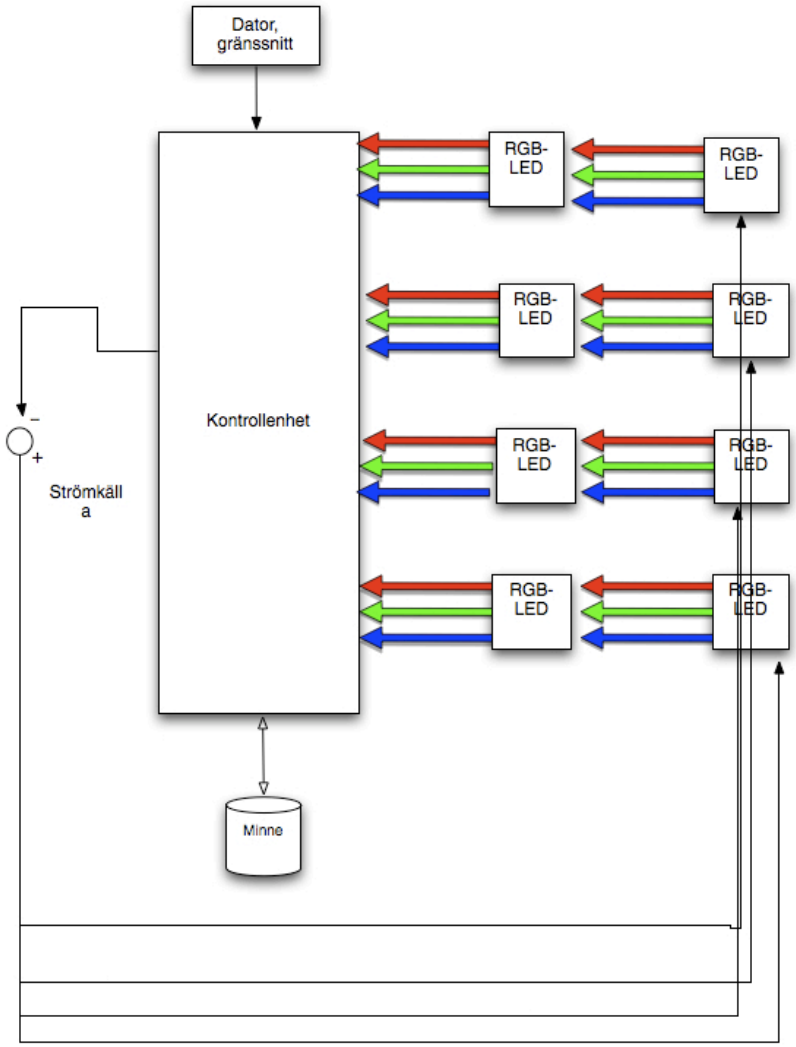
När man vill styra spänningen direkt med en mikrokontroller är en vanlig metod PWM, pulsbreddsmodulation. PWM innebär att mikrokontrollern sänder ut en digital signal dvs. en signal med två möjliga nivåer; antingen ett eller noll. En PWM-signal har en viss frekvens och därmed även en viss periodtid under vilken signalen kan vara på hela tiden, en del av tiden eller inte alls. Då frekvensen är tillräckligt hög skapas en analog signal, vars amplitud kan varieras genom att variera hur stor del av en period signalen är på (se figur 8). Om signalen inte alls är på får man naturligtvis en utsignal med amplituden noll volt. Är signalen på hela tiden får man den amplitud som den digitala signalen har. Om signalen är på hälften av tiden är utsignalen hälften av den digitala signalens amplitud, är den på en tredjedel av tiden är utsignalen en tredjedel av den digitala signalen osv. Den procentuella delen av tiden som signalen är på kallas duty cycle. (Barr 2001)

Genom att variera på signalens duty cycle kan man justera ljusstyrkan hos en lysdiod. Då man varierar duty cyclen på de tre kanalerna till en RGB-lysdiod kan man justera dess färg. I kommersiella installationer använder man lysdiods-plattor som arbetar med en högre spänning än Arduino klarar av att leverera, därför måste signalen förstärkas med transistorer innan de kopplas till lysdioder.




*Figur 8. PWM för olika duty cycler samt Arduino-kommandot för dessa (Arduino 2010c).*

Gemensamt för alla direkta styrmetoder är att man har en eller flera grupper av lysdioder (vanliga eller RGB) som är kopplade direkt i serie med strömkällan och lysdioderna. Styrenheten reglerar direkt mängden ström som passerar i kanalerna och därmed även lysdiodens intensitet (vanliga lysdioder) eller färg (RGB). Även om det finns många olika sätt att producera denna styrning är slutresultatet som man kan uppmäta med en multimeter detsamma som om man ersätter styrenheten med vridmotstånd och ställer in färgen med den. Har man fler grupper av lysdioder är den enda skillnaden att man har fler ingångar för fler katoder såsom framgår ur figur 9.



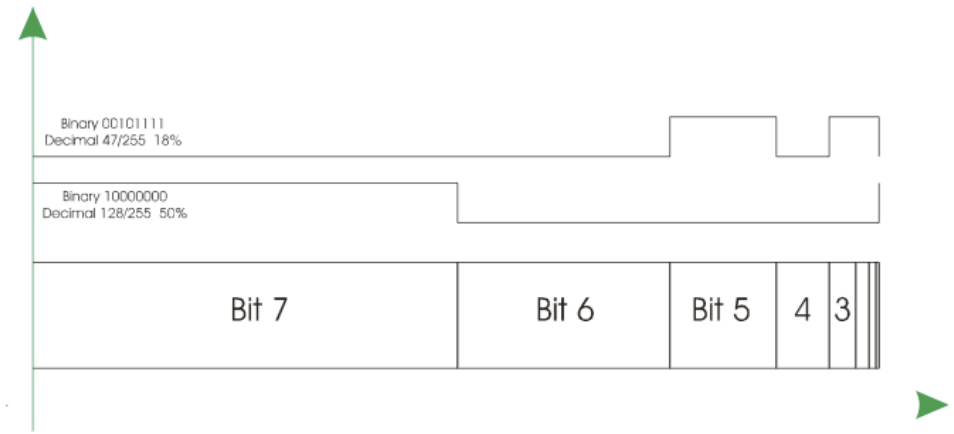

*Figur 9. Allmänt blockschema över direkt styrning av RGB-lysdioder.*

### 3.4.2 Bit Angle Modulation och andra modulationsmetoder

PWM är inte den enda direkta modulationsmetoden som används. Lysdioder kan även styras med hjälp av en växelströmssignal som slås av och på, genom frekvensmodulation eller med en metod som kallas Bit Angle Modulation, förkortat BAM.

BAM är en signal som påminner om en PWM-signal eftersom den är på eller av under en viss del av en tidperiod. BAM-signalen är indelad i åtta intervall av olika längd, det första intervallets längd är hälften av en tidsperiod, den andra en

fjärdedel av en tidsperiod osv. Signalen kan liknas vid en digital byte där det första intervallet representerar Most Significant Bit och det sista Least Significant Bit. Denna digitala byte kan representera totalt 256 olika amplitud-nivåer mellan fullständigt på (11111111) och inte alls på (00000000). Då man använder BAM kontrollerar mikrokontrollern åtta gånger per period om signalen skall vara på eller av till skillnad från 256 gånger vid PWM. (se figur 10, Howell 2002:8-9)




*Figur 10. En period av en BAM-signal (Howell 2002:8).*

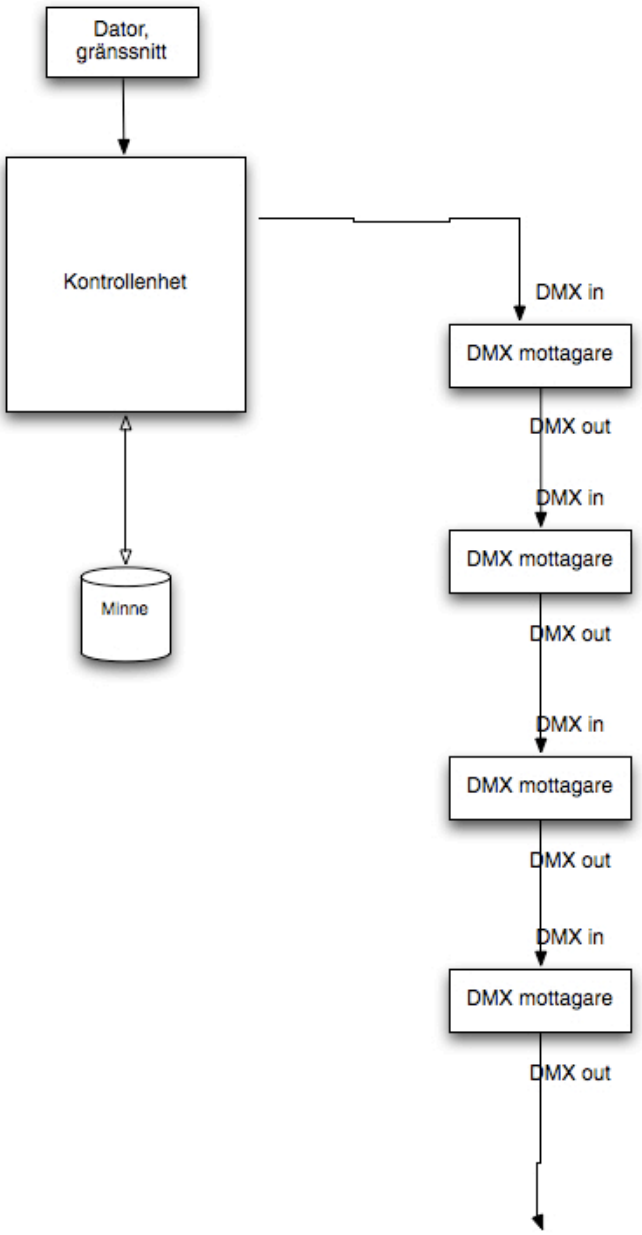
En annan metod som används är frekvensmodulation som även den är en variant av PWM. Man använder lika stora pulser men avståndet mellan pulserna, dvs. pulsernas frekvens varieras. Ljusintensiteten är därmed omvänt proportionell mot avståndet mellan pulserna.

Ytterligare en modulationsmetod som existerar är Lehmann-modulation där en period är indelad i mindre segment. Modulationsmetoden är patenterad, vilket betyder att den inte kan användas i öppen design. (Sukale 2008:77-78)

### 3.4.3 DMX

Ovanstående modulationsmetoder fungerar bra då man har ett förhållandevis litet antal lysdioder eller -grupper som skall styras skilt (jfr. figur 9). Om man har 100 lysdioder som alla skall ha samma ljusstyrka eller färg är det enkelt att seriekoppla dessa och styra dem med ett vridmotstånd eller en mikrokontroller. Om dessa däremot skall styras enskilt krävs minst 101 kablar, en gemensam plus- och en egen minuskabel för varje lysdiod. Är det fråga om RGB-lysdioder med tre styrkanaler stiger antalet till minst 301 enskilda kablar, om man inte dessutom utökar antalet matingskablar! En sådan installation skulle vara både dyr och kräva tid och utrymme, för att inte tala om vilket arbete en lokalisering och reparation av ett kabelfel skulle medföra. Då det är fråga om en liten styrkrets som Arduino kommer man inte så långt eftersom utgångarna tar slut ganska fort.

Denna typ av installationer sköts istället genom "utlokalisering" av signal-styrningen. Huvudstyrenheten skickar ut data till styrenheter i ett nätverk. Styrenheterna är försedda med adresser och vanligtvis seriekopplade (se figur 11). De kan plocka ut det data som är addresserad till dem. Den mest utbredda standarden för denna typ av styrning är Digital Multiplexed, förkortat DMX. Standarden går även under namnet DMX-512 eftersom det ger möjlighet att individuellt styra 512 olika armaturer. Protokollet används även för andra effekter inom scenteknik bl.a. motorer och rökmaskiner kan kontrolleras med DMX. Den första versionen togs fram på initiativ av United States Institute for Theatre Technology, USITT år 1986, senare versioner har publicerats 1990 och 2004. Syftet med DMX var att göra olika tillverkares styrdon och armaturer kompatibla, vilket de inte var vid denna tidpunkt. DMX definierar elektrisk karakteristik, dataformat, protokoll och typ av uttag. (United States Institute for Theatre Technology Inc. 2010)




*Figur 11. Blockschema för ett DMX-nätverk.*

Ett DMX-nätverk har en sändare (master) vars kommandon mottagarna lyder. Sändaren är den enda som har befälsrätt – inget skickas tillbaka överhuvudtaget. Då det inte finns någon inbyggd kontrollmekanism kan DMX inte användas för pyroteknik och andra farliga applikationer (United States Institute for Theatre Technology 2010). Mottagarna har två uttag och kablarna kopplas från den ena mottagarens "ut" till följande mottagares "in". Även om strukturen i ett DMX-

nätverk vanligen är av linje-typ man kan dela upp nätverket i grenar med hjälp av sk. "splitters" vid behov.

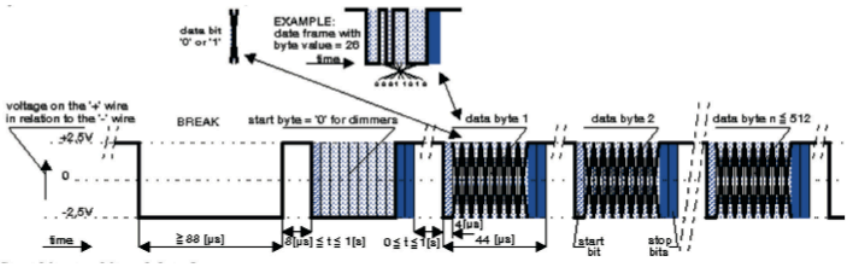
Kabeln baserar sig på RS-485-standarden och innehåller en partvinnad ledning för dataöverföring (data+ och data-) för att motverka störning samt jord (Philips 2008:17). RS-485 möjliggör kommunikation mellan maximalt 32 olika mottagare/sändare som befinner sig på upp till en kilometers avstånd från varandra med en hastighet på upp till 10 Mbps (Nelson 1995). För DMX-protokollets del har dock specificerats något mindre värden (se tabell 3).

*Tabell 3: Teknisk specifikation för DMX enligt Philips (2008:24).*

| | |
|---|---|
| Enheter per uttag | max. 30 (utan booster) |
| Adresser | max. 512 |
| Spänningsnivå | hundratals millivolt |
| Hastighet | 40 x 512 värden/sekund |
| | +/- 250 kbaud |
| Terminering | 120Ω i slutet av kabeln |
| Max. kabellängd | 500 meter utan förstärkare, obegränsad med förstärkare |
| Kabeltyp | Tvinnad parkabel<br>100-120 Ω |
| | Cat.5 S/UPT, F/UTP, SF/UTP |
| | Cat.6 U/FTP, S/FTP, S/STP |
| | Cat.7 S/FTP, S/STP |
| Kabeltopologi | seriell |
| Terminationsmotstånd | 120 Ω |
| Säkerhet | SELV (skyddsspänning) |

Mottagaren som kan vara antingen fysiskt inbyggd i eller separat från armaturen som skall styras. Mottagaren producerar en analog signal som styr armaturen som om den skulle vara kopplad direkt till huvudstyrenheten. Mottagarna seriekopplas till sändaren, vilket innebär att det i en mottagare skall finnas både ett "DMX in" och "DMX ut"-uttag (Philips 2007:19). Dessa bör vara av typen XLR5 eller RJ-45. XLR3 som används för ljudöverföring får inte användas eftersom DMX-element kan förstöras av att någon i misstag kopplar en ljudkabel med mycket högre spänning till uttaget (United States Institute for Theatre Technology 2010). I praktiken tillverkas ändå många DMX-lampor med XLR3-uttag som är mer praktiska ur kundens synvinkel eftersom denna kabel- och uttagstyp är lättare att hitta i elektronikaffärer. DMX-lampan som var tillverkad av Velleman var byggd på detta sätt. Av dessa orsaker används XLR3-uttag och kablar i detta arbete.

DMX är ett asynkront och enkelriktat dataprotokoll (Sukale 2008:36). Som man kan se i figur 12 nedan börjar en sändning med en "Break", som består av en binär nolla som sänds ut under 88µs, varefter värdena skickas ut för varje adress. Ett värde för en adress påbörjas alltid med en etta som är 8µs lång, varefter startbit, den egentliga datan och stopbit följer (Ness & Cuartilles 2006).

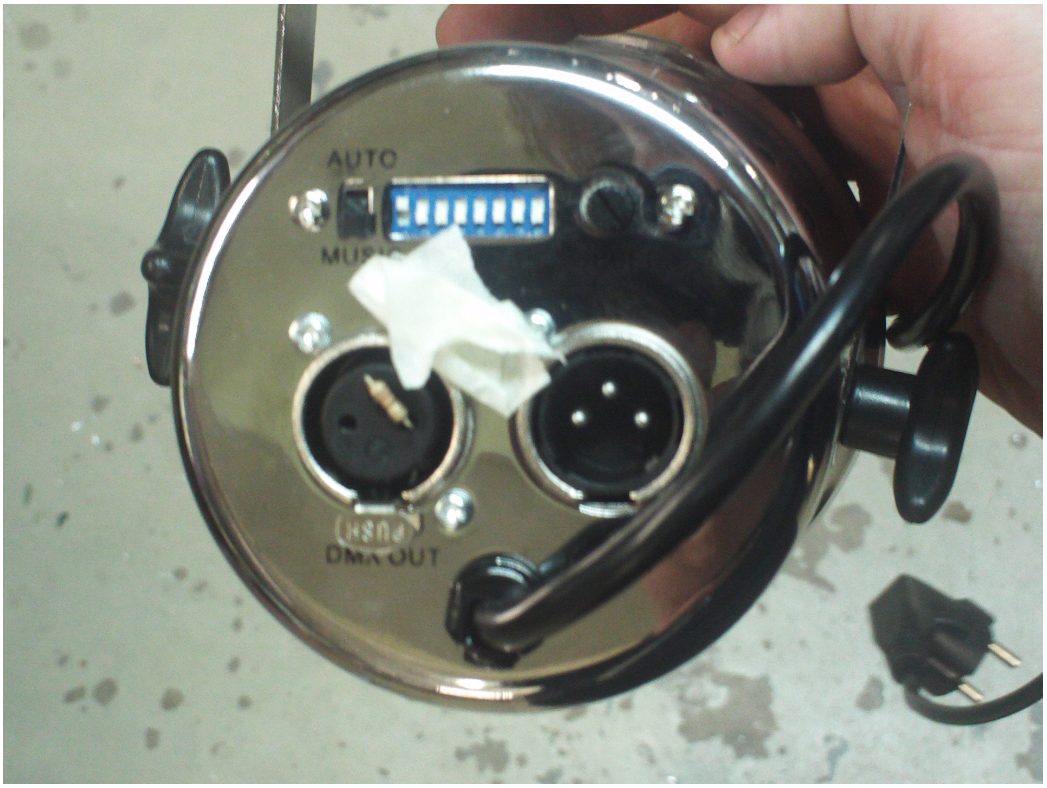



*Figur 12. DMX-protokollets struktur (Ness & Cuartilles 2006).*

Adressen ställs in manuellt på mottagaren, vanligen med en DIP-switch (se figur 13). Dataöverföringen sker så att adressdatan som sådan inte sänds, istället sänds alla värden i ordningsföljd och mottagaren räknar fram till sin egen adress . RGB-lysdioder kräver tre adresser; en för varje kanal. Adressallokeringen sker

vanligen så att man ställer in adressen för den första kanalen (rött) med DIP-switchen varpå de följande två kanalerna automatiskt representerar grönt och blått. (Philips 2008:11-16)

En nackdel med denna typ av system är att de inte är lönsamma ifall man endast vill kontrollera ett fåtal olika lysdioder, eftersom sändare, mottagare och kablar är relativt dyra. DMX byggs ändå in som en egenskap i detta arbete eftersom det är en central teknik inom belysningsstyrning.




*Figur 13. DMX-lampan som DMX-styrningen testades med.*

### 3.4.4 Övriga styrmetoder

Det finns även andra styrmetoder för lysdioder och andra belysningsarmaturer, men alla styrmetoder kan givetvis inte användas överallt. Digital Addressable Lighting Interface, DALI är ett protokoll som enligt Simpson används för att kontrollera "kommersiell och arkitektonisk belysning" och beskrivs som en rak motsats till DMX (Simpson 2003:292). Protokollet presenteras ändå i Simpsons

bok genast efter DMX i samma kapitel, så att man vid första anblick kan tro att det är ett användbart alternativ till DMX. I DALIs manual nämns ändå ingenting om lysdioder (DALI 2001). I Simpsons bok framkommer att protokollet bl.a. har en betydligt mindre överföringshastighet än DMX och stöder endast 64 enskilda belysningsarmaturer (Simpson 2003:292-300). Protokollet påminner mer om X10 och de andra systemen som presenteras senare i boken; det är användbart för centralstyrd på- och avkoppling av lampor men för grovt och klumpigt för detta arbetes ändamål; dessutom är denna typ av teknik är förhållandevis dyr och svårtillgänglig för privatpersoner.

Lysdioder kan styras med sk. led-drivers som Philips' 4794. Kretsen, som även lätt kan användas tillsammans med Arduino, är ett skiftregister som används för att antingen slå av eller på ett stort antal lysdioder. Eftersom denna typ av styrning är binär, är denna typ av krets oanvändbar i detta projekt.

En teknik som kunde användas är MIDI-protokollet. MIDI är en förkortning av Musical Instrument Digital Interface och utvecklades ursprungligen för musik-branschens behov men används även inom ljusstyrning med hjälp av utvidgningen MIDI Show Control. Det går att bygga ett MIDI-gränssnitt av en Arduino med hjälp av endast ett fåtal externa komponenter. Protokollet är användbart inom underhållningsindustrin där man vill styra ljuseffekter i takt med musik. Det kommer ändå inte att implementeras i detta arbete eftersom de som behöver denna teknik sannolikt alltid har tillgång till eller behöver en större kontrollanläggning. För övrigt är belysningsarmaturer som kan kontrolleras med MIDI dyra och relativt svåra att få tag på.

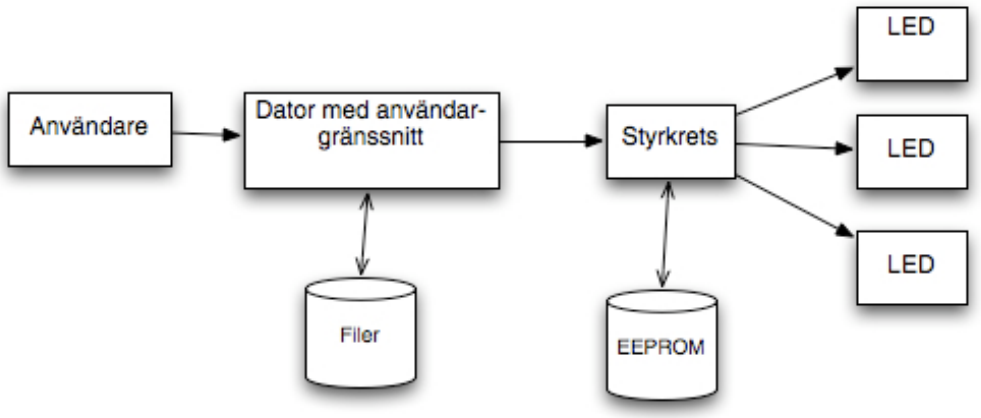
I sammanhanget kan även nämnas Art-Net som är en metod för sändning av DMX-signaler över ett Ethernet-nätverk. Detta möjliggör utnyttjande av redan befintlig nätverksarkitektur samt tillåter snabbare signalöverföring. (Howell 2007)

# 4 RESULTATREDOVISNING

## 4.1 Översikt

Arbetets mål var att skapa en krets som skickar ut varierande styrsignaler som programmeras på en dator varefter kretsen kan fungera självständigt. Processen kan indelas i fyra delar; användaren matar in data, data skickas till styrkretsen, data sparas så att den senare kan läsas och slutligen används detta data för att skapa dessa signaler (se figur 14). Eftersom arbetet betonar själva styrningen kommer denna att behandlas utförligast i denna resultatredovisning.

Just ur de delar av koden som handhar själva skapandet av styrsignalerna framgick det att det som ser logiskt ut på papper och även fungerar som en enskild kod ofta antingen slutar fungera eller stör andra funktioner då man kopierar in delarna i huvudprogrammet. Detta ledde till att alternativa lösningar fick prövas med varierande resultat, och i de fall där funktionerna fortfarande ledde till problem och konflikter var den slutliga lösningen att välja bort mindre viktiga funktioner. I detta kapitel beskrivs programmens huvuddrag samt de problem som uppstod. Koderna som sådana finns fullständigt bifogade i slutet av detta arbete; användargränssnittets i bilaga 3 och kontrollerns i bilaga 4. Vissa funktioner som inte fungerat har sparats som bortkommenterade i koden.




*Figur 14. Allmänt schema för systemet.*

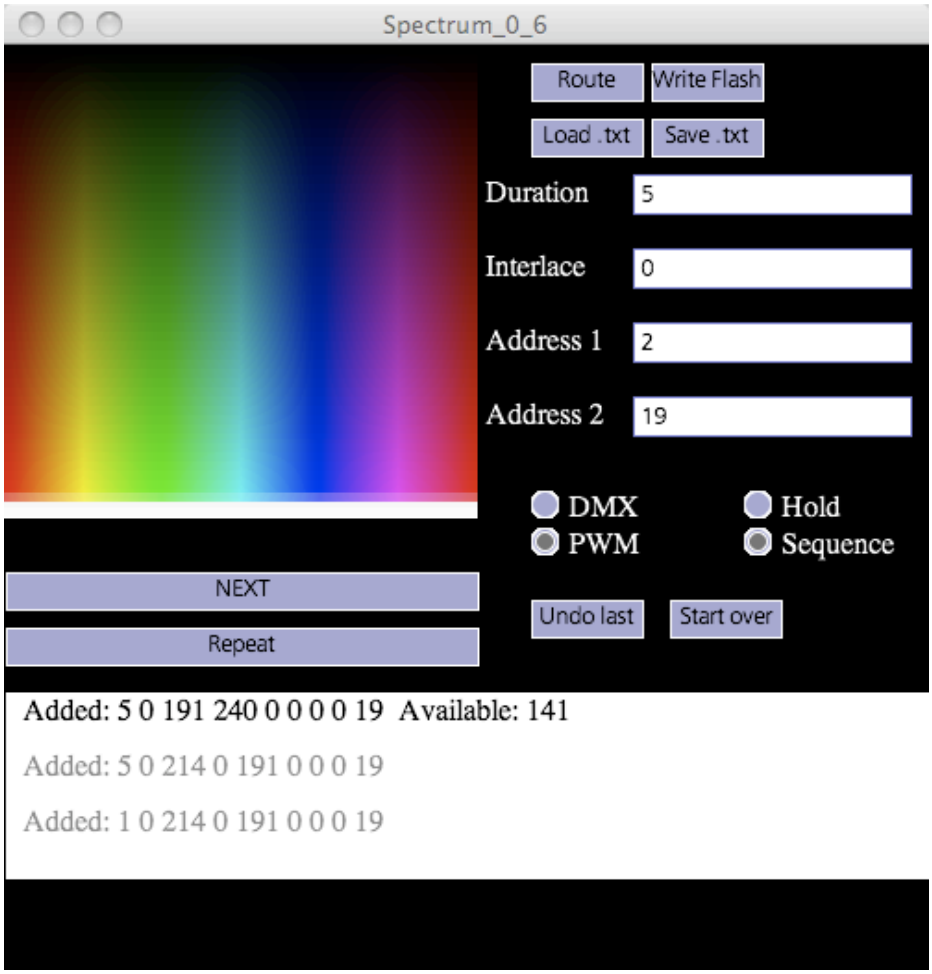
## 4.2 Dataöverföring, struktur och lagring

### 4.2.1 Användargränssnittet

Gränssnittet som döptes till Spectrum efter färgspektret som dominerar rutan där användaren väljer färg är den enda kontakten som slutanvändaren har med systemet. Det låter användaren snabbt mata in nödvändig information som sedan skickas till styrenheten samt kan spara och läsa information från textfiler. Gränssnittet är enkelt och innehåller det som användaren behöver och inget onödigt (se figur 15). Programmet är skrivet med Processing använder sig av programbiblioteket Interfascia för knappar och textrutor.

Färgsekvenserna byggs upp av "händelser" som användaren programmerar genom att svara på följande frågor:

- Vilken färg?
- När skall händelsen börja?
- Hur lång tid?
- Vilken eller vilka komponenter gäller det?




*Figur 15. Användargränssnittet.*

Användaren matar in totalt sju parametrar för en händelse som programmet sedan lagrar i sju listor, där data för en händelse finns på samma rad i varje lista. Parametrarna är: tid, interlace, röd, grön, blå, adress 1 och adress 2. I programmet används de engelska motsvarande beteckningarna time, interlace, red, green, blue, address1, och address2.

Parametern "time" berättar under hur lång tid händelsen skall genomföras. Färgparametrarna jämförs med dem från den föregående händelsen. Är dessa

oförändrade hålls lysdiodens färg konstant under tidsperioden som finns i parametern time. Finns det en skillnad förändras lysdiodens färg från den föregående händelsens värden till de i den nuvarande. Om "time"-parametern är noll ändras färgen omedelbart.

Parametern "interlace" innebär att en händelse utförs samtidigt som en annan pågår. Detta innebär att systemet kan kontrollera flera händelser samtidigt, t.ex. en lysdiodsgrupp kan byta färg från rött till vitt samtidigt som en annan lyser med konstant orange sken. Interlace-parametern läses av redan vid föregående händelse; parametern berättar hur snabbt händelsen skall börja efter att den föregående händelsen börjat.

Adressparametrarna används för att beteckna vilken komponent händelsen gäller. Dessa kan användas på fyra sätt. Har man inte angett någon adress skickas signalerna ut genom alla PWM-pinnar. Om endast parametern "address 1" är definierad skickas en DMX-signal ut med denna adress. Är endast parametern address 2 definierad skickas signalerna ut till den PWM-port som har denna adress. Om båda adressparametrarna är definierade skickas signalen ut genom DMX-porten till alla belysningselement med adresser från "address 1" till och med "address 2".

Då det endast är möjligt att spara en byte i denna tabell betyder det att adressen inte kan vara större än 255. Eftersom DMX-protokollet är designat för att styra 512 enskilda belysningsarmaturer innebär det att det inte utnyttjas till fullo. Denna begränsning är ändå betydelselös eftersom det endast finns en fysisk DMX-port på kontrollern till vilken man enligt protokollets specifikationer högst kan ansluta 32 eller 30 mottagare varpå det maximala kanalantalet som kan behövas vid färgstyrning uppgår till 3*32 dvs. 96 (Sukale 2008:36, Philips 2008:24). Detta kan försvaras med denna produkt är avsedd för mindre applikationer, där man inte har hundratals belysningsarmaturer. För övrigt kan nämnas att minnets begränsningar, som presenteras senare i detta kapitel, snabbt skulle komma emot även om det vore möjligt att utnyttja alla 512 adresser.

Parametrarna skickas iväg tillsammans med en "etikett" som läggs till i början av varje värde för att underlätta sorteringen av data på systemplattformen. Sorteringen bygger på en kod från ett exempelprogram där man styr tre kanaler genom att mata in data i terminalens gränssnitt (McRoberts 2009:58-65).

### 4.2.2 Arduinons program "Color Controller"

På Arduinon väntar programmet som döpts till "Color Controller" på inkommande data; då det inte kommer någon data avläses minnet. Dataöverföringen sker över en USB-kabel; efter en enkel "handskakning" skickar datorns applikation data händelse för händelse till Arduinon. Överföringen sker med en hastighet av 9600 baud. Data kan skickas som "route" eller "write". Då data skickas som "route" sparas den endast i programmets tabeller som initialiseras varje gång Arduinon startas om. Detta är bra då man vill testa en färgsekvens utan att radera en tidigare sådan. Skickas data däremot som "write" sparas den dessutom permanent i Arduinons EEPROM, vilket i praktiken innebär att den tidigare sekvensen skrivs över. Då ett "write"-kommando skickas, skickas samtidigt också antalet händelser som ett skilt värde. Informationen sparas på minnesplats 1023 i EEPROM och används av programmet som gräns för hur långt programmet skall gå innan det skall börja om från minnesplats noll då det läser av sparad data, dvs. det berättar vid vilken minnesplats den aktuella sekvensen tar slut.

Arduino Duemilanove som har ett minne på 1024 byte kan spara totalt 146 händelser då en händelse kräver 7 byte minnesutrymme. Om det funnits mer minnesutrymme kunde fler parametrar ha använts för att underlätta styrningen. Mängden händelser borde räcka för de användningsområden denna kontroller riktar sig till. Större datamängder kräver antingen en plattform med större kapacitet (Arduino Mega) eller användning av ett externt minne. Totalt 14 utgångar finns men antalet sjunker i praktiken till 12, eftersom utgångarna 0 och

1 måste vara lediga under seriekommunikation och syftet är att bygga en permanent "shield", dvs. användaren skall inte behöva ta isär styrenheten varje gång nya värden skall matas in. Då PWM-styrning kräver tre utgångar (röd,grön,blå) betyder det att tre RGB-lysdioder eller RGB-lysdiodsgrupper kan styras separat, då även utgångar behövs för DMX-signal och indikeringslampa.

## 4.3 Styrningens planerade databehandling

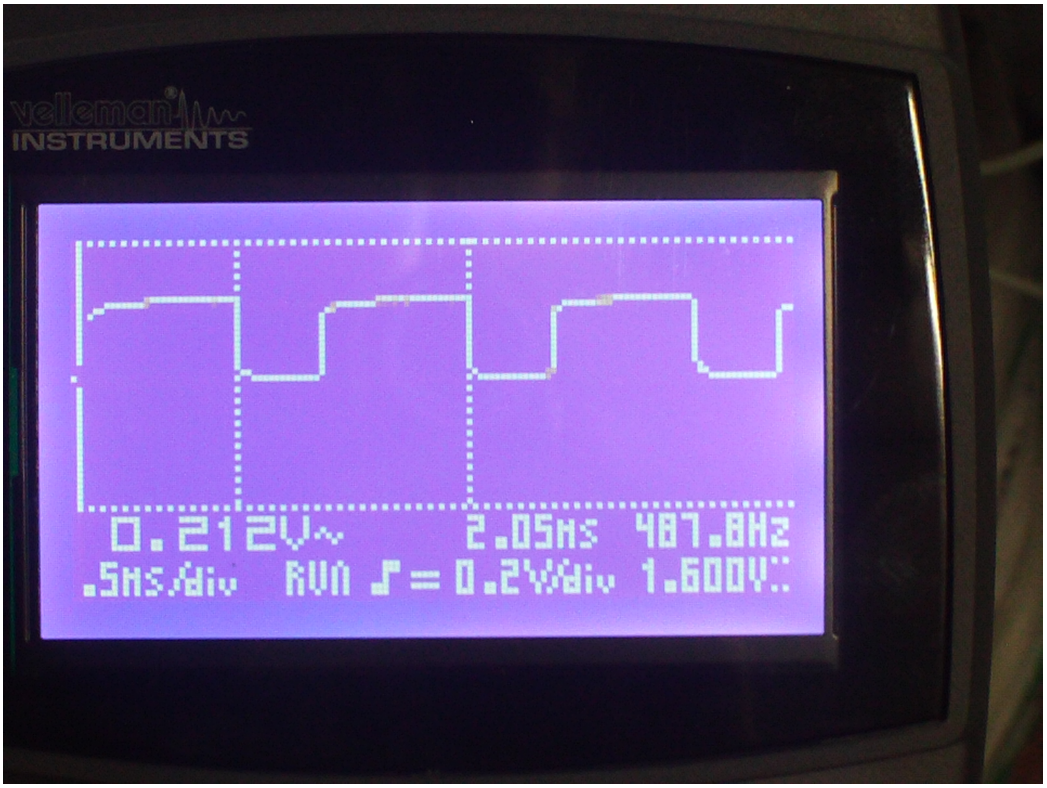
Då programmet körs läses data från minnet om inte nyare data kommit in från datorn. Programmets uppgift är att antingen producera analoga styrsignaler med hjälp av pulsbreddsmodulation eller fungera som en DMX-sändare. Till detta används de sju parametrarna som tidigare nämndes. Eftersom antalet händelser som ryms i minnet blir mindre om antalet parametrar per händelse ökar är det naturligt att låta mikrokontrollern "läsa mellan raderna" där det är möjligt. Vissa av de funktioner som kunde ha varit separat utskrivna om det funnits mer plats eller om datorn kontinuerligt skickat data till styrenheten har istället integrerats i de parametrar som finns för att spara utrymme.

Programmet kollar hur data skall skickas ut beroende på adresserna. Är den första adressparametern tom skickas data ut som PWM-signal till den PWM-port som är beskriven i den andra adressrutan, finns det en adress i endast den första rutan betraktas det som en DMX-adress. Är båda adressrutorna ifyllda skickas signalen ut till DMX-adresserna i intervallet mellan de adresser som specificerats i adressrutorna.

Styralgoritmen baserar sig i sin korthet på tre ärvärden (nuvarande), tre börvärden (följande) och en tid. Genom att kontrollera skillnaden mellan dessa värden, och dividera det med tiden får man hur stora steg ärvärdet skall förändras per en viss förutbestämd tidsperiod för att komma till börvärdet på den tid som angivits. Om man omedelbart vill förflytta sig till börvärdet är tidsvariablen noll. Två specialfall är repetition av föregående sekvens och

konstant färg. För att inte behöva lägga till fler variabler sköts dessa med två händelser; man hoppar omedelbart till det värde som man vill börja från, varpå man kan upprepa förflyttningen eller hålla värdet konstant (börvärde=ärvärde).

Styrprogrammet har en upplösning på 100 millisekunder, vilket genom ett test i praktiken visat sig vara tillräckligt för denna applikation. Detta lämnar mycket tid för plattformen att göra annat vid behov. Signalen som moduleras ut då man använder kommandot analogWrite() har, som framgår ur figur 16, en frekvens på ungefär 490 Hz vilket räcker till.




*Figur 16. PWM modulering för ungefär 70% med mikrokontrollerns egna PWM-register.*

Det finns dock ett problem; använder man detta kommando har man endast tillgång till sex stycken PWM-utgångar. Detta beror inte på bootloadern eller på något annat som utgör själva "Arduino-delen" utan på plattformens mikro-kontroller ATMega328 som endast stöder sex PWM-utgångar (Atmel 2009:1). Då vi är intresserade av att öka antalet utgångar till nio är det enklaste sättet att

skapa funktioner som helt enkelt slår av och på de digitala utgångarna med pauser, alltså sk. "bit banging PWM" (Shirriff 2009). En annan lösning är att låta en extern IC-krets som Texas Instruments' TLC5940 sköta moduleringen. I detta arbete användes ändå den förstnämnda metoden, eftersom meningen är att se hur långt man kan komma genom att bara skriva kod.

Det är i teorin möjligt att implementera också DMX genom denna typ av "bit banging", men då hade koden för denna blivit längre och mer komplicerad eftersom det handlar om en så tidskritisk applikation som att skicka ut data. Dessutom skulle en eventuell felsökning och fininställning ha tagit oacceptabelt lång tid. Planen var att använda ett färdigt programbibliotek vid namn DmxSimple.

## 4.4 Implementering av styrningen i praktiken

### 4.4.1 PWM

Under arbetets gång framkom ett litet problem som visade sig få större konsekvenser. Eftersom RGB-lysdioder använder sig av tre kanaler bör dessa styras samtidigt. För den ovannämnda metoden fanns ett kodexempel för att skapa en PWM-signal för en utgång med en frekvens på 1 kHz. Koden sattes in i en funktion där den utfördes tre gånger; en gång per kanal, vilket gav en tre gånger längre pulstid. Detta innebär att frekvensen och duty cyclen är en tredjedel av den ursprungliga. Som sådan fungerade denna kodsnutt problemfritt då den testades separat, dvs. uppladdades som ett enskilt program.

```
void pwm(int erre, int gee, int bee) //alltid 100 ms
{
 for(int i; i<33; i++) // r,g och b tar 3 ms att genomföra 3*33=99ms
 {
 digitalWrite(rPin,HIGH);
 delayMicroseconds(erre);
 digitalWrite(rPin,LOW);
```

```
  delayMicroseconds(1000-erre); // cykel för röd (tot. 1000 us)
  digitalWrite(gPin,HIGH);
  delayMicroseconds(gee);
  digitalWrite(gPin,LOW);
  delayMicroseconds(1000-gee); // cykel för grön (tot. 1000 us)
  digitalWrite(bPin,HIGH);
  delayMicroseconds(bee);
  digitalWrite(bPin,LOW);
  delayMicroseconds(1000-bee); // cykel för blå (tot. 1000 us)
  }
}
```

Problemen uppstod då den kopierades in i det egentliga programmet. Koden, som då den kördes som ett ensamt program producerat nästan lika jämna förflyttningar som analogWrite() kommandot gav då den kopierats in i det egentliga programmet upphov till hysteriskt blinkande. Då denna metod inte verkar vara användbar finns det tre uttag mindre än planerat för lysdiods-styrning. Även om detta inte verkar vara ett större problem blir situationen allvarligare då man planerar att utöka programmet. Mega, den största, snabbaste och dyraste plattformen inom Arduino-serien erbjuder 42 digitala utgångar vilket betyder att man kunde styra 14 skilda RGB-lysdiodsgrupper som tillsammans med DMX-stöd skulle bilda ett system som räcker till allt utom scentekniska applikationer. Om man i praktiken endast kan använda de utgångar som kan skapa PWM-signaler med hjälp av mikrokontrollerns egna PWM-register kommer man inte ens nära detta antal. Mega stöder PWM direkt på 12 utgångar, vilket innebär att över 2/3 av utgångarna i detta fall förblir oanvända! Som tidigare nämnts finns det en möjlighet att använda externa IC-kretsar som sedan skapar PWM-signaler, och en tredje möjlig lösning är att använda sig av lågnivåfunktioner för att sätta benen högt och lågt eftersom dessa utförs snabbare än de som skrivs för Arduino. Slutsatsen är i varje fall att Arduino ensam inte är den bästa möjliga lösningen för applikationer som kräver ett stort antal PWM-utgångar.

### 4.4.2 DMX

Det andra problemet uppenbarade sig vid implementeringen av DMX-delen. Planen var att använda programbiblioteket DmxSimple eftersom manuell implementering av protokollet visade sig vara alltför omfattande. Det visade sig snart att den testade lampan inte lydde de kommandon som skickades ut då programbiblioteket testades med exempel som enligt programbibliotekets författare skulle fungera. I denna typ av applikation kan problemet finnas på fyra platser; i koden, i styrelektroniken, i bussen och i mottagaren. Sammansättningen testades med en exempelkod som använde funktioner på en lägre nivå. Lampan pulserade rött precis som den skulle, vilket i praktiken betydde att felet låg i det tidigare testade programbiblioteket.

Testprogrammet med dessa DMX-funktioner fungerade som de skulle, men problemen var långt ifrån över. Då funktionerna kopierades in som en del av det egentliga programmet visade det sig att de störde dataöverföringen från datorn till Arduinon. Lysdioderna uppförde inte sig som förväntat. Vid en avläsning av innehållet i EEPROMen där värden för tid, färg och adress sparas visade det sig att Arduinon inte hade fått med alla värden som skickats från datorn. I praktiken betyder detta att all data inte hamnar på rätt plats i tabellen; tidsvariablen från följande händelse hamnade på adressvariablens plats, tidsvariablerna ersattes med färgvariabler och färgvariablerna med värdena för adressvariablerna. DMX-sändningen handhas av funktioner som inte kallas på om det inte är nödvändigt, men felet har troligen något att göra med "pins_arduino.h" som inkluderas i början av koden om man vill sända DMX. Hur som helst innebär detta att det inte samtidigt kan finnas både DMX och seriell kommunikation.

Det finns tre dåliga sätt att lösa problemet. Man kan lämna bort dataöverföringen och ha slutanvändaren att skriva in de önskade värdena i Arduinos programvara och sedan ladda upp programmet på nytt varje gång användaren vill ha nya färger och tider. Detta skulle vara så långt från användarvänlighet som man kan komma. Ett mer användarvänligt alternativ är att utelämna lagringen av data och

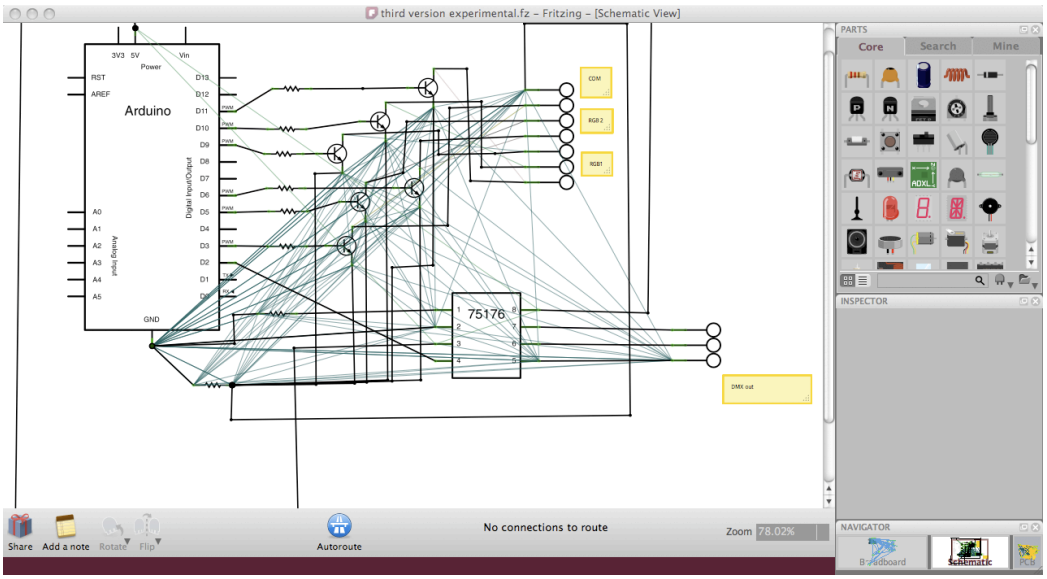
hålla kretsen konstant kopplad till en dator som slavenhet. Datorn sköter i detta fall styrningen och Arduinon fungerar som en slags "shield". Då är slutresultatet är ett datorstyrt USB-DMX gränssnitt gjort av delar som kostar under hälften av vad en fabrikstillverkad enhet kostar och som dessutom ger möjlighet att styra lys-diodsgrupper direkt med PWM; en kombination som inte finns i dessa dyrare produkter. Problemet återstår att detta tar upp en hel dator som sänder ut styr-kommandona som Arduinon sedan översätter. Dessutom är det sannolikt att dataöverföringen även i detta fall störs av DMX-koden och att Arduinon vidarebefodrar förvrängd data som sedan gör att lamporna som är kopplade till DMX-nätverket inte kan styras korrekt. Den tredje lösningen är att utelämna DMX-styrningen helt vilket gjordes eftersom resten av koden då fungerade felfritt.

## 4.5 Kretsen

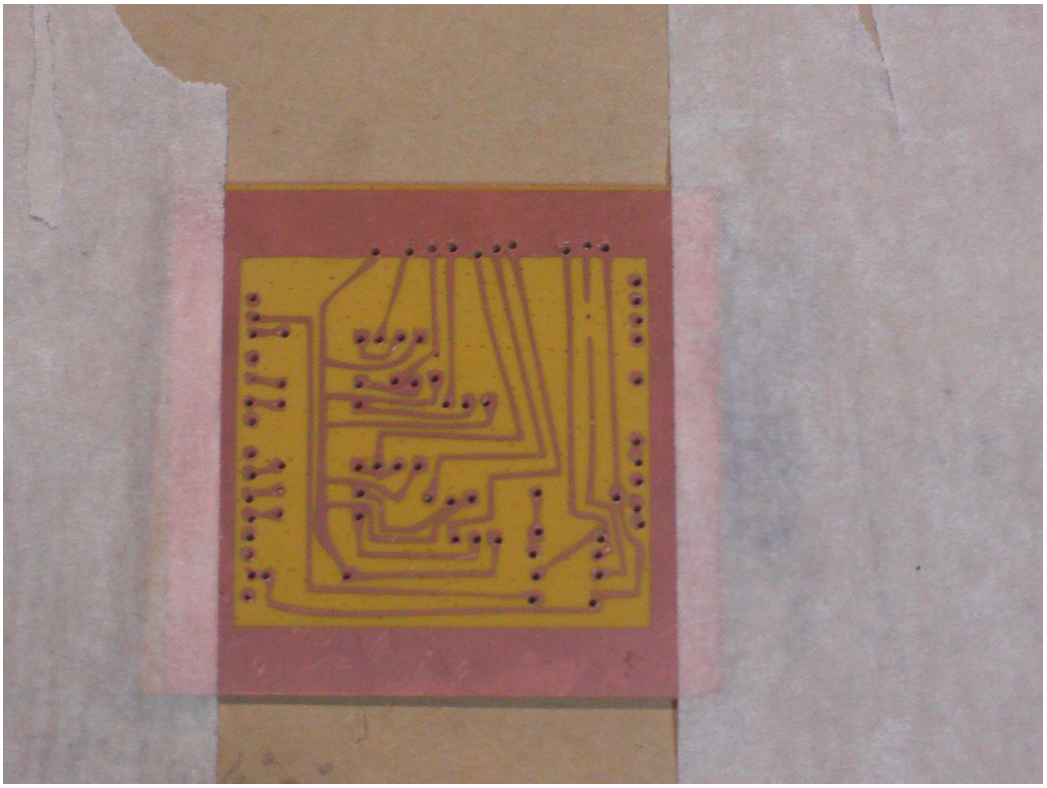
Eftersom man inte direkt kan koppla lysdioder och DMX-mottagare till Arduinons uttag var det nödvändigt att utöka plattformen med extra komponenter. Kretsen för detta projekt byggdes i form av en shield som lätt kan tryckas fast ovanpå Arduinon, men testades först i sedvanlig ordning på ett breadboard, där det var lätt att prova olika komponenter. Det finns två enskilda delar i denna krets; en för varje styrmetod. DMX-delen är inte implementerad i koden på grund av ovannämnda problem. De komponenter som skulle ha använts finns ändå med på kretskortet så att protokollet kan användas ifall ett användbart programbibliotek skulle dyka upp. Kretsschemat finns i bilaga 2.

Vid ritning av kretsscheman i detta arbete användes programmet Fritzing (se figur 17). Programmet utvecklas vid Fachhochschule Potsdam och riktar sig särskilt till kretsritning för Arduino. Fritzing följer samma filosofi som Arduino och Processing, istället för att studera manualer i timtal skall en ny användare snabbt kunna komma igång med arbetet och testa programmets egenskaper. Komponenterna läggs ut på en virtuell breadboard varefter programmet kan göra

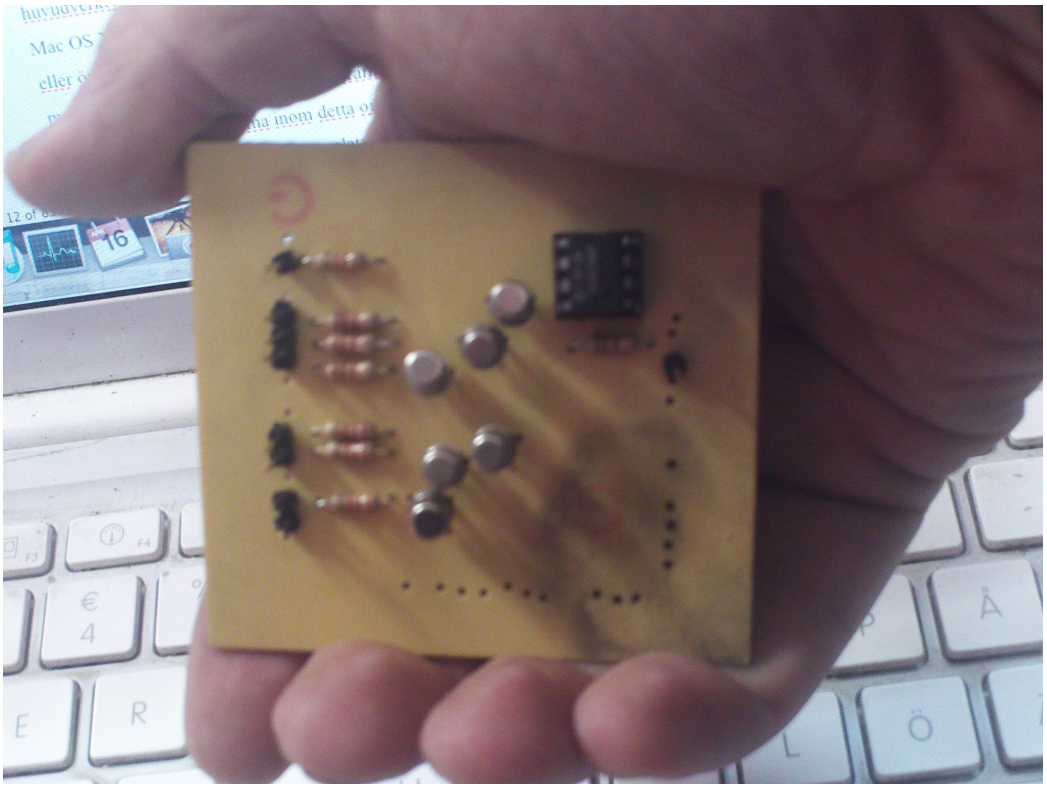
tekniska ritningar och kretsscheman. Programmet är fortfarande under utveckling och programmeringsteamet "kan inte garantera någonting". I detta arbete uppkom inga problem. (Fritzing 2010)




*Figur 17. Arbetsmiljön Fritzing.*



*Figur 18. Kretskortet etsat och borrat.*



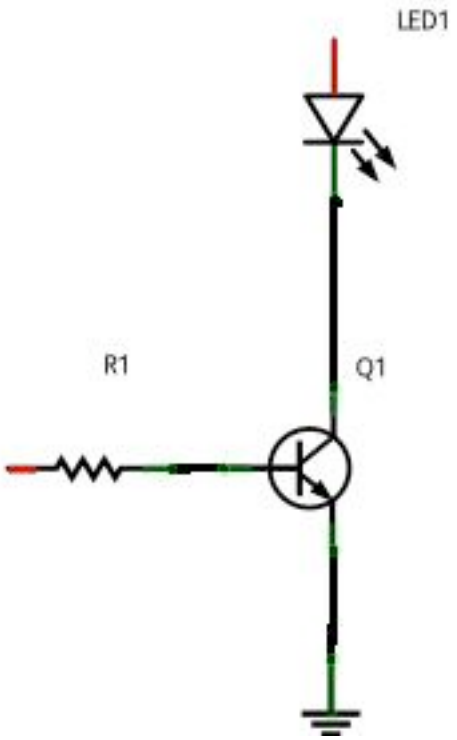
*Figur 19. Kretskortet med alla komponenter förutom kontakterna.*

Kretsen tillverkades på ett sätt som avviker från det normala men som gav ett relativt bra slutresultat (se figur 18, 19 ovan) då man beaktar författarens ringa erfarenhet inom detta ämne. Schemat konverterades till en vektorfil och skars ut på vinyltejp med en skärplotter. Schemat tejpades på kretskortmaterial av koppar som sedan etsades med en järntrikloridlösning. Etsningschemat finns i bilaga 1. Även om förväntningarna inte var särskilt höga var det som kom fram under tejpen ett dugligt kretskort. Arduinons storlek gjorde att komponenterna måste vara så nära varandra att kretskortsbanorna i vissa fall kom så nära varandra att man måste reparera dem i efterhand. Slutsatsen är att denna metod är användbar om man jobbar med elektronik då och då (dvs. inte varje vecka) men har tillgång till en skärplotter. En fördel är att man kan korrigera banor i sista stund då man har tejpat kretskortet, men metoden kan ändå inte rekommenderas för små kretskort.

### 4.5.1 PWM-delen

Även om den kan styra ett mindre antal lysdiodsgrupper tar PWM-delen rent fysiskt upp en betydligt större del av kretsen. Denna del kopplas direkt till lysdioderna och fungerar som motstånd som reglerar strömmen som flyter mellan dessa och strömkällan. Den centrala komponenten i denna del är transistorerna som det finns en per kanal av.

Då RGB-lysdioder vanligen är av typen "gemensam-anod" betyder detta att lysdiodernas anoder gemensamt kopplas till strömkällans pluspol medan de tre katodkanalerna passerar genom kretsens transistorer som är av typen NPN. Transistorn är en halvledarkomponent som låter ström passera från kollektor till emitter om ström leds in genom basen. Detta innebär att transistorn kan användas som en förstärkare. Arduinons utspänning på maximalt 5 V förstärks så att man kan styra större spänningar. Basen kopplas till benet som skickar ut PWM-signalen genom ett 2,2 kΩ motstånd (se figur 20 nedan).



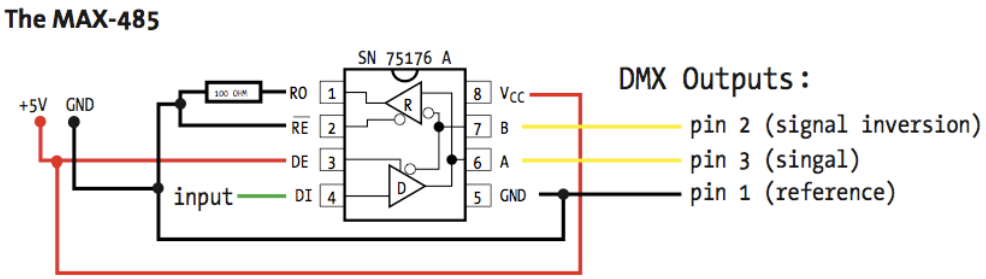

*Figur 20. Kretsschema över styrelektroniken för en PWM-kanal.*

Kretsens uppgift är inte att mata ström till lysdioderna utan att begränsa strömmen som passerar. Matningen kommer från en transformator, en switchad (pulserande) strömkälla eller vanliga batterier och gränsen för hur många lysdioder man kan koppla in beror på hur mycket dessa klarar av. Antalet lysdioder som kan seriekopplas begränsas också av hur stor kollektorström transistorn tål. I denna krets används transistorer av typen 2n2221 som enligt försäljarens hemsida tål en ström på maximalt 100 mA. Kretsdesignen baserar sig på en som fanns på sidan OpenDMX (2010), dock med vissa förändringar. Transistortypen ersattes eftersom den som föreslagits var svårare att få tag på. Då man testar kretsen med batterier krävs det att jorden även ansluts till Arduinons jordben. Ett skyddsmotstånd har kopplats in ifall att man även vill reglera lysdioder av starkare strömkällor. Om man dessutom byter ut transistorerena till de som avsågs i originaldesignen kommer man upp till en

maximal kollektorström på 800 mA , vilket betyder totalt 2,4 A då kanalerna möts!

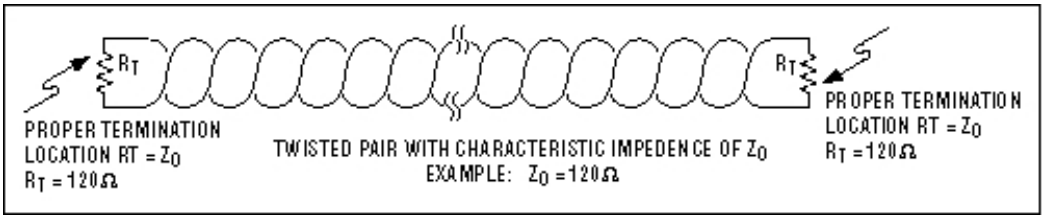
### 4.5.2 DMX-delen

Delen som kontrollerar DMX-utsignalen består endast av två komponenter; en IC-krets och en resistor. Designen är densamma som använts av Ness och Cuartilles (2006, se figur 21).




*Figur 21. Kretsschema över DMX-delen (Ness & Cuartilles 2006).*

Mikrokontrollern producerar signalen, men eftersom protokollet kräver att signalen skickas både som positiv och negativ användes en differentiell buss-sändare (eng. Differential Bus Transceiver) betecknad SN75176. Till denna IC-krets matas signalen tillsammans med en spänningsförsörjning på 5 volt, och de tre utsignalena data+, data- och jord kopplas till ett XLR3-uttag. Med en XLR-kabel ansluter man mottagaren som man vill kontrollera. Testobjektet som användes var en av Velleman tillverkad lampa av modell VDPLPS36C. Denna innehåller ett antal röda, gröna och blå lysdioder samt en mottagare för DMX-signaler. RS-485 kräver terminering av dataledarna med ett 120 Ω motstånd i varje ände av nätverket (se figur 22 samt figur 13), men eftersom testnätverket endast innehöll en sändare räcker det med ett enda termineringsmotstånd i slutändan (Maxim 2001, Nelson 1995:4).




*Figur 22. Terminering av en RS-485 buss (Maxim 2001).*

# 5 DISKUSSION

## 5.1 Översikt

Arbetets mål var att bygga ett Arduino-baserat kontrollsystem för RGB-lysdioder som skulle vara färdigt att använda som sådant. Det var meningen att kontrollsystemet skulle fungera självständigt efter programmering, dvs. att det inte permanent skulle behöva ta upp en dator för att fungera. Som det framgick ur resultat-redovisningen gick det inte att få med alla önskade funktioner. I denna diskussion föreslås en del ändringar i den nuvarande konfigurationen så att man få med alla funktioner som planerades från början. Dessutom presenteras möjligheter att utvidga systemet ytterligare.

Om man snabbt ser på specifikationerna i början av arbetet kan man konstatera att projektet nog uppfyller dem på alla punkter; användaren kan själv planera händelseförlopp för mer än en lysdiodsgrupp och systemet måste vara anslutet till en dator endast då ett nytt händelseförlopp laddas upp. Användaren har betydligt mer kontroll över systemet än med en en IKEA-byggsats men systemet är ändå relativt litet och lätt att använda. Under arbetets gång dök vissa otrevliga överraskningar upp som ändå kunde lösas.

## 5.2 Lösningsförslag till ovanstående problem

Olyckligtvis uppstod de kvarstående problemen på värsta möjliga plats; i själva styrningen. Då styrning av RGB-lysdioder kräver tre gånger mer utgångar än en vanlig lysdiod tar PWM-utgångarna slut tre gånger snabbare. Eftersom plattformen endast kan skicka ut analoga signaler på sex utgångar räcker detta endast till två skilda lysdioder eller lysdiodsgrupper. Försöken att använda de digitala utgångarna såg lovande ut fram till att detta testades som en del av det egentliga programmet. För att öka antalet PWM-utgångar rekommenderas den integrerade kretsen TLC5940 som särskilt marknadsförs för lysdioder. På

Arduinos diskussionsforum, personliga hemsidor och Youtube presenteras flera byggprojekt där en eller flera TLC5940 har kopplats till en Arduino för att skapa olika slags färgeffekter med lysdioder. TLC5940 ser ut som en perfekt lösning, men den finns inte i alla elektronikförsäljares sortiment. Ett annat problem är att man inte kan vara säker på om även den stör seriekommunikationen mellan datorn och Arduinon.

DMX-styrningen är ett svårare problem att lösa. Eftersom den fungerande exempelkoden gav upphov till störningar i seriekommunikationen är man tvungen att ändra koden. En bra lösning vore att kunna separera programmen helt och hållet, vilket är omöjligt eftersom Arduino endast kan spara ett program i taget. DMX-sändningen kunde liksom PWM-styrningen skötas av en extern IC-krets, men det verkar inte finnas någon enskild krets som kan producera DMX-signaler. Kanske en planering av en sådan integrerad krets kunde vara ett passande ämne för ett examensarbete. Beslutet att lämna bort DMX-styrningen vägdes mot alternativet att lämna bort dataöverföringen helt och hållet vilket inte kom på fråga. DMX-styrning var ändå inte det centrala temat i arbetet, utan den skulle endast fungera som en "förlängning" för de fall då användaren ville ansluta fler PWM-grupper än det fanns uttag.

## 5.3 Förslag till vidareutveckling

Även om ovannämnda problem kommer att lösas finns det fortfarande en del utvecklingsmöjligheter. Genom att utöka minnet kan man spara flera färg-sekvenser. Som tidigare nämnts stöder Arduino olika typer av externa minnen. Bland de projekt som presenteras på plattformens hemsida finns en krets för SD-kort som innebär att man i praktiken kan öka minneskapaciteten oändligt.

En egenskap som kunde göra systemet ännu mindre datorberoende är fjärr-programmering med t.ex. Bluetooth eller Zigbee. Det finns en shield för Zigbee-kommunikation och en av Arduino-plattformerna, ArduinoBT, kan

kommunicera över Bluetooth-protokollet. Då programmen inte skrivs för en viss mikrokontroller utan för operativsystemet dvs. Arduino-bootloadern som finns på den, kan koden kopieras utan några som helst förändringar precis som ett datorprogram. Genom att använda dessa lösningar kan nya färgsekvenser programmeras från en dator som befinner sig i rummet brevid. Processing kan kompilera program för både Windows, Macintosh och Linux och skapa Java-applets. Man kan kommunicera med en Arduino med alla språk som stöder seriell datakommunikation, vilket möjliggör mer avancerade användargränssnitt. En intressant sammansättning kunde vara ett system där användaren kan programmera nya färgsekvenser över en Bluetoothförbindelse från en mobiltelefon. I sin rapport över mikrokontrollerprogrammering på Macintosh-datorer förutspår McKerrow (2007:9) för övrigt att även detta område kommer att dra nytta av att programmerare börjat göra program för Apples mobiltelefon iPhone, eftersom dessa programmerare även kan vara intresserade av annan slags programmering för operativsystemet.

I sin nuvarande form är styrkretsen en slags "svart låda" som endast består av ingångar och utgångar. Det naturliga för en teknisk apparat är att styrningen sker med knappar och en eventuell skärm som finns integrerade i själva apparaten. Frågan är dock hur lång programkoden skulle bli om användargränssnittet skulle finnas på plattformen. Under arbetets gång hittade jag ett projekt vid namn Pyxis OS där en aktiv byggare har gjort grunden till en programmerbar handdator med hjälp av några Arduinon staplade ovanpå varandra. Detta kunde utvecklas till en trådlös programmeringsenhet. Genom en längre utveckling i denna riktning kunde man låta användaren skapa sk. interaktiva installationer som diskuteras i Sukales (2008) diplomarbete.

# 6 SLUTSATSER OCH AVSLUTNING

I sin enklaste form där man har en eller fler lysdioder som alla lyser lika starkt är tekniken enkel, man behöver bara hålla strömmen under en viss gräns. Om man automatiskt vill variera ljusstyrka och färg kommer man till en helt annorlunda situation; man behöver ett styrsystem som i många fall inte finns färdigt att köpa. Denna styrkrets riktar sig till de fall då man har en förhållandevis liten mängd lysdioder som man vill kontrollera. Även om vissa funktioner inte finns med fungerar i alla fall de centrala delarna och med de tillägg som nämnts kan kretsens egenskaper utökas.

Arduino är som sådan är en trevlig plattform att jobba med och jag kan framför allt rekommendera den för hobbybruk. Programsyntaxen är enkel och det finns mycket information, tips och kommenterade exempelprogram på internet. En typisk mikrokontroller välkomnar nybörjaren med räknare, timers och register som tar timtal att bekanta sig med. På en Arduino kan nybörjaren få en lysdiod att blinka på ett par minuter och bygga en trafikljuskorsning med tryckknapp på några timmar. Arduino som sådan räcker till många byggprojekt och dessutom har man tillgång till alla mikrokontrollerns lågnivåfunktioner om man vill. Ett av de första projekten jag stötte på då i början av detta arbete var ett oscilloskop där data samlades in med en Arduino och sedan visades på datorskärmen med ett program. Det i diskussionen omnämnda Pyxis OS visar att Arduino kan användas även i mer krävande applikationer. Genom arbetet med Arduino har jag själv fått idéer till nya intressanta byggprojekt där den kan användas.

# KÄLLOR


Arduino. 2010a, *Arduino – HomePage* [www]. Tillgänglig: http://www.arduino.cc Hämtad 13.4 2010.

Arduino 2010b, *Arduino – ArduinoBoardDuemilanove* [www]. Tillgänglig: http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove Hämtad 13.4 2010.

Arduino. 2010c, *Arduino – PWM* [www]. http://www.arduino.cc/en/Tutorial/PWM Hämtad 13.4 2010.

Atmel. 2009, *ATMega 328* [www]. Tillgänglig: http://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf Hämtad 13.4 2010.

Barr, Michael. 2001, *Introduction to Pulse Width Modulation* [www]. Tillgänglig: http://www.embedded.com/columns/beginerscorner/9900283?_requestid=109398 Hämtad 28.1 2010.

Clarke, Trevor. 2009, *The A-Z of Programming Languages: Arduino's Tom Igoe* [www]. ComputerWorld, publicerad 12.10 2009. Tillgänglig: http://www.computerworld.com.au/article/321749/a-z_programming_languages_arduino_tom_igoe/ Hämtad 14.3 2010.

Compulite. 2006, *Compulite – Vector Family* [www]. Tillgänglig: http://compulite.com/index.php?page_id=23 Hämtad 14.3 2010.

Dali AG. 2001, *Dali manual* [www]. Tillgänglig: http://www.dali-ag.org/c/manual_gb.pdf Hämtad 13.4 2010.

Djajadiningrat, Tom. 2004, *PIC Microcontroller Programming on Mac OS X* [www]. Tillgänglig: http://www.mactech.com/articles/mactech/Vol.20/20.02/PICMicrocontroller/index.html
Hämtad 14.3 2010.

Finlex. 2010, *Ordningslagen* [www]. Tillgänglig: http://www.finlex.fi/sv/laki/ajantasa/2003/20030612 Hämtad 21.3 2010.

Fremont Street Experience. 2007, *Facts* [www]. Tillgänglig: http://www.vegasexperience.com/#/about/ Hämtad 28.4 2010.

Fritzing. 2010, *Welcome – Fritzing* [www]. Tillgänglig: http://fritzing.org/welcome/ Hämtad 13.4 2010.

Gibb, Alicia. 2010, *New media art, design, and the Arduino microcontroller: A malleable tool* [www]. Tillgänglig: http://aliciagibb.com/thesis Hämtad 3.2 2010.

Haug, Eberhard. 2007, *LEDs Grundlagen* [www]. Tillgänglig: http://www.led-treiber.de/html/leds_grundlagen.html Hämtad 30.1 2010.

Howell, Wayne. 2002, *App Note 11:An overview of the electronic drive techniques for intensity control and colour mixing of low voltage light sources such as LEDs and LEPs* [www]. http://artisticlicence.com/WebSiteMaster/App Notes/appnote011.pdf Artistic Licence (UK) Ltd. Hämtad 19.12 2009.

Howell, Wayne. 2007, *What is Art-Net?* [www] Tillgänglig: http://artisticlicence.com/WebSiteMaster/App Notes/appnote013.pdf Artistic Licence (UK) Ltd. Hämtad 13.4 2010.

Jeppson, Kjell. 1984, *Halvledarteknik-komponenter och teknologi för integrerade kretsar*. Lund: Studentlitteratur. 408 s. ISBN 91-44-21101-5.

Maxim IC. 2001, *Guidelines for Proper Wiring of an RS-485 (TIA/EIA-485-A) Network* [www]. Tillgänglig: http://www.maxim-ic.com/app-notes/index.mvp/id/763 Hämtad 26.1 2010.

McKerrow, Phillip. 2007, *Software development of embedded systems on Macintosh* [www]. Tillgänglig: http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1705&context=infopapers Hämtad 31.10 2009.

McRoberts, Mike. 2009, *Arduino starters kit manual - a complete beginners guide to the Arduino* [www]. Tillgänglig: http://www.earthshinedesign.co.uk/ASKManual/Site/ASKManual.html Hämtad 1.12 2009.

Nelson, Todd. 1995, *The practical limits of RS 485* [www]. Tillgänglig: http://national.com/an/AN/AN-979.pdf National Semiconductor. Hämtad 13.4 2010.

Ness, Tomek & Cuartilles D. 2006, *[S]ending DMX with the Arduino* [www]. Tillgänglig: http://tomekness.files.wordpress.com/2007/01/dmx_and_arduino_tutorial.pdf Hämtad 13.4 2010.

OpenDMX. 2010, *Arduino RGB Mixer* [www]. Tillgänglig: http://opendmx.net/index.php/Arduino_RGB_Mixer Hämtad 22.2 2010.

Philips. 2008, *Introduction to DMX* [www]. Tillgänglig: http://www.illuminazione.philips.it/gb_en/architect/luminaire/controls/training/Introduction%20to%20DMX.pdf Hämtad 12.4 2010.

Processing. 2010, *Overview \ Processing.org* [www]. Tillgänglig: http://processing.org/about/ Hämtad 13.4 2010.

Salzberg, Jeffrey & Kupferman, Judy. 2009-2010, *Stage Lighting for Students* [www]. Tillgänglig: http://www.stagelightingprimer.com Hämtad 15.3 2010.

Shirriff, Ken. 2009, *Secrets of Arduino PWM* [www]. Tillgänglig: http://www.arcfn.com/2009/07/secrets-of-arduino-pwm.html Hämtad 12.3 2010.

Simpson, Robert. 2003, *Lighting Control - Technology and Applications*. Oxford, Burlington: Focal Press. 576s. ISBN 0 240 51566 8.

Siemens Aktiengesellschaft. 1990, *Halbleiter: Technische Erläuterungen und Kenndaten für Studierende*. Berlin, München: Siemens Aktiengesellschaft. 270 s. ISBN: 3-8009-1554-5.

Subramanian, Muthu; Schuurmans, Frank; Pashley, Michael. 2002, *Red, Green, and Blue LED based white light generation: Issues and control* [www]. Tillgänglig: http://focs.eng.uci.edu/papers%20for%20GaN/White%20LED/Red,%20Green,%20and%20Blue%20LED%20based%20white%20light.pdf Hämtad 12.4 2010.

Sukale, Martin. 2008, *Konstruktion eines Netzwerkes eingebetteter Systeme für interaktives Design* [www]. Diplomarbeit, Hamburg: Hochschule für Angewandte Wissenschaften Hamburg, Fakultät Technik und Informatik, Department Informatik. Tillgänglig: http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/diplom/sukale.pdf Hämtad 12.4 2010.

United States Institute for Theatre Technology. 2010, *DMX512 FAQ* [www]. Tillgänglig: http://www.usitt.org/DMX512FAQ.aspx Hämtad 17.1 2010.

Velleman nv. 2010, *RGB LED CONTROLLER WITH REMOTE CONTROL* [www]. Tillgänglig: http://www.velleman.eu/distributor/products/view/?id=380596 Hämtad 28.4 2010.

Viinamäki, Pasi. 2005, *Väliaikaisten valaistusjärjestelmien tekninen suunnittelu* [www]. Tutkintotyö, Tampere:Tampereen ammattikorkeakoulu, Viestinnän osasto. Tillgänglig:

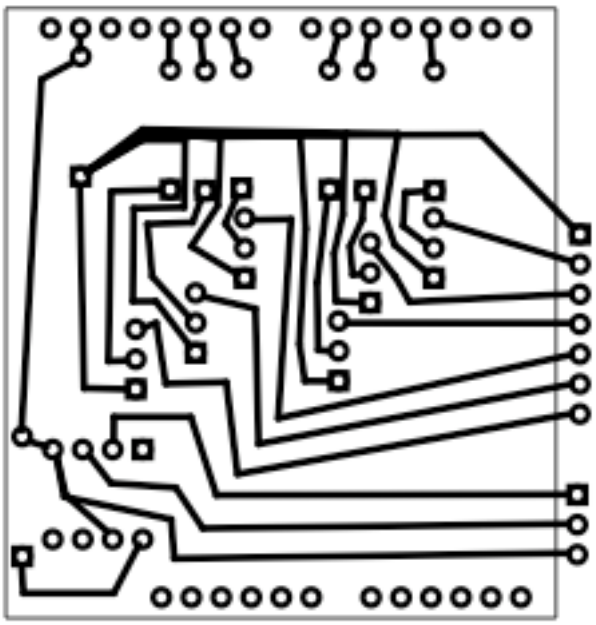
https://publications.theseus.fi/bitstream/handle/10024/10610/TMP.objres.372.pdf?sequence=2 Hämtad 4.4 2010.

Wikipedia. 2010a. *Arduino* [www]. Tillgänglig: http://en.wikipedia.org/w/index.php?title=Arduino&oldid=354108367 Hämtad 13.4 2010.

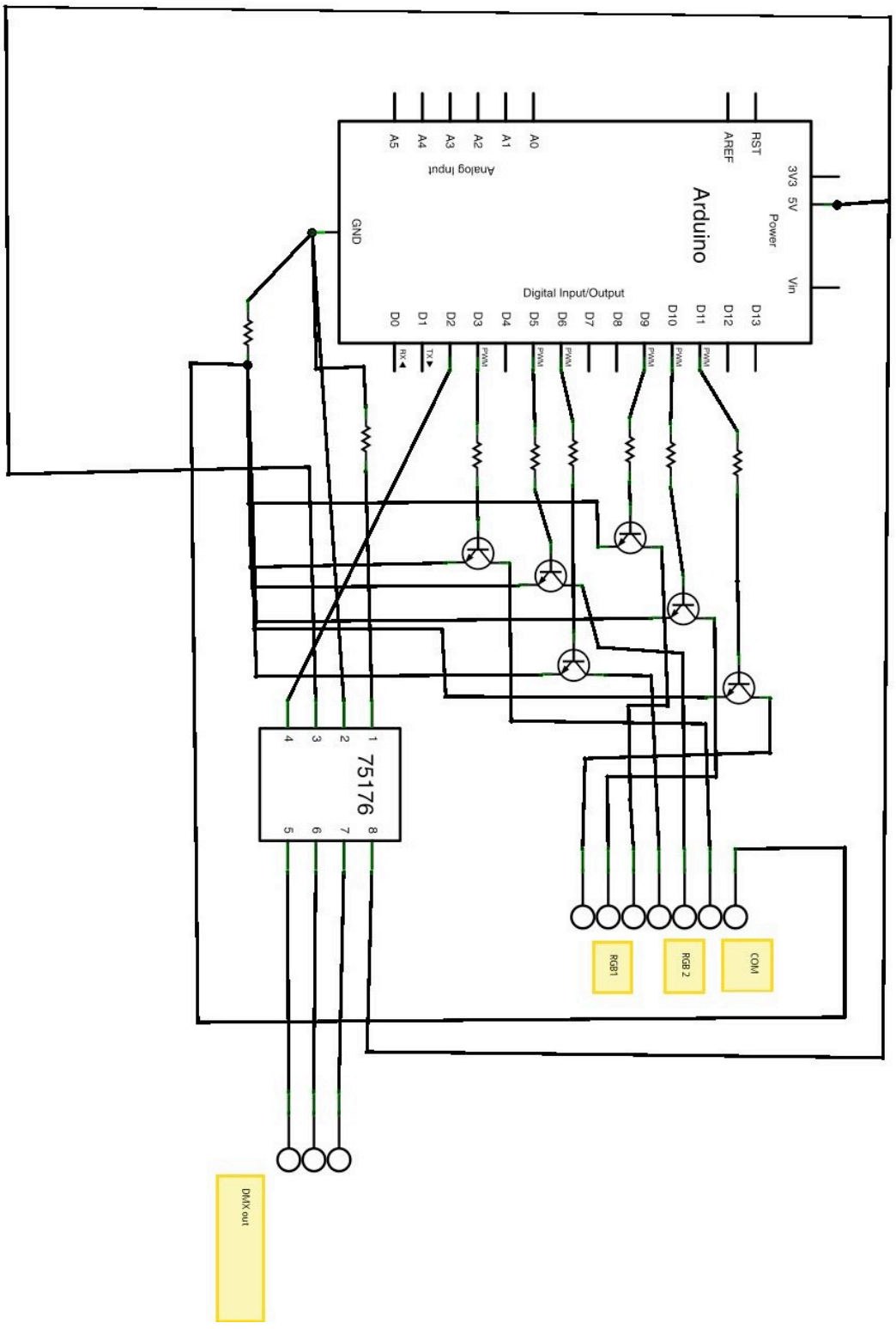
Wikipedia. 2010b. *Arduin of Italy* [www]. Tillgänglig: http://en.wikipedia.org/w/index.php?title=Arduin_of_Italy&oldid=349706494 Hämtad 13.4 2010.

## BILAGOR

Bilaga 1 - Kretskortsmodellen. Observera att komponenterna inte kommer på samma sida som lödningen och kopparbanorna. På grund av detta bör åtminstone de kopplingar som går till och från IC-kretsarna vara "spegelvända" eftersom dessa inte kan vändas lika lätt som transistorer och motstånd.



Bilaga 2 - Kretsschema



Bilaga 3 - Användargränssnittets källkod.

```
// SPECTRUM VERSION 0.6

// delete last has been removed

import processing.serial.*;
Serial port;
import interfascia.*;
// interface library, from http://www.superstable.net/interfascia/download.htm


// COLOR SPECTRUM AND INTERFASCIA BUTTONS


int u;
float rod;
float gron;
float bla;
int rosso;
int verde;
int blu;
GUIController c;
IFButton b1, b2, b3, b4, b5, b6;
IFRadioButton b7, b9, b11, b12;
IFButton b8, b10;
IFRadioController rc1;
IFRadioController rc2;
IFLabel l;
IFTextField terminalone;
IFTextField terminaltwo;
IFTextField terminalthree;
IFTextField terminalfour;
String one;
String two;
String three;
String four;
int numero;

//int[] tb; //CURRENT TABLE!
int[] time; //time for this session
int[] elapsed; // interlace?
int[] r; // target red
int[] g; // target green
int[] b; // target blue
int[] pwmdmx; // pwm or dmx
int[] sqhold; // sequence or hold
int[] add1; //address one, or from address
int[] add2; // to address (if !=null), get...
// and check upper...

// sequence always goes from current to target
// if you want to jump directly, make it as an event and set time to 0
// ======================
void setup() //begin
{
 size(500,500); //screen
 background(0);
 PFont font; // font for the terminal
```

```
font = loadFont("Serif-16.vlw");
textFont(font);

makecolor(); // make a spectrum
defineinput(); // make some buttons
createtable(); // make tables

drawTerminal("0"); // draw a terminal window at the bottom

 port = new Serial(this, Serial.list()[0], 9600); //serial

}
// ======================
void draw() // main - empty but nevertheless required
{

}
// ======================
void makecolor() // make the spectrum
{
  noStroke(); // this part comes from
//http://processing.org/reference/colorMode_.html
colorMode(HSB, 255); // set the color mode to Hue-Saturation-Brightness
int s = 255;
float mellan = 0;
for (int h = 0; h < 256; h++) { //hue, run up
  for (int b = 0; b < 256; b++) { // saturation, run up

  mellan = 256*log((256-b)/5);
  s = int(mellan);
    stroke(h, s, b); //swapping 2nd and 3rd parameter will make
    //the upper level light or dark
    point(h, b);

  }

}
}

void defineinput() // buttons and textfields using the interfascia library
{
  c = new GUIController (this); // controllers
  rc1 = new IFRadioController("Plex");
  rc2 = new IFRadioController("Keeper");
b1 = new IFButton("Route", 285, 10, 60, 20); // buttons...
b2 = new IFButton("Load .txt", 285, 40, 60, 20);
b3 = new IFButton("Save .txt", 350, 40, 60, 20);
b6 = new IFButton("Write Flash", 350, 10, 60, 20);
b4 = new IFButton("Repeat", 1, 315, 255, 20);
b5 = new IFButton("NEXT", 1, 285, 255, 20);
b7 = new IFRadioButton("", 285, 240, rc1);
b11 = new IFRadioButton("", 285, 260,rc1);
b8 = new IFButton("Undo last", 285, 300, 60, 20);
b10 = new IFButton("Start over", 360, 300, 60, 20);
b9 = new IFRadioButton("", 400, 240, rc2);
b12 = new IFRadioButton("", 400, 260, rc2);



terminalone = new IFTextField("time", 340, 70, 150);
terminaltwo = new IFTextField("elapsedtime", 340, 110, 150);
terminalthree = new IFTextField("address1", 340, 150, 150);
terminalfour = new IFTextField("address2", 340, 190,150);
```

```
fill(255);
//textFont(font, 12);
text("Duration", 260,85);
text("Interlace",260,125);
text("Address 1",260,165);
text("Address 2",260,205);
text("DMX",305,255);
text("PWM",305,275);
text("Hold",420,255);
text("Sequence",420,275);

b1.addActionListener(this); //buttons
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
b5.addActionListener(this);
b6.addActionListener(this);
b8.addActionListener(this);
b10.addActionListener(this);

b7.addActionListener(this); // radio PWM or DMX?
b11.addActionListener(this);

b9.addActionListener(this); // radio hold or sequence?
b12.addActionListener(this);

terminalone.addActionListener(this); //text boxes
terminaltwo.addActionListener(this);
terminalthree.addActionListener(this);
terminalfour.addActionListener(this);

 c.add (b1);
  c.add (b2);
   c.add (b3);
    c.add (b4);
     c.add (b5);
      c.add (b6);
      c.add (b7);
     c.add (b8);
      c.add (b9);
      c.add (b10);
      c.add (b11);
      c.add (b12);
      c.add(terminalone);
      c.add(terminaltwo);
      c.add(terminalthree);
      c.add(terminalfour);
}
// COLORS...

void mousePressed(){ // press the mouse to go here
if(mouseX<256 && mouseY<256) //just the values of the color table
{
  u = get(mouseX,mouseY); // get the variable

  rod = red(u); // split it up into red
  gron = green(u); // green
  bla = blue(u); // and blue
   rosso = int(rod); // we gotta make the floats to ints
  verde = int(gron);
  blu = int(bla);
  println("R: " + rosso + ", G: " + verde + ", B:" + blu); // print into console
```

```
  }

}
// BUTTON ACTION
 void actionPerformed (GUIEvent e) {
  if (e.getSource() == b1) {
    println("route");
    send(0);
    // call serial with current table, send message

  } else if (e.getSource() == b2) {
    println("load");
    loadSomething();
    drawTerminal("File loaded");
  }
  else if (e.getSource() == b3) {
    println("save");
    saveSomething();
    drawTerminal("File saved");
  }
  else if (e.getSource() == b4) {
    println("repeat");
    copyEvent();
  }
  else if (e.getSource() == b5) {
    println("NEXT"); // read the panel input
    readInstruments();
  }
  else if (e.getSource() == b6) {
    println("writeflash");
    send(1);
     // call serial with current table, send message, memorize == true

  }
  else if (e.getSource() == b8) {
    println("Undo last"); // undo last
 // This feature didn't work but created problems. Removed for now.

  }
  else if (e.getSource() == b10) {
    println("Start over");
    createtable();
  }
  else if (e.getSource() == terminalone) {
     if (e.getMessage().equals("Completed")) {
       //if this isn't here the program continuously reads the input
    one = terminalone.getValue();
    if(one !=null) println("T1: " + one); }
    //terminal 1

  }else if (e.getSource() == terminaltwo) {
 if (e.getMessage().equals("Completed")) {
    two = terminaltwo.getValue();
    if(two !=null) println("T2: " + two); }
    //terminal 2

  }else if (e.getSource() == terminalthree) {
     if (e.getMessage().equals("Completed")) {
    three = terminalthree.getValue();
    if(three !=null) println("T3: " + three);
     }
    //terminal 3
```

```
  }
  else if (e.getSource() == terminalfour) {
     if (e.getMessage().equals("Completed")) {
    four = terminalfour.getValue();
    if(four !=null) println("T4: " + four);
     }
    //terminal 4
  }
}
// ============ ARRAYS
void createtable() // new tables
{
 time = new int[0]; // and as tables
elapsed = new int[0];
r = new int[0];
g =  new int[0];
b = new int[0];
pwmdmx = new int[0];
sqhold = new int[0];
add1 = new int[0];
add2 = new int[0];

}

// ======================
void saveSomething() //THE SAVE FUNCTION
{

   String[] material = new String[r.length]; // make a NEW string table
  //with the length of the table named x
  for (int i = 0; i < r.length; i++) { // drive i up to the length value of x
     material[i] = time[i] + "\t" + elapsed[i] + "\t" + r[i] + "\t" + g[i] + "\t" + b[i] + "\t" + pwmdmx[i] + "\t"
+ sqhold[i] + "\t" + add1[i] + "\t" + add2[i]; //writing INTO the lines table
  }
  String path; // path instance

  path = selectOutput(""); // select the output
  if(path != null)
  {
  saveStrings(path , material);
  numero = 0;
  }
}
// ======================
void loadSomething() //THE LOADING FUNCTION
{
  String[] check;
  String path;
  int t;
  int elap;
  int dmx;
  int hold;
  int a1;
  int a2;


  //int index = 0;
  path = selectInput(""); // name of the input
  check = loadStrings(path); //all the data
  createtable(); //new table

  for (int index = 0; index < check.length; index++)
```

```
  {
   String[] delar = split(check[index], '\t');  //splitter
   if(delar.length == 9)
   {
    //make everything to variables and then append?
   t = int(delar[0]) ;
   time = append(time,t);

   elap = int(delar[1]) ;
   elapsed = append(elapsed,elap);

   rosso = int(delar[2]);
   r = append(r,rosso);

   verde = int(delar[3]);
   g = append(g,verde);

   blu = int(delar[4]);
   b = append(b,blu);

  dmx = int(delar[5]);
  pwmdmx = append(pwmdmx,dmx);

  hold = int(delar[6]);
  sqhold = append(sqhold, hold);

  a1 = int(delar[7]);
  add1 = append(add1, a1);

  a2 = int(delar[8]);
  add2 = append(add2, a2);

   }



  }
}
// ======================

void readInstruments() //read the parameters from the screen

{
 if(b9.isSelected()) // if the color should be kept, first move there in 0 time
 {
 time = append(time,0);
 elapsed = append(elapsed,0);
 r = append(r, rosso); //array, variable
 g = append(g, verde);
 b = append(b, blu);
 pwmdmx = append(pwmdmx,int(b7.isSelected()));
 sqhold = append(sqhold,int(b9.isSelected()));
  add2 = append(add2,int(terminalfour.getValue()));
 if(b7.isSelected()) // if dmx is chosen
 {
 add1 = append(add1,int(terminalthree.getValue()));
 }
 else if(!b7.isSelected())  // if pwm is chosen
 {
  add1= append(add1,0);
 }

 }
```

```
  time = append(time,int(terminalone.getValue()));
  elapsed = append(elapsed,int(terminaltwo.getValue()));
  r = append(r, rosso); //array, variable
  g = append(g, verde);
  b = append(b, blu);
  pwmdmx = append(pwmdmx,int(b7.isSelected()));
  sqhold = append(sqhold,int(b9.isSelected()));
  add2 = append(add2,int(terminalfour.getValue()));
   if(b7.isSelected()) // if dmx is chosen
  {
  add1 = append(add1,int(terminalthree.getValue()));
  }
  else if(!b7.isSelected())  // if pwm is chosen
  {
    add1= append(add1,0);
  }

  drawTerminal("0");
}

void copyEvent() // COPY THE LAST EVENT
{
  int i=time.length;

 // instantly jump to the initial position of last event
  time = append(time,0);
  elapsed = append(elapsed,0);
  r = append(r, r[i-2]); //array, variable
  g = append(g, g[i-2]);
  b = append(b, b[i-2]);
  pwmdmx = append(pwmdmx,pwmdmx[i-2]); // need to be here not to loose the count
  sqhold = append(sqhold,sqhold[i-2]);
  add2 = append(add2,add2[i-1]); // concerning the last addresses
  add1 = append(add1,add1[i-1]);

  //and, do the event anew
  time = append(time,time[i-1]);
  elapsed = append(elapsed,elapsed[i-1]);
  r = append(r, r[i-1]); //array, variable
  g = append(g, g[i-1]);
  b = append(b, b[i-1]);
  pwmdmx = append(pwmdmx,pwmdmx[i-1]);
  sqhold = append(sqhold,sqhold[i-1]);
  add2 = append(add2,add2[i-1]);
  add1 = append(add1,add1[i-1]);

  drawTerminal("0");
}
// ======================
//THE COMMUNICATION FUNCTION
// ======================

void send(int typ)
{
  int lng=time.length; // number of events
  int sent=0;
  // OPEN UP THE COMMUNICATION
  for(int i=0; i<10; i++)
  {
    if(typ==1) port.write("w"+lng); // send out data and wait
    else if(typ==0) port.write("o");
    delay(500);
```

```
   if(port.available()>0) // if the device responds
   {
    sent=1;
   }
  }
  // if the device doesn't respond
  if(sent==0) drawTerminal("COMMUNICATION ERROR, RECONNECT DEVICE");
  if(sent==0) return;


 drawTerminal("Sent write");
  for(int p=0; p<lng; p++) //from zero to length
  {
   // int tm=p/lng;
        port.write("t"+time[p]+ ' ' + "l"+elapsed[p]+ ' ' + "r"+r[p]+ ' ' + "g"+g[p]+ ' ' +"b"+b[p]+ ' ' +
"a"+add1[p]+ ' ' +"d"+add2[p]); // transmission
    delay(100); // maximal amount of events will take 14.6 seconds to transmit
   // drawTerminal(tm+"%");
  }
 drawTerminal("Data sent");
  delay(1000); // just in case
  port.write("q"); // quit command
 drawTerminal("Connection terminated");
}


// ======================
void drawTerminal(String msg) // make a terminal and print some text
{
  int i = time.length;
  int av = 146-i;
  //Tables HAVE to be printed
  //println("i: " + i);
  fill(255);
  rect(1,350, 498, 100);

  fill(0);
if (msg.equals("0")) //new
  {
   if(i !=0)
   {
      text("Added: " + time[i-1] + " " + elapsed[i-1] + " " + r[i-1] + " " + g[i-1] + " " + b[i-1] + " " +
pwmdmx[i-1] + " " + sqhold[i-1] + " " + add1[i-1] + " " + add2[i-1] + "  " + "Available: " + av, 10, 365);
   if(i>1)
   {
    fill(127);
      text("Added: " + time[i-2] + " " + elapsed[i-2] + " " + r[i-2] + " " + g[i-2] + " " + b[i-2] + " " +
pwmdmx[i-2] + " " + sqhold[i-2] + " " + add1[i-2] + " " + add2[i-2] + " ", 10, 395);
   }
   if(i>2)
   {
          text("Added: " + time[i-3] + " " + elapsed[i-3] + " " + r[i-3] + " " + g[i-3] + " " + b[i-3] + " " +
pwmdmx[i-3] + " " + sqhold[i-3] + " " + add1[i-3] + " " + add2[i-3] + " ", 10, 425);
   }
 }
  }
 else // another message
  {
   fill(255,0 ,0);
   text(msg, 10, 365);
 }

}
```

# Bilaga 4 - Systemplattformens källkod

```
#include <EEPROM.h> // required for EEPROM

#include "pins_arduino.h" //required for DMX

// colorcontroller 0.5


char buffer[42]; // buffer for the received data, sized for the worst case
int red,green,blue; // a variable for each channel

// ATTENTION!!!!!
int rPin = 11; // define pins
int gPin = 10;
int bPin = 9;

int rPin2 = 3; // second pwm port
int gPin2 = 5;
int bPin2 = 6;

int linePin=13; // communication indicator

// concerning DMX
int sig = 2; //DMX signal pin
int value = 0;
int valueadd = 3;
byte dmxChannel[256]; // the current value of a certain channel
byte channelReference[256]; // channel reference start counting from these


// arrays and indexes for them
byte r[146]; // arrays for the variables
byte g[146];
byte b[146];
byte t[146];
byte l[146];
byte a[146];
byte d[146];
int ri; // counters for the arrays, use separate ones just in case
int gi;
int bi;
int ti;
int li;
int ai;
int di;
byte line=0; // communication established or not?

//reference values
byte r1r=0; //1st port, beginning with reference one red
byte r1g=0;
byte r1b=0;

byte r2r=0; //2nd port
byte r2g=0;
byte r2b=0;
```

```
int h=0; // write to this EEPROM address
int w=0; // write or don't write?
int lng; // number of events
int mk=10; // delay in milliseconds keep TEN
int position=0;

void setup() // SETUP
{
 pinMode(linePin,OUTPUT); // serial communication?
 pinMode(rPin,OUTPUT); // if we'll go digital these are needed
 pinMode(gPin,OUTPUT);
 pinMode(bPin,OUTPUT);
 pinMode(sig, OUTPUT);
 Serial.begin(9600);
 Serial.flush();
doTables(); // get the information from EEPROM
}

void loop()  // LOOP
{
 //Serial.println(lng);
//EECheck(); // check content
if(Serial.available()>0 && line==0) // if there's something new
 {
  line=1; // line open
  Serial.print("A"); //ready to receive
  checkinput();
}

 if(line==1) // TRANSMISSION. Send q to terminate
 {
  checkinput();
 }

 while(Serial.available() == 0 && line==0) // if Serial is unavailable
 { // loop the memory
   colorloop(position); // position defines event
   position++;
  // Serial.println(position); // for test purposes
   if(position==lng) position=0;
 }
}

void checkinput() //check input
{
  digitalWrite(linePin, HIGH);
   int index=0;
   delay(100);
   int numChar = Serial.available();
   if (numChar>42) {
     numChar=42;
   }

   while (numChar--) {
     buffer[index++] = Serial.read();
   }
  splitString(buffer);
}

void colorloop(int x) // COLOR ALGORITHM
{
 // time & target for now, interlace calculation later
// Serial.println(micros());
```

```
  int q = t[x] * mk; // total time for this very transition in "centiseconds"
   int sc = int(d[position]); // the switch case that didn't work

if(a[position]==NULL) // if 1st address field is empty, PWM
{

 if(sc==1) //port #1
{
  int rDiff;
  int gDiff;
  int bDiff;

  if(x !=0) rDiff = r[x] - r1r; //delta color...
  else if(x==0) rDiff = r[x]; // (special case)
  int rTransition = rDiff/q; //...per time unit

  if(x !=0) gDiff = g[x] - r1g;
  else if(x ==0) gDiff = g[x];
  int gTransition = gDiff/q;

  if(x !=0) bDiff = b[x] - r1b;
  else if(x==0) bDiff = b[x];
  int bTransition = bDiff/q;

  for(int i=0;i < q;i++) // during this particular transition
  {
    int rOut = (i*rTransition)+r1r;
    analogWrite(rPin,rOut);
    int gOut = (i*gTransition)+r1g;
    analogWrite(gPin,gOut);
    int bOut = (i*bTransition)+r1b;
    analogWrite(bPin,bOut);
   delay(mk);  // delay 100 ms => 10Hz resolution
  }
  r1r=r[x]; // set the current values as reference points
  r1g=g[x];
  r1b=b[x];
}

  else if(sc==2) //port #2
  {
  int rDiff;
  int gDiff;
  int bDiff;

  if(x !=0) rDiff = r[x] - r2r; //delta color...
  else if(x==0) rDiff = r[x]; // (special case)
  int rTransition = rDiff/q; //...per time unit

  if(x !=0) gDiff = g[x] - r2g;
  else if(x ==0) gDiff = g[x];
  int gTransition = gDiff/q;

  if(x !=0) bDiff = b[x] - r2b;
  else if(x==0) bDiff = b[x];
  int bTransition = bDiff/q;

  for(int i=0;i < q;i++) // during this particular transition
  {
    int rOut = (i*rTransition)+r2r;
    analogWrite(rPin2,rOut);
    int gOut = (i*gTransition)+r2g;
    analogWrite(gPin2,gOut);
```

```
    int bOut = (i*bTransition)+r2b;
    analogWrite(bPin2,bOut);
    delay(mk);  // delay 100 ms => 10Hz resolution

   }
   r2r=r[x]; // set the current values as reference points
   r2g=g[x];
   r2b=b[x];
   }

   else // "default" port
   {
   int rDiff;
   int gDiff;
   int bDiff;

   if(x !=0) rDiff = r[x] - r1r; //delta color...
   else if(x==0) rDiff = r[x]; // (special case)
   int rTransition = rDiff/q; //...per time unit

   if(x !=0) gDiff = g[x] - r1g;
   else if(x ==0) gDiff = g[x];
   int gTransition = gDiff/q;

   if(x !=0) bDiff = b[x] - r1b;
   else if(x==0) bDiff = b[x];
   int bTransition = bDiff/q;

   for(int i=0;i < q;i++) // during this particular transition
   {
    int rOut = (i*rTransition)+r1r;
    analogWrite(rPin,rOut);
    int gOut = (i*gTransition)+r1g;
    analogWrite(gPin,gOut);
    int bOut = (i*bTransition)+r1b;
    analogWrite(bPin,bOut);
    delay(mk);  // delay 100 ms => 10Hz resolution
   }
   r1r=r[x]; // set the current values as reference points
   r1g=g[x];
   r1b=b[x];
   }
}

/*else if(a[position] !=NULL) // if there IS an address do DMX
{
    for(int i=0;i < q;i++) // during this particular transition
   {
    int rOut = (i*rTransition)+r[x-1]; // get some values
    int gOut = (i*gTransition)+g[x-1];
    int bOut = (i*bTransition)+b[x-1];

   // dmxoutput(rOut,gOut,bOut); // dmx, the function recognizes the address
    delay(mk); // this 100 ms resolution is more than enough for DMX, but
               // keeping it saves us additional scaling calculations
               // works also well for interlace if processDMX also

   }
}*/
}


// SPLITSTRING => from McRobert's Arduino manual
```

```
void splitString(char* data) // split arrived data
{
  char* parameter;
  parameter = strtok (data, " , ");

  while (parameter != NULL) {
    setData(parameter); // set data, ie. check the content
    // and react
    parameter = strtok (NULL, " , ");
  }

  for (int x=0; x<42; x++) {
    buffer[x] = '\0' ;
  }
  Serial.flush();
}

// SET DATA
void setData(char* data)
{

  // LED SET is similar but contains more if((data[0] == 'w'))
  //to control lots of stuff

  //Serial.println(data[0]);


  // TIME, T
  if (data[0] == 't') {  // if the data arrived is of this type
    int tAns = strtol(data+1, NULL, 10);
    tAns = constrain(tAns,0,255);
    t[ti] == tAns; // save it into the array
    if(w==1) //and if it should be stored
    {
      EEPROM.write(h,tAns); // do it
      h++;
    }
    // delay(100);
    ti++;

  }
  // INTERLACE, L
  else if(data[0] == 'l')
  {
    int lAns = strtol(data+1, NULL, 10);
    lAns = constrain(lAns,0,255);
    l[li] == lAns;
    if(w==1)
    {
      EEPROM.write(h,lAns);
      h++;
    }
    //   delay(100);
    li++;
  }
  // RED, R
  else if (data[0] == 'r') {
    int rAns = strtol(data+1, NULL, 10);
    rAns = constrain(rAns,0,255);
    r[ri] == rAns;
    if(w==1)
    {
```

```
    EEPROM.write(h,rAns);
    h++;
   }


}
// GREEN, G
else if(data[0] == 'g')
{
  int gAns = strtol(data+1, NULL, 10);
  gAns = constrain(gAns,0,255);
  g[gi] == gAns;
  if(w==1)
  {
    EEPROM.write(h,gAns);
    h++;
  }

}
// BLUE, B
else if (data[0] == 'b')
{
  int bAns = strtol(data+1, NULL, 10);
  bAns = constrain(bAns,0,255);
  b[bi] == bAns;
  if(w==1)
  {
    EEPROM.write(h,bAns);
    h++;
  }


}
// ADDRESS1, A
else if (data[0] == 'a')
{
  int aAns = strtol(data+1, NULL, 10);
  aAns = constrain(aAns,0,255);
  a[ai] == aAns;
  if(w==1)
  {
    EEPROM.write(h,aAns);
    h++;
  }
  ai++;

}
// ADDRESS 2, D
else if (data[0] == 'd') {
  int dAns = strtol(data+1, NULL, 10);
  dAns = constrain(dAns,0,255);
  d[di] == dAns;
  if(w==1)
  {
    EEPROM.write(h,dAns);
    h++;
  }
  di++;

}
// WRITE, W
else if(data[0] == 'w') // write data to eeprom
{
```

```
    lng = strtol(data+1, NULL, 10);
    lng = constrain(lng,0,146);
    w=1; //set flag, NEVER FORGET TO DO THIS
    EEPROM.write(1023,lng); // write the length
    Serial.flush(); //ditch this and wait for the relevant data

  }
  // ROUTE, O
  else if (data[0] == 'o')
  {
    //just route, don't write
    Serial.flush(); //empty serial buffer and wait for the data
  }

  else if (data[0] == 'q')
  {
    line=0; // quit transmission
    digitalWrite(linePin,LOW); // and switch of the led too
    w=0; // ensure that write is zero the next time we begin
    Serial.flush();

  }


}

// TEST FUNCTION
void EECheck() // Print memory
{
   for(int u=0; u<lng; u++)
  {
  Serial.print(t[u], DEC);
  Serial.print(' ');
  Serial.print(l[u], DEC);
  Serial.print(' ');
  Serial.print(r[u], DEC);
  Serial.print(' ');
  Serial.print(g[u], DEC);
  Serial.print(' ');
  Serial.print(b[u], DEC);
  Serial.print(' ');
  Serial.print(a[u], DEC);
  Serial.print(' ');
  Serial.print(d[u], DEC);
  Serial.println(' ');
  }
  delay(10000);
}

void doTables() // format all arrays by calling this function
{
   lng = EEPROM.read(1023);
  int y=0;
    for(int x=0; x<lng; x++) //make arrays...
  {
    t[x] = EEPROM.read(y); //Time
    y++;
    l[x] = EEPROM.read(y); //interLace
    y++;
    r[x] = EEPROM.read(y); //Red
    y++;
    g[x] = EEPROM.read(y); //Green
    y++;
```

```
    b[x] = EEPROM.read(y); //Blue
    y++;
    a[x] = EEPROM.read(y); //Address 1
    y++;
    d[x] = EEPROM.read(y); //aDdress 2
    y++;
   }
}

void pwm(int erre, int gee, int bee) //function runs for 100 ms
{
Serial.print(erre, DEC);
Serial.print(" ");
Serial.print(gee, DEC);
Serial.print(" ");
Serial.println(bee, DEC);
  for(int i; i<33; i++) // one loop is 3000 us or 3 ms,  3*33=99ms
  {
  digitalWrite(rPin,HIGH);
  delayMicroseconds(erre);
  digitalWrite(rPin,LOW);
  delayMicroseconds(1000-erre); //cycle for red, 1000 us

  digitalWrite(gPin,HIGH);
  delayMicroseconds(gee);
  digitalWrite(gPin,LOW);
  delayMicroseconds(1000-gee); //cycle for green 1000 us

  digitalWrite(bPin,HIGH);
  delayMicroseconds(bee);
  digitalWrite(bPin,LOW);
  delayMicroseconds(1000-bee); // cycle for blue 1000 us
  }


}

/*

// THIS FOLLOWING IS SLIGHTLY MODIFIED FROM
// http://www.tinker.it/en/uploads/Products/Heartbeat_02.pde
/*void setDmxChannel(byte channelID, byte value) {
            if (channelID < 256)
                        dmxChannel[channelID] = value;
}

void shiftDmxOut(int pin, int theByte)
{
  int port_to_output[] = {
    NOT_A_PORT,
    NOT_A_PORT,
    _SFR_IO_ADDR(PORTB),
    _SFR_IO_ADDR(PORTC),
    _SFR_IO_ADDR(PORTD)
    };

    int portNumber = port_to_output[digitalPinToPort(pin)];
  int pinMask = digitalPinToBitMask(pin);

  // the first thing we do is to write te pin to high
  // it will be the mark between bytes. It may be also
  // high from before
  _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
```

```
  delayMicroseconds(10);

  // disable interrupts, otherwise the timer 0 overflow interrupt that
  // tracks milliseconds will make us delay longer than we want.
  cli();

  // DMX starts with a start-bit that must always be zero
  _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;

  // we need a delay of 4us (then one bit is transfered)
  // this seems more stable then using delayMicroseconds
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
  asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

  for (int i = 0; i < 8; i++)
  {
    if (theByte & 01)
    {
      _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;
    }
    else
    {
      _SFR_BYTE(_SFR_IO8(portNumber)) &= ~pinMask;
    }

    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");
    asm("nop\n nop\n nop\n nop\n nop\n nop\n nop\n nop\n");

    theByte >>= 1;
  }

  // the last thing we do is to write the pin to high
  // it will be the mark between bytes. (this break is have to be between 8 us and 1 sec)
  _SFR_BYTE(_SFR_IO8(portNumber)) |= pinMask;

  // reenable interrupts.
  sei();
}


void processDmx() {
        // sending the dmx signal

  // sending the break (the break can be between 88us and 1sec)
  digitalWrite(sig, LOW);

  delay(10);

  dmxChannel[0] = 0;

  for (int count = 0; count <= 256; count++)
```

```
 {
        if (count < 256)
        shiftDmxOut(sig, dmxChannel[count]);
        else
                shiftDmxOut(sig,0);
 }


 //sending the dmx signal end
}*/
```