UNIVERZA NA PRIMORSKEM

Fakulteta za matematiko, naravoslovje in informacijske tehnologije, Koper

Računalništvo – 1. stopnja

ALEKSANDAR TOŠIĆ

CENOVNO UGODNA NAPRAVA ZA IZBOLJŠANJE VEČPREDSTAVNOSTNIH VSEBIN

ZAKLJUČNA PROJEKTNA NALOGA

Mentor: doc. dr. PETER KOROŠEC

Koper, september 2011

UNIVERSITY OF PRIMORSKA
Faculty of Mathematics, Natural Sciences and Information Technologies
Koper

Computer Science – 1st degree

ALEKSANDAR TOŠIĆ

COST EFFECTIVE DEVICE FOR MULTIMEDIA ENHANCEMENT

FINAL PROJECT PAPER

Mentor: PETER KOROŠEC, PhD

Koper, September 2011

ZAHVALA

Doc. dr. Petru Koroščcu se zahvaljujem, da me je sprejel pod svoje mentorstvo in pa za strokovno svetovanje ter spodbudo, ki mi ju je nudil pri izdelavi projektne zaključne naloge. Največja zahvala gre staršem in punci, ki so mi stali ob strani vsa študijska leta ter me podpirali.

POVZETEK

V zadnjih letih je razvoj zabavne industrije prinesel precej inovacij, ki strmijo k boljši poosebitvi multimedijskih vsebin. Tako je izum 3D zvočnega sistema, kljub dvodimenzionalnemu prikazu vizualnih vsebin , pripomogel k zaznavanju pozicije posameznih objektov, ki oddajajo zvok. Na ta način pri ogledu filma dobimo občutek, da je govorec v filmu za nami, saj zvok oddajajo zadnji zvočniki. Trenutno aktualen izum je 3D televizija, ki s pomočjo posebnih očal, objektom v filmu doda globino. Pred kratkim je v komercialno rabo prišel Philipsov televizor Ambilight, posebnost katerega je, da s pomočjo svetlobe ustvarja z vsebino barvno usklajen ambient, torej ozadje zaslona na steni. Philipsov produkt, kljub daljši prodajni dobi, ni pritegnil zaslužene pozornosti. Glavni razlog je v tem, da je sistem vgrajen zgolj v Philipsove televizijske sprejemnike in ga posledično ni moč kupiti kot samostojen produkt, ki bi omogočal namestitev na zaslone televizijskih sprejemnikov drugih znamk. Zaključna projektna naloga opisuje postopek izdelave produkta za osebne računalnike, ki predstavlja Philipsovemu Ambilightu tako cenovno kot funkcijsko konkurenco. Opisuje pa tudi možnosti nadaljnjega razvoja in uporabe svetlobnih učinkov v večpredstavnostnih vsebinah.

Ključne besede: 3D zvočni sistem, 3D televizija, Ambilight, svetleča dioda, svetlobni učinki, večpredstavnostne vsebine

ABSTRACT

The entertainment technology is driven by the idea of making multimedia experience as realistic as possible. Despite the invention of 3D video and 3D sound systems the user experience can still be greatly enhanced. One of many ways is the new Philips product named Ambilight. Its main function is to create an ambient matching the color of the television display using light emmiting diodes. Ambilight didn't reach its market potential mainly because its embeded into Philips television sets therefore is not compatible with any other display device. The final project paper describes the process of making a complementary device to the existing Ambilight both cheeper and more functional. Also covered are posible future development and use of light effects in multimedia.

Keywords: 3D video, 3D sound, Ambilight, light emmiting diodes, light effects, multimedia

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

KAZALO ALGORITMOV

SEZNAM KRATIC

CPU: Central processing unit (centralna procesna enota)

USB: Universal Serial Bus (univerzalno serijsko vodilo)

IDE: Integrated Drive Electronics (vmesnik za povezovanje naprav v računalniku)

LED: Light-emitting diode (svetleča dioda)

RGB LED: Red-green-blue light emitting diode (rdeča-zeleno-modra svetleča dioda)

CMYK: Cyan-magenta-yellow-key (sinje–škrlatno–rumena ključna barva)

PWM: Pulse-width modulation (pulzno-širinska modulacija)

FET: Field-effect transistor (tranzistor na poljski pojav ali unipolarni tranzistor)

MOSFET: Metal–oxide–semiconductor field-effect transistor (kovinsko-oksidni tranzistor na poljski pojav)

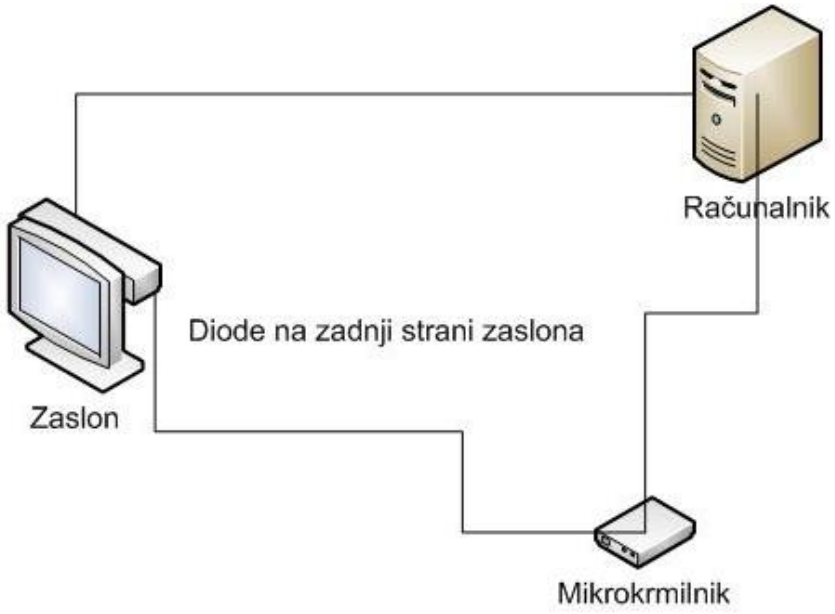
FPS: Frames per second (slika na sekundo)

1 UVOD

V zgodnjih 70-tih letih prejšnjega stoletja je glasbena skupina Pink Floyd med prvimi uporabljala svetlobne učinke, s katerimi je glasbi dodala tudi grafično podobo. Od tedaj se razni svetlobni učinki uporabljajo za boljše in bolj živahno doživljanje večpredstavnostnih vsebin. Elektronska glasba je dodatno pripomogla k razvoju svetlobnih učinkov, vendar jih v večini primerov uporablja za dezorientacijo množice poslušalcev [11]. Namen uporabe svetlobnih učinkov v večpredstavnostnih vsebinah pa je ravno obraten. Menjava barve svetlobe sinhrono s trenutnim prikazom na zaslonu, namreč gledalcu pomaga pri boljši predstavitvi in interpretaciji ambienta, predstavljenega na zaslonu. Delovanje naprave lahko programsko prilagajamo svojim potrebam, saj jo lahko uporabljamo za igranje računalniških iger, poslušanje glasbe, ogled filma ali branje. Tako na primer z belo svetlobo obsijana površina za zaslonom očem občutno zmanjša stres .

2 STROJNA REŠITEV

Tradicionalni osebni računalniki (za domačo rabo) ne omogočajo veliko možnosti upravljanja električnih komponent. Iz programskega vidika smo glede kontrole električnega toka, napetosti itd., precej omejeni, zanjo lahko uporabimo mikrokrmilnike. Na Sliki 2.1 vidimo osnovni koncept povezovanja računalnika, mikrokrmilnika in zaslona: mikrokrmilnik z računalnikom povežemo preko poljubnega standarda za komunikacijo med napravami, na računalniku se bo nato poganjal program, ki bo zajemal prikazano vsebino na zaslonu in iz nje izračunal povprečno barvo. Ta podatek bo preko vodila posredoval mikrokrmilniku, ki ga bo interpretiral in temu primerno spremenil stanje diod.



Slika 2.1: Model strojne izvedbe

2.1 Mikrokrmilnik

S pojmom mikrokrmilnik razumemo integrirano vezje s centralno procesno enoto (angl. Central Processing Unit, CPU), ustreznimi registri, oscilatorjem in ostalimi komponentami [15]. Na splošno imajo mikrokrmilniki skoraj vse glavne sestavine računalnika, vendar pa

so na račun nižjega urnega takta počasnejši pri izvajanju ukazov. Mikrokrmilniki so bili sprva namenjeni krmiljenju raznih sistemov, ki so bili običajno programirani v nizkonivojskih programskih jezikih, kot je na primer zbirni jezik (angl. assembly language), danes pa se uporabljajo tudi v visokonivojskih jezikih, kot je na primer programski jezik C [6].

2.1.1 Izbira primernega mikrokrmilnika

Pri izbiri mikrokrmilnika je potrebno jasno zastaviti parametre elementov, potrebnih za izdelavo naprave, saj lahko zaradi široke izbire mikrokrmilnikov prihranimo z izbiro in nakupom takega, ki bo naše potrebe ter zahteve ne le zadovoljil temveč jih tudi ne bo presegal.

Zato, da bi bila naprava kompatibilna z vsemi osebnimi računalniki, je potrebno zagotoviti uporabo uveljavljenega standarda za način komunikacije med mikrokrmilnikom in računalnikom. Prav zato smo izbrali mikrokrmilnik z vnaprej realiziranim vmesnikom za univerzalno zaporedno vodilo (angl. universal serial bus, USB), ki poleg komunikacije z računalnikom, omogoča tudi napajanje mikrokrmilnika z električno energijo. Po standardu USB mora gostitelj USB priklopa zagotoviti vsaj 500 mA toka in 5 V napetosti na vsak USB priklop. Mikrokrmilniki ne porabijo veliko električne energije, zato bomo poskusili napajati tudi diode s pomočjo USB vmesnika. Poleg zahteve po USB vmesniku, smo pri izbiri mikrokrmilnika upoštevali tudi njegovo razširjenost in priljubljenost. Slednja namreč močno vpliva na količino gradiva, ki je prosto dostopno na internetu. Zaradi lažjega reševanja morebitnih težav z mikrokrmilnikom in programom zanj, smo izbrali mikrokrmilnik s široko podporo po celem svetu in zelo aktivnim forumom.

Po pregledu ponudbe in razmisleku na osnovi zastavljenih parametrov, smo tudi z ozirom na ceno, izbrali mikrokrmilnik Arduino Duemilanove, prikazan na Sliki 2.2. Projekt Arduino se je pričel leta 2005 v Italiji in je odprtokodna razvijalna platforma. Osnovna ideja je bila izdelati cenovno ugodno platformo, predvsem za študente. Do februarja 2010 je bilo izdelanih in odposlanih že 120.000 mikrokrmilnikov. Programska jezika za programiranje Arduino mikrokrmilnikov sta programski jezik C in C++. Arduino ima tudi svoje prosto dostopno integrirano razvojno okolje (angl. integrated development environment, IDE), skupaj z mnogimi knjižnicami, ki znatno olajšajo naše delo. Poleg tega

je Arduino platforma dobro podprta s strani odprtokodnega projekta Processing, ki s pomočjo dodatnih knjižnic poenostavi delo z mikrokrmilnikom [13].

Slika 2.2: Mikrokrmilnik Arduino Duemilanove

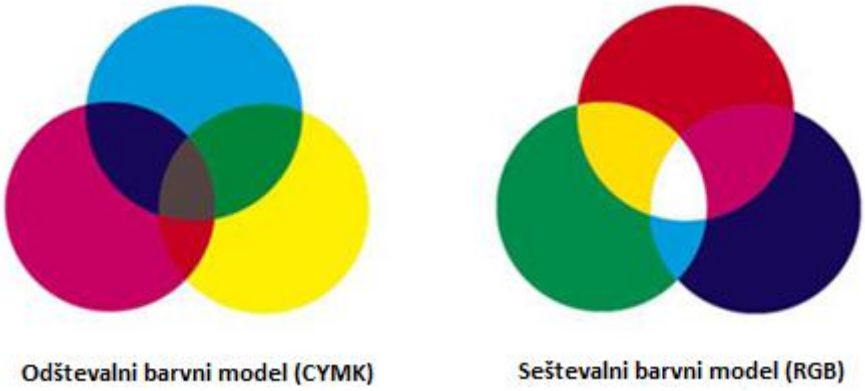
2.2 LED diode

LED je angleška kratica za "light emitting diode" [2]. Prvič so bile predstavljene leta 1962 kot praktična elektronska komponenta. Na začetku razvoja so oddajale zelo malo rdeče svetlobe, danes pa so na voljo tudi ultravijolične in infrardeče valovne dolžine. V osnovi strukturi so to polprevodne naprave, ki proizvajajo svetlobo pri prehodu električnih nabojev preko silicijevega spoja. Njihova življenjska doba je tudi do petkrat daljša od tradicionalnih žarnic – zaradi tega in zaradi svoje majhnosti, so danes nepogrešljive v mnogih električnih napravah.

2.2.1 Barvni modeli

Barvni modeli so abstraktni modeli, ki opisujejo načine mešanja barv. Običajno barve predstavimo z n-tericami treh ali štirih celih števil, imenovanih barvne komponente. Z barvnimi modeli lahko v digitalnem svetu predstavimo posamezno barvo. Najbolj razširjen barvni model je RGB, njegova teorija pa temelji na človeški percepciji barv [4]. RGB je seštevalni barvni model za mešanje svetlobe in se pogosto uporablja pri predstavi digitalnih barv. Najtemnejšo barvo dobimo, če je intenziteta vseh treh žarkov najnižja, najsvetlejšo pa s polno intenziteto vseh treh žarkov. Za mešanje fizičnih barv se pogosto uporablja CMYK barvni model, ki spada med odštevalne barvne modele [17]. Pri

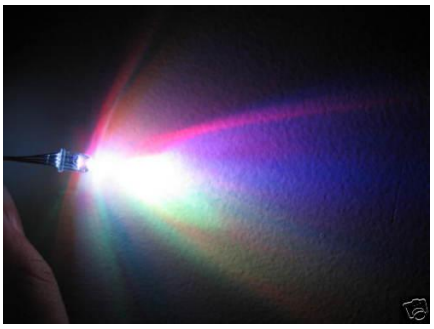
seštevalnih barvnih modelih se svetlobni žarki med seboj seštevajo, pri odštevalnih barvnih modelih pa se svetloba pri odboju absorbira, kar lahko vidimo na Sliki 2.3. Bel papir je bel zato, ker se vsa svetloba od njega odbija, pri rdečem papirju pa se večji del zelene in modre svetlobe absorbira, odbije se le rdeča svetloba.



Slika 2.3: Primerjava CMYK in RGB barvnega modela [14]

2.2.2 Barvne LED diode

Barvne LED diode (RGB LED) so diode, ki lahko oddajajo več barv. V sami osnovi delujejo enako kot navadne LED diode, le da je ena RGB LED dioda sestavljena iz treh navadnih. Vsaka od diod oddaja eno izmed osnovnih barv, t.j. rdečo, zeleno, ali modro. Te majhne diode so jedro RGB diode in so postavljene ena ob drugi. Zaradi njihove bližine se svetlobni spektri prekrivajo, kot lahko vidimo na Sliki 2.4. S pomočjo mešanja osnovnih treh barv lahko v odvisnosti od intenzitete posamezne barve, ustvarimo poljubno barvo.



Slika 2.4: Presek svetlobnih spektrov [8]

Običajno imajo RGB diode 4 nožice: ena omogoča ozemljitev, na ostale tri pa dovajamo električno energijo. S spremembo napetosti lahko spreminjamo moč posamezne barve. Za programsko spreminjanje napetosti uporabljamo pulzno - širinsko modulacijo.

2.2.3 Pulzno - širinska modulacija

V osnovi lahko s krmilnikom preklapljamo napetost med 0 in 5 V na posamezni nožici, imenovani izhodna nožica (angl. output pin) - če diodo priklopimo nanjo, bomo lahko samo bodisi prižgali bodisi ugasnili diodo. Potrebujemo torej način za upravljanje električne energije na izhodu s pomočjo mikrokrmilnika. To lahko počnemo s pulzno - širinsko modulacijo (angl. pulse width modulation, PWM) [16]. V preteklosti, ko je električna naprava potrebovala samo del celotne električne energije, je bila ta upravljana z zaporedno vezavo upora, na primer v obliki električne komponente, imenovane reostat. Delovanje reostata je zelo potratno, saj ne omogoča napajanja naprav z manj električne energije, kot jo imamo na voljo, ampak deluje le kot filter med napajanjem in napravo. Prav zaradi potrate električne energije se je porodila želja po učinkovitejši napravi. Tako se je leta 1960 pojavil PWM, prvič v ojačevalcu zvočnega signala, deluje pa tako, da stikalo s krmilnikom preklapljamo zelo hitro - dlje kot je stikalo vklopljeno, višja je električna energija na izhodu. Frekvenca, s katero preklapljamo napetost, je v veliki meri odvisna od naprave, ki jo uporablja. Frekvenca, s katero PWM preklaplja stikalo, mora biti namreč dovolj visoka, da ne vpliva na delovanje naprave, saj v nasprotnem primeru naprava nebi delovala pravilno, kar lahko privede tudi do okvare. V primeru, da bi PWM deloval na frekvenci 30 Hz, bi tako lahko s prostim očesom zaznali utripanje LED diod. PWM na Arduino mikrokrmilniku deluje na frekvenci 490Hz kar je dovolj za napajanje LED diod. Teorija PWM-ja je sicer veliko bolj kompleksna [5], za naše namene pa je dovolj razumevanje osnovnega principa delovanja, ki je bilo predstavljeno v tem poglavju.

2.3 Električno vezje

Električno vezje je sestavljeno iz električnih komponent (upori, tranzistorji, kondenzatorji, induktorji in diode), ki so povezane z žicami ali vodili, po katerih teče električni tok. V osnovi poznamo dva različna tipa električnih vezij: digitalna [10] in analogna [9]. Pri digitalnih vezjih električni signali predstavljajo numerične vrednosti. Najbolj uporabljena je binarna predstavitev, pri kateri ničelna napetost zavzame vrednost 0 in visoka napetost

vrednost 1. Digitalna vezja pogosto uporabljajo tranzistorje za realizacijo osnovnih funkcij Boolove algebre. Pri analognih vezjih se lahko skozi čas napetost in tok spreminjata, z namenom predstavitve informacije. Analogna vezja so zgrajena iz komponent, ki zaradi svojih fizikalnih omejitev, lahko proizvajajo naključne spremembe v električni energiji, te pa so vzrok izgube informacij. Analogna vezja delimo na vzporedna in zaporedna.

2.3.1 Primerjava različnih načinov vezave

Zaporedna vezava je vezava električnih komponent eno za drugo - primer take vezave so okrasne lučke. Pri tovrstni vezavi tok teče skozi vse električne komponente in v primeru izpada ene, z delovanjem prenehajo vse. Pri vzporedni vezavi pa so vse komponente priključene na isto napetost, tok pa je različen v odvisnosti od posamezne komponente. V primeru izpada ene izmed komponente, tako ostale še vedno delujejo. Opazovanje toka in upora pri obeh načinih pokaže, da pri zaporednih vezjih vsaka električna komponenta poviša upor na celotnem vezju in posledično tok pada. V našem primeru bi zaporedna vezava lahko pomenila, da bi zadnja RGB LED dioda prejemala manj toka in zato oddajala manj svetlobe. Zato bomo uporabili mešano vezavo in sestavili več manjših delov, v katerih bodo 3 RGB LED diode vezane zaporedno. Te dele bomo med seboj nato povezali vzporedno. V primeru, da se katera koli RGB LED dioda pokvari, bo z delovanjem prenehal le eden izmed delov, torej 3 RGB LED diode. Vsak del je tako iz vidika električnega vezja svoj element. S tem načinom vezave smo zmanjšali število elementov v vzporedni vezavi, zato tok ne bo upadel toliko, da bi vplivalo na svetilnost RGB LED diod.

2.3.2 Tranzistorji

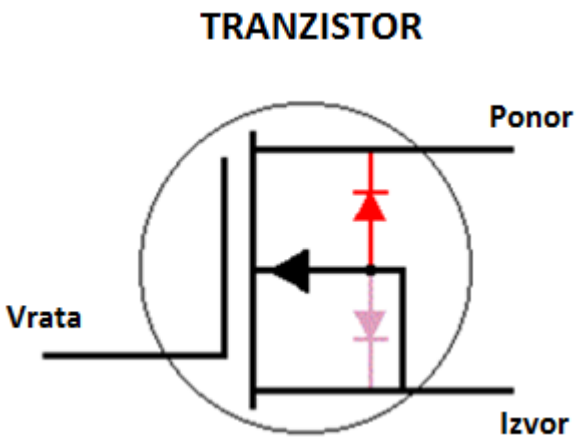
Tranzistorji so polprevodniški elementi, ki se uporabljajo v električnih napravah kot stikala in ojačevalci signala. Tranzistorje kot stikala vidimo predvsem pri digitalnih vezjih, medtem ko so tranzistorji za ojačitev signala bolj pogosti pri analognih vezjih. Poznamo dva tipa tranzistorjev: bipolarne tranzistorje, pri katerih ojačitev signala upravljamo s pomočjo toka [12] in tranzistorje na poljski pojav ali unipolarne tranzistorje (angl. field-effect transistor, FET), ki jih upravljamo s pomočjo napetosti [7].

2.3.2.1 Ojačitev signala

Pri izbiri mikrokrmilnika smo navedli, da ga bomo napajali preko USB vmesnika. Uporaba standardnih vmesnikov nam seveda pomaga pri uporabnosti naprave, v tem primeru pa nas

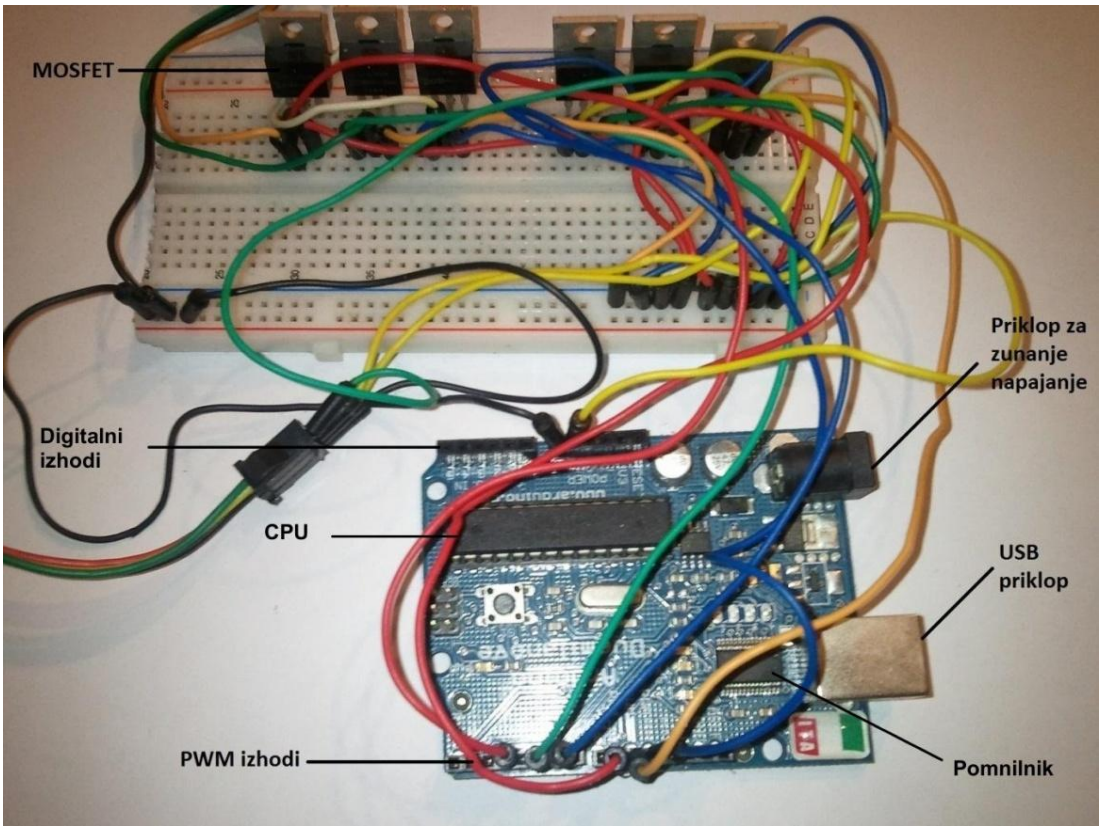
USB standard tudi omejuje. Mikrokrmilnik za svoje delovanje namreč resnično ne potrebuje celotne električne energije, ki jo nudi USB vmesnik in bo zato delovanje nemoteno, vendar bodo diode potrebovale veliko več električne energije. Količina potrebne električne energije je odvisna predvsem od količine diod. Poleg tega napajanje iz USB vmesnika nudi le 5 V napetosti, kar pomeni, da bomo omejeni tudi pri nakupu diod, ki obratujejo na 5 V napetosti. Prva zamisel je, da bi mikrokrmilnik napajali z večjo napetostjo, vendar žal to ni mogoče, saj so električne komponente, s pomočjo katerih je mikrokrmilnik realiziran, omejene na delovanje s 5 V napetostjo. Nastalo težavo lahko rešimo z ojačitvijo signala. Izhod PWM na mikrokrmilniku je namreč napetosti 5 V, mi pa potrebujemo 12 V, kar bomo rešili z uporabo tranzistorjev, natančneje s kovinsko-oksidnim tranzistorjem na poljski pojav (angl. metal–oxide–semiconductor field-effect transistor MOSFET, MOS-FET, oz. MOS FET) [1]. Delovanje MOSFET tranzistorja je za naše namene preobsežna tema, zato bomo predstavili le osnovo njegovega. Tipičen MOSFET tranzistor, kot ga vidimo na Sliki 2.5, ima tri nožice: izvor (angl. source), ponor (angl. drain), in vrata (angl. gate). Spremembo signala na izhodu krmilimo s pomočjo napetosti na vratih.

Za lažjo predstavo si zamislimo tranzistor, ki deluje na vodo. Naš izvor naj bo bazen napolnjen z vodo, vrata pa delujejo kot zamašen odtok - če zamašek odpremo, bo voda stekla proti ponoru, ki je v našem primeru odtočna cev. Količino vode,ki jo bomo spustili skozi odtočno cev, lahko nadziramo z odpiranjem zamaška.



Slika 2.5: Kovinsko-oksidni tranzistor na poljski pojav

Za ojačanje signala, potrebujemo dodaten vir električne energije. Neposredno napajanje z napajalnikom računalnika ni enostavno izvedljivo, oziroma je nevarno za povprečnega uporabnika računalnika, zato bomo morali električno energijo dovajati iz dodatne vtičnice. V ta namen smo s pomočjo spletne trgovine Ebay [19] kupili poceni napajalnik s standardnim evropskim priklopom za vtičnico, izhodno napetostjo 12 V in tokom 2 A. Za priklop zunanjega napajalnika se je izbrani mikrokrmilnik ponovno izkazal kot dobra izbira, saj omogoča možnost priklopa zunanjega napajanja. PWM izhod mikrokrmilnika bomo povezali z MOSFET-om na nožico, imenovano vrata. Napetost zunanjega napajanja povežemo z izvorno nožico, kot je prikazano na Sliki 2.6, tako bomo na ponorni nožici dobili ojačan signal, ki ga oddaja PWM.



Slika 2.6: Ojačitev signala s pomočjo tranzistorjev MOSFET

3 PROGRAMSKA REŠITEV

Prvi korak pri pisanju programa je izbira ustreznega programskega jezika. Zaradi omejenosti pri poznavanju in uporabnosti posameznih programskih jezikov, sta v ožji izbor prišla C/C++ in Java. Programski jezik C/C++ je primeren, predvsem zaradi omogočanja klicev sistemskih funkcij in programiranja na nižjem programskem nivoju od Javinega. Java pa je primerna zato, ker programe izvaja v virtualnem okolju, omogoča zagon in v večini primerih tudi izvajanje istega programa na različnih platformah ter operacijskih sistemih. S stališča sledenja začetni ideji o čim širši uporabnosti našega izdelka, je Java boljša izbira, saj bodo uporabniki napravo lahko koristili na različnih operacijskih sistemih za namizne računalnike, zato smo jo tudi izbrali kot programski jezik za programiranje aplikacije.

3.1 Program za mikrokrmilnik

Kot smo omenili v poglavju 2.1.1, ima Arduino aktivno skupnost in svoje razvojno okolje. Preden pričnemo s programiranjem, je torej potrebno naložiti in namestiti razvojno okolje, ki ga lahko z interneta snamemo na uradni spletni strani http://arduino.cc/en/Main/Software.

Program, ki ga napišemo za mikrokrmilnik, naložimo s pomočjo USB vmesnika, zato je potrebno naložiti in namestiti še gonilnike, ki bodo operacijskemu sistemu omogočili prepoznavanje mikrokrmilnika Arduino kot naprave. Tako razvojno okolje kot gonilniki so dostopni za vse najbolj uporabljane operacijske sisteme.

Mikrokrmilnik bo opravljal tri pomembne naloge: čakanje na podatke, njihovo sprejemanje in interpretacija. Na USB vmesniku bo tako čakal na morebitne podatke, ki jih bo pošiljal program, izvajan na računalniku. Ko bodo podatki poslani, jih bo moral mikrokrmilnik pravilno sprejeti, nato pa jih interpretirati in ustrezno spremeniti napetost na PMW izhodih.

Mikrokrmilnik bo tako sprejel tri vrednosti med 0 in 255, vsaka od njih predstavlja svetilnost posamezne barve na RGB diodah. Programsko PWM izhod upravljamo s pomočjo funkcije »analog.Write (*vrednost*, *nožica*)«, ki prejme dva parametra. Parameter *vrednost* predstavlja celoštevilsko vrednost od 0 do 255, ki predstavlja svetlost,parameter *nožica* pa referenco za enega izmed PWM izhodov. Natančnejše delovanje funkcije Write je prikazano v dokumentaciji, dostopni na uradni spletni strani http://www.arduino.cc/ [18]. CPU-ji na mikrokrmilnikih so v odnosu do CPU-jev v računalnikih relativno počasni, zato bomo zahtevnejše operacije prepustili aplikaciji, ki se bo izvajala na osebnem računalniku.

Algoritem 3.1: Psevdokoda algoritma za mikrokrmilnik

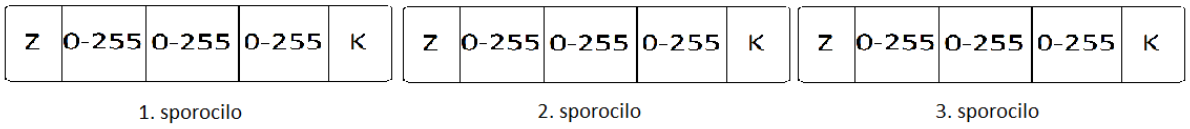
```
void loop ()
{
    if (Serial.read() >= 4)
    {
            //Preberi RGB vrednosti
            red = Serial.read();
            green = Serial.read();
            blue = Serial.read();
            //Nastavi izhodno napetost PWM izhoda
            analogWrite (RedPin, red);
            analogWrite (GreenPin, green);
            analogWrite (BluePin, blue);
      }
}
```

Funkcija »loop()«, ki je predstavljena v Algoritmu 3.1, se izvaja kot glavna zanka v mikrokrmilniku in najprej preveri, če so bili vsaj štirje biti poslani preko USB vmesnika. V kolikor je ta pogoj izpolnjen, predpostavimo, da je aplikacija, izvajana na računalniku, podatke poslala v pravilni obliki in da med pošiljanjem ni prišlo do napake, zato konsistence ter pravilnosti podatkov dodatno ne preverjamo in jih le zapišemo v spremenljivke, ki predstavljajo posamezno barvo RGB barvnega modela. S pomočjo prebranih vrednosti, ustrezno spremenimo napetost na PWM izhodih.

3.1.1 Sinhronizacija računalnika z mikrokrmilnikom

Razvojno okolje za Arduino mikrokrmilnike ima tudi vgrajen vmesnik za pošiljanje podatkov mikrokrmilniku, njegov namen pa je možnost razhroščevanja. Po nekaj

opravljenih testih je bila vidna razlika v barvah, ki so bile poslane in barvah svetlobe, ki so jo oddajale diode. Z razhroščevanjem smo odkrili, da se napaka skriva v sprejemanju podatkov iz USB vmesnika. Če smo poslali več sporočil zapored, je lahko prišlo do napačne interpretacije podatkov, saj mikrokrmilnik ni ločeval med posameznimi sporočili, ampak samo med vrednostmi. Tako se je pogosto dogajalo, da je mikrokrmilnik zadnjo izmed treh vrednosti v sporočilu prebral kot prvo vrednost naslednjega sporočila in obratno. Da bi se izognili tem težavam, vpeljemo dva kontrolna znaka, ki bosta označila začetek in konec posameznega sporočila. Začetek sporočila lahko označimo z znakom "Z" in konec sporočila z znakom "K", kot je razvidno iz Slike 3.1.



Slika 3.1: Zaporedno pošiljanje sporočil mikrokrmilniku

Popravimo program za mikrokrmilnik in dodamo dva pogoja, ki preverjata in skrbita, da se prebrani podatki začnejo z znakom "Z" ter končajo z znakom "K" - tako smo implementirali zelo enostaven komunikacijski protokol, ki ga lahko vidimo v Algoritmu 3.2.

Algoritem 3.2: Popravljena psevdokoda algoritma za mikrokrmilnik

```
void loop ()
{
    if (Serial.read() >= 4)
    {
        if(Serial.read() == 'Z' && Serial.read(3) == 'K'){
            //Preberi RGB vrednosti
            red = Serial.read();
            green = Serial.read();
            blue = Serial.read();
            //Nastavi izhodno napetost PWM izhoda
            analogWrite (RedPin, red);
            analogWrite (GreenPin, green);
            analogWrite (BluePin, blue);
        }
    }
}
```

3.2 Aplikacija za osebne računalnike

Aplikacija za osebne računalnike bo pred zagonom samega algoritma omogočala nastavitev nekaterih parametrov, s katerimi bo lahko uporabnik vplival na delovanje algoritma in prilagodil delovanje naprave svojim potrebam. Aplikacija bo komunicirala z mikrokrmilnikom s pošiljanjem podatkov preko USB vmesnika. Za njeno implementacijo bomo uporabili Java razred Serial in odprli zaporedna vrata (angl. serial port) na USB vmesniku. Zaporedna vrata so vmesnik, pri katerem se podatki med pošiljateljem in prejemnikom prenašajo v obe smeri, bit za bitom. V glavni zanki programa bo algoritem zajel sliko in izračunal njeno povprečno RGB vrednost ter jo nato poslal mikrokrmilniku preko zaporednih vrat.

3.2.1 Algoritem

Jedro algoritma je zajemanje vsebine na zaslonu, kolikor hitro je možno. Video vsebine na računalniku imajo sicer lahko različne hitrosti osveževanja slike, vendar je minimalna hitrost le nekaj nad 23 slik na sekundo (angl. frames per second, FPS). Za zaznavanje slike kot tekočega videa, zadostuje 23 FPS, predvajanje video vsebin z občutno manj FPS pa povzroči zaznavo posamezne osvežitve slike in posledično pokvari uporabniško izkušnjo. V kolikor bo naša aplikacija vsebino zajemala občutno počasneje od 23 FPS, bo mikrokrmilnik z istim intervalom nastavljal nove vrednosti diodam, kar lahko povzroči zakasnitve in neusklajenost delovanja svetlobnih učinkov ter video vsebine. Torej je naš cilj zajeti vsaj 23 slik na sekundo. Vsebino na zaslonu bomo zajemali s pomočjo Java razreda Robot, rezultat zajema pa je slika trenutne vsebine na zaslonu. Vhodno-izhodne operacije so zelo počasne, zato se jim bomo izognili in posamezno sliko shranili v instanco Java razreda BufferedImage za delo z digitalnimi slikami. BufferedImage objekt je sestavljen iz barvnega modela in slikovnih točk. Sedaj si oglejmo kaj digitalna slika sploh je - po osnovni opredelitvi je to numerično zapisana dvodimenzionalna slika. Digitalne slike delimo na rastrske [22] in vektorske [21]. Rastrske slike so zapisane s končno mnogo vrednostmi, imenovanimi slikovne točke (angl. pixels), ki so osnovni gradniki digitalne slike. Rastrske slike imajo končno mnogo slikovnih točk v vsaki vrstici in v vsakem stolpcu, torej predstavljajo dvodimenzionalno matriko, katere elementi so slikovne točke. Slednje so zelo majhne, zato veliko število točk vidimo kot sliko. V našem primeru sicer ne potrebujemo prosojnosti, vendar pa so same RGB vrednosti dobrodošle, saj

uporabljamo isti barvni model pri RGB diodah. Vektorske slike pa za predstavitev slike uporabljajo geometrijske primitive (točke, daljice, poligone itd.), zasnovane na matematičnih formulah, s katerimi zapišemo digitalno sliko. Velikost vektorskih slik je zato za razliko od rastrskih slik primerno manjša. Zapis digitalne slike kot matematične formule, pa nam na drugi strani omogoča povečanje ali pomanjšanje slike brez izgube kvalitete, saj pri rastrskih slikah pri povečavi izgubimo ostrino slike. V našem primeru potrebujemo le barve posameznih slikovnih točk, zato uporaba vektorskih slik ni primerna. Knjižnice za delo z vektorskimi slikami so namreč veliko bolj zahtevne, saj je za pridobivanje podatkov o sliki potreben izračun formul.

Za izračun povprečne barve na sliki, se bomo morali z zanko sprehoditi po matriki slikovnih točk in iz vsake slikovne točke prebrati RGB vrednosti, z njihovo pomočjo pa izračunati povprečje vseh vrednosti slikovnih točk, kot to lahko vidimo v Algoritmu 3.3. Za sprehod po matriki potrebujemo dve t.i. "*for*" zanki: s prvo se sprehajamo po stolpcih matrike, z drugo pa po vrsticah. Naš algoritem bo poleg posnetkov zaslona moral izračunati tudi povprečno RGB vrednost posameznega posnetka in vse to storiti vsaj 23-krat na sekundo. S pomočjo funkcije getRGB(*i,j*) dobimo RGB vrednost poljubne slikovne točke. Funkcija sprejme dva parametra: parameter i je številka vrstice, parameter j pa številka stolpca.

Algoritem 3.3: Psevdokoda algroritma za izračun povprečne RGB vrednosti

```
for i  to image.x {
    for j  to image.x {
        rgb = screenshot.getRGB(i,j)
        IzracunPovprecneVrednosti(rgb);
    }
}
```

Paziti moramo na hitrost izvajanja algoritma, zato poglejmo njegovo časovno zahtevnost. Časovna zahtevnost izračuna povprečne RGB vrednosti je $O\ n * m$, pri čemer n predstavlja širino in m višino slike. Velikost slike je enaka ločljivosti zaslona, širina zaslona pa je odvisna od višine zaslona, zato lahko pokažemo, da je časovna zahtevnost $O(n^2)$. Ker sta m in n odvisni spremenljivki, lahko m zapišemo kot $m = c * n$, pri čemer je c določena konstanta. Časovno zahtevnost lahko torej zapišemo kot $O(n * n * c)$ in jo brez konstante c zapišemo kot $O(n^2)$ [3].

Najenostavnejša rešitev bi imela zahtevnost $O(n^2)$, vendar pa se izkaže, da je izvajanje prepočasno za ogled video vsebin. Velikost posnetka zaslona enaka ločljivosti zaslona, zato smo lahko z manjšanjem števila slikovnih točk oziroma ločljivosti, zmanjšali časovno zahtevnost zajema slike in posledično izračuna povprečne RGB vrednosti. Algoritem smo testirali na različnih ločljivostih in dobili naslednje rezultate, prikazane v Tabeli 1.

Tabela 1: Rezultati prvega testiranja algoritma na različnih ločljivostih

Ločljivost [slikovnih točk]	Število slik na sekundo
800x600	13
1024x768	13
1280x1024	12
1600x900	12
1920x1080	11

Iz tabele je razvidno, da kljub temu, da je algoritem hitrejši pri nižjih ločljivostih, to samo po sebi ni rešilo problema. Večina današnjih zaslonov namreč prikazuje slike na visokih ločljivostih, zato ne moremo zahtevati od uporabnika, da zniža ločljivost z namenom uporabe naše naprave.

3.2.2 Optimizacija kode

Naš program je sestavljen iz dveh delov: zajem vsebine zaslona in izračun povprečne RGB vrednosti. Ločeno smo izmerili čas izvajanja obeh delov in ugotovili, da veliko program porabi več časa za zajem vsebine na zaslonu, kot za izračun povprečne barve. Na procesorju s frekvenco 3.4 GHz je program za enkratni zajem slike in izračun povprečne RGB vrednosti potreboval 90,75 ms, pri čemer je del programa za zajem slike potreboval ~68 ms (oziroma 75 % časa). V iskanju hitrejše metode za zajem vsebine na zaslonu, smo ugotovili, da so vse metode veliko bolj, torej preveč kompleksne, nekatere pa zahtevajo celo uporabo knjižnic, omejenih na specifičen operacijski sistem, kot je na primer knjižnica za grafiko Directx [20], ki jo uporabljajo operacijski sistemi Windows. Ker želimo, da bila

bi naša naprava kompatibilna na vseh pogosto uporabljenih operacijskih sistemih za namizne računalnike, smo se odločili vztrajati pri Java razredu Robot in poskusili optimizirati trenuten način.

Slika 3.2: Prikaz povečanega izbranega dela slike

Na levi strani Slike 3.2 vidimo podobo, ki jo je program zajel med predvajanjem filma. Če približamo sliko do te mere, da bodo vidne slikovne točke, opazimo, da v naključnem koščku slike pogosto nastopijo le odtenki ene barve, kot je razvidno iz desne strani Slike 3.2. Del slike, ki prikazuje približek enega dela slike, smo dali kot vhod našemu programu, ki izračuna povprečno barvo, pri izračunu pa upošteva vse slikovne točke. Program nam je vrnil polje vrednosti [125, 45, 35], pri čemer je prva vrednost količina rdeče, druga zelene in tretja modre barve. Za primerjavo smo iz slike 3.2 izbrali eno izmed slikovnih točk blizu sredine slike in njena RGB vrednost zapisana v polju je bila [129, 40, 32]. Opazili smo, da sta si barvi tako numerično kot grafično zelo podobni. Napaka je torej zelo majhna in je pri barvi svetlobe prostemu očesu neopazna. Ugotovili smo torej, da v kolikor pri računanju povprečne barve spustimo nekaj slikovnih točk, s tem ne izgubimo pomembnih vrednosti, ki bi lahko odločilno vplivale na povprečno RGB vrednost. Na ta način zmanjšamo količino podatkov, ki jih mora program obdelati in posledično pridobimo na času, potrebnem za obdelavo posamezne slike. V program vpeljemo novo spremenljivko imenovano odmik, ki bo določil število slikovnih točk, ki jih preskočimo pri računanju

povprečne RGB vrednosti. Ob nastavitvi spremenljivke na vrednost 5, pomeni, da bo algoritem za izračun uporabil vsako peto slikovno točko v sliki. Vrednost spremenljivke nastavimo v odvisnosti od ločljivosti zaslona in zmogljivosti procesorja na katerem poganjamo program. S pomočjo te spremenljivke lahko tudi uporabnik nastavi natančnost algoritma. Z uporabo tega pristopa seveda nismo pohitrili časa, potrebnega za zajem vsebine na zaslonu, ampak smo ugotovili, da ne potrebujemo celotne slike. Razred Robot nam omogoča tudi zajem delov slike, torej lahko zajamemo le dele, ki jih uporabljamo pri izračunu. Po optimizaciji algoritma smo ponovno testirali hitrost izvajanja algoritma s spremenljivko odmik, nastavljeno na 5 in dobili občutno boljše rezultate, ki so prikazani v Tabeli 2.

Tabela 2: Rezultati drugega testiranja algoritma po optimizaciji na različnih ločljivostih

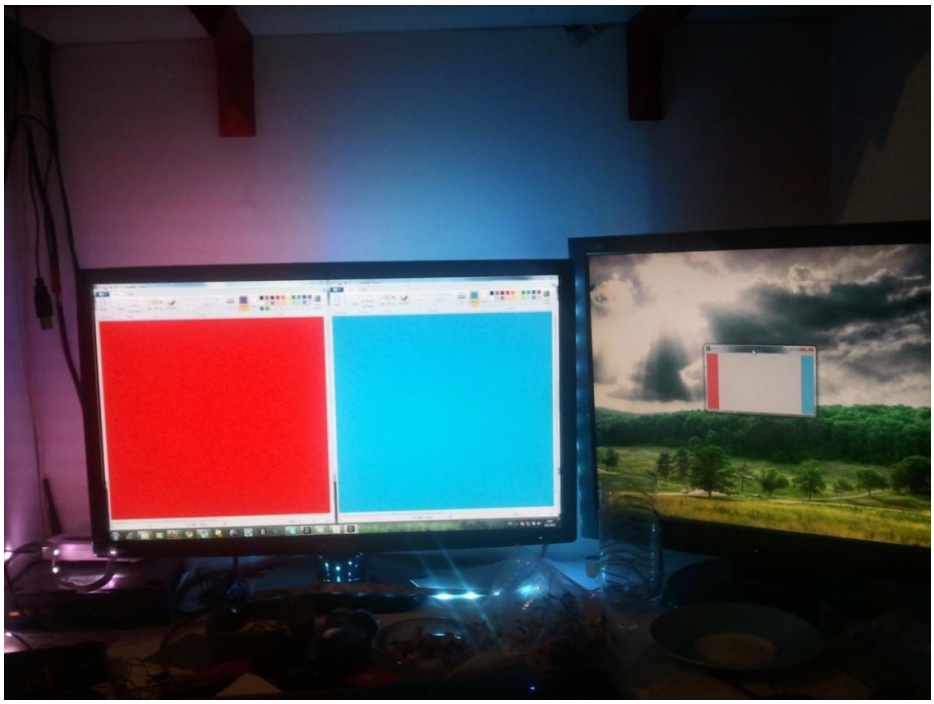
Ločljivost [slikovnih točk]	Število slik na sekundo
800x600	25
1024x768	25
1280x1024	24
1600x900	23
1920x1080	22

Ugotovili smo, da je optimizacija močno vplivala na hitrost zajema vsebine zaslona in posledično na odzivnost naprave brez izgube natančnost izračuna povprečne RGB vrednosti.

3.2.3 Dodatne funkcionalnosti

Pri ogledu poljubnega filma lahko opazimo, da je velikokrat dinamičen le določen del slike, medtem ko so ostali deli relativno statični. Najpogosteje do tega pride v akcijskih filmih, kjer se filmske scene menjajo v sekundnih intervalih. V primeru eksplozije avtomobila na levem delu zaslona, bi pri izračunu povprečne RGB vrednosti prevladal odtenek oranžne ali rdeče barve, kljub temu, da je eksplozija na levi strani slike, pa bi naprava osvetlila celotno ozadje zaslona. V izogib takim primerom, lahko sliko razdelimo na več delov in izračunamo povprečno RGB vrednost posameznega dela. Nadalje je

potrebno prilagoditi tudi program na mikrokrmilniku in vsakemu sporočilu po prvemu kontrolnemu znaku dodati še znak, s katerim označimo kateremu delu zaslona pripada sporočilo. Pripravimo še eno vezje, kot je opisano v poglavju 2.3, ga povežemo na še neizkoriščene PWM izhode na mikrokrmilniku in že lahko osvetlimo levo stran neodvisno od desne, kot je prikazano Sliki 3.3.



Slika 3.3: Ločeno osvetlitev levega in desnega ozadja zaslona

Taka naprava se najbolje izkaže pri ogledu filmov, vendar se filmi glede na barvno dinamiko med seboj razlikujejo, zato jih delimo v več kategorij. Tako so nekateri filmi barvno razigrani in zelo dinamični, drugi pa so zelo temačnih barv ter statični. Delovanje naprave se torej mora prilagoditi filmu, da bi lahko ustvarili ambient, ki je filmu komplementaren, kot je razvidno na Sliki 3.4. V ta namen programu dodamo dve spremenljivki: svetlost in intenzivnost. Spremenljivka svetlost bo povprečno RGB vrednost bodisi potemnila ali posvetila. V primeru, da je svetlost pozitivna, bo RGB vrednost svetlejša, v nasprotnem primeru pa temnejša. Intenzivnost pa bo določila, kako drastično naj se naprava odzove na nenadne spremembe barv. S nastavljanjem teh dveh spremenljivk uporabniku prepustimo izbiro kategorije filma.

Slika 3.4: Delovanje naprave pri ogledu filma

Uporabnik lahko izbira med tremi žanri: akcija, drama in srhljivka. Pri načinu akcija sta intenzivnost in svetilnost visoki, zato se naprava na vsako hitro spremembo močno odzove, kar je prikazano na Sliki 3.5. V načinu drama sta tako svetilnost kot intenzivnost nizkih vrednosti in naprava deluje podobno kot Philipsov Ambilight. Srhljivke so znane po temnih barvah in v srhljivih trenutkih kontrastno svetlih barvah, zato sta v tem načinu intenzivnost visoka ter svetilnost nizka, naprava pa večji del filma osvetljuje zelo malo. V srhljivih in nenadnih trenutkih se naprava zaradi nastavljene intenzivnosti drastično odzove ter nenadno osvetli prostor, s tem pa poveča kontrast med temnimi in svetlimi barvami.

Slika 3.5: Primer osvetlitve pri akcijskem filmu

3.2.4 Testiranje in primerjava zmogljivosti

Implementaciji aplikacije je sledila faza testiranja, znotraj katere smo delovanje aplikacije preverili na različnih operacijskih sistemih. Java je resnično izpolnila obljube in aplikacija se je brez večjih težav zagnala na vseh operacijskih sistemih, kljub temu pa so bili rezultati opazno različni, kot je prikazano v Tabeli 3. Na operacijskih sistemih Windows je aplikacija delovala gladko in v povprečju dosegla pričakovanih 23 FPS. Za testiranje delovanja na Linux operacijskemu sistemu smo uporabili distribucijo Ubuntu 10.04 in aplikacija je v povprečju dosegla 11 FPS. Prav tako občutno počasneje je aplikacija delovala na operacijskem sistemu MacOs. Razlog za tako različne rezultate je različna implementacija Java razreda Robot, ki ga uporabljamo za zajem vsebine na zaslonu. Rezultati za operacijska sistema Linux in Windows so med seboj primerljivi, saj so testi bili uporabljeni na istem računalniku, za operacijski sistem OSX pa je primerjava nekoliko težavna zaradi drugačne konfiguracije strojne opreme. OSX namreč poganjajo samo računalniki podjetja Apple, zato testiranje na istem računalniku (kot ostala dva operacijska sistema), ni bilo možno. Vsak operacijski sistem uporablja svoje knjižnice za izris grafične

vsebine in prav te razlike so razlog, zakaj je zajemanje vsebine na zaslonu tako različno. Ena od možnih rešitev je izdelava programa za posamezen operacijski sistem, kar bi omogočilo uporabo knjižnic in načinov za zajem vsebine na zaslonu, specifičnih za operacijski sistem. Na operacijskem sistemu Windows bi tako lahko uporabili knjižnico DirectX, ki je za grafične operacije že optimizirana s strani proizvajalca. Drugi razlog za slabo implementacijo razreda Robot je po vsej verjetnosti uporaba in interes samih razvijalcevopreme. Zajemanje vsebine zaslona namreč ni pogosto uporabljena funkcija in tudi redko zahteva tako odzivnost, kot smo jo potrebovali. Zato do sedaj tudi ni bilo potrebe po njeni optimizaciji.

Tabela 3: Rezultati testiranja aplikacije na različnih operacijskih sistemih

Operacijski sistem	Ločljivost [slikovnih točk]	Število slik na sekundo	CPU
Windows	1920x1080	23	3.4 GHz
Linux	1920x1080	11	3.4 GHz
OSX	1440x900	17	2.2 GHz

4 PRIMERI UPORABE

Naprava je sicer najbolj uporabna pri ogledu filmov, poleg tega pa so še druge možnosti uporabe. Napravo tako lahko uporabimo tudi pri igranju računalniških iger, kar lahko vidimo na sliki 4.1. Računalniške igre so danes zelo zahtevne za poganjanje, zato ni veliko neizkoriščenega procesorskega časa, ki bi ga lahko operacijski sistem namenil izvajanju naše aplikacije. Posledično se lahko nekatere igre počasneje izvajajo. Glede na to, da je delovanje aplikacije prilagodljivo, lahko zmanjšamo količino slikovnih točk, ki jih algoritem zajema in tako razbremenimo procesor. V primeru, da se računalniška igra izvaja počasi, lahko z zmanjšanjem količine slikovnih točk, ki jih algoritem zajema, dosežemo bolj tekoče izvajanje igre. Tako kot filmi, se tudi računalniške igre delijo na kategorije, zato je uporaba navedenih načinov delovanja, uporabna tudi pri igrah.

Slika 4.1: Svetlobni učinek pri igranju računalniške igre

Napravo lahko uporabimo tudi pri branju iz zaslona. Pri branju v temnih prostorih je, zaradi kontrasta med belo podlago in temnim ozadjem, stres za oči zelo velik. S pomočjo naprave lahko osvetlimo površino za zaslonom in s tem zmanjšamo kontrast ter stres. Branje je torej bolj prijetno, izognemo se razdraženosti in utrujenosti oči.

5 RAZŠIRITEV Z ZAJEMANJEM ZVOKA

Za razliko od konkurenčnega izdelka Ambilight, bomo v program vpeljali zvok. Večina večpredstavnostnih vsebin ima tudi zvočno podobo, ki je v veliki meri odvisna od grafične. Z analizo zvoka se lahko odločamo s kolikšno intenzivnostjo je primerno osvetliti ozadje. Idejo o zvoku razširimo na 3D, tako lahko na vsak zvočnik 3D zvočnega sistema napeljemo diode, z analizo posameznega izhodnega signala zvočne kartice pa ugotovimo pozicijo trenutno najbolj dinamičnega dogajanja v filmu. S pomočjo zvočnikov, opremljenih z diodami, lahko poleg površine za zaslonom, obarvamo celoten prostor, v katerem gledamo film. Če lahko predstavimo zvok v 3D obliki, lahko enako storimo tudi s svetlobnimi učinki. V primeru, ko v akcijskem filmu avtomobil eksplodira na desni strani, bi naprava zaznala najvišjo amplitudo na desnih zvočnikih in posledično obarvala desno steno v barvo eksplozije. Poleg tega bi lahko uporabili napravo tudi pri poslušanju glasbe, kjer bi barvo, ki jo diode oddajajo, določili s frekvenco in moč z amplitudo. Med poslušanjem glasbe bi bil tako prostor osvetljen v ritmu.

Zvočnike, na katerih bi bile diode, bi bilo sicer nekoliko nerodno povezati, saj bi bilo, poleg obstoječe napeljave, potrebno napeljati dodaten kabel za napajanje in kontrolo diod. Možna rešitev bi bila uporaba optičnih vlaken za prenos svetlobe iz naprave do zvočnikov. V tem primeru bi bile vse diode na napravi sami, z optičnimi vlakni pa bi svetlobne žarke poslali do ustreznega zvočnika.

6 ZAKLJUČEK

V zaključni nalogi je opisan postopek izdelave cenovno ugodne naprave za izboljšanje večpredstavnostnih vsebin. Želeli smo cenovno in funkcijsko konkurenčno napravo obstoječemu Philipsovemu Ambilightu. Naprava sestoji iz mikrokrmilnika, rdeče-zeleno-modrih svetlečih diod in aplikacije za izračun povprečne barve na zaslonu. Za izdelavo naprave smo izbrali mikrokrmilnik Arduino, aplikacijo pa implementirali v programskem jeziku Java. Pri izvedbi zaključne naloge smo naleteli na vrsto problemov. Prvi problem je bila ojačitev signala, ki smo jo rešili z dodatnim napajalnikom in tranzistorji. Naslednji problem je bilo počasno izvajanje aplikacije, rešili smo ga z zmanjšanjem količine podatkov za obdelavo. Nazadnje smo pri testiranju ugotovili, da se aplikacija počasi izvaja na operacijskih sistemih OSX in Linux. Zaradi slabe implementacije Java razreda Robot, težave ni bilo moč rešiti na enostaven način. Ena od možnih rešitev je izdelava namenske aplikacije za posamezen operacijski sistem.

Kljub temu, da naprava ni dosegla pričakovanih rezultatov, kar zadeva kompatibilnost, menimo, da je konkurenčna obstoječemu Ambilightu. Dosegli smo namreč ne le večjo funkcionalnost, saj se naprava lahko prilagaja različnim večpredstavnostnim vsebinam, temveč tudi veliko večjo intenzivnost, posledničnо pa resnično izboljšano uporabniško izkušnjo. V nadaljnjem razvoju bo potrebno veliko pozornosti vložiti v kompatibilnosti z ostalimi operacijskimi sistemi in dodajanju novih funkcionalnosti. Do morebitnega komercialnega produkta je pot še dolga, kljub temu pa se naprava že sedaj lahko kosa s konkurenčnim izdelkom Ambilightom, saj je trenutna proizvodna cena naprave v skupnem seštevku nižja od 50 evrov.

7 LITERATURA

[1] Vrej Barkhordarian, Power MOSFET Basics, International Rectifier, El Segundo, Ca.

[2] Duan Kelvin Seling, Light emitting diodes, (DEC 2, 2002)

[3] Donald Knuth, *The Art of Computer Programming*, 3. izdaja, Addison-Wesley, 1997, str. 107 – 123.

[4] Danny Pascale, A review of RGB color space ... from xyz to R'G'B', (2002 - 2003)

[5] Angel V. Peterchev, Digital Pulse-Width Modulation Control in Power Eletronic Cirucits: Theory and Applications, Harvard University, (1999)

[6] Yesu Thommandru, Programming a PIC Microcontroller, Iowa State University – ECpE, (2006)

[7] http://aries.ucsd.edu/najmabadi/CLASS/ECE60L/02-S/NOTES/FET.pdf

[8] http://cdn2.iofferphoto.com/img/item/162/653/980/uWwD.jpg

[9] http://en.wikipedia.org/wiki/Analog_electronics

[10] http://en.wikipedia.org/wiki/Digital_electronics

[11] http://en.wikipedia.org/wiki/Liquid_light_show

[12] http://picprojects.org.uk/projects/rgb2/

[13] http://processing.org/

[14] http://rosegraphics.biz/blog/tag/rgb-color/

[15] http://srinivasarao.webs.com/micro-cntrl.pdf

[16] http://webtools.delmarlearning.com/sample_chapters/04.pdf

[17] http://www.sketchpad.net/basics4.htm

[18] http://www.arduino.cc/

[19] http://www.ebay.com/

[20] http://www.d-silence.com/feature.php?id=254

[21] http://www.1stwebdesigner.com/design/pixel-vector-graphics-difference/

[22] http://www.tailrecursive.org/postscript/image.html

.