TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonika instituut

Mehhatroonikasüsteemide õppetool

MHT40LT

*Madis Saks*

# LED kuup reklaamimaterjali kuvamiseks

Bakalaureusetöö

Autor taotleb

tehnikateaduste bakalaureuse

akadeemilist kraadi

Tallinn
2014

# AUTORIDEKLARATSIOON

Deklareerin, et käesolev lõputöö on minu iseseisva töö tulemus.

Esitatud materjalide põhjal ei ole varem akadeemilist kraadi taotletud.

Töös kasutatud kõik teiste autorite materjalid on varustatud vastavate viidetega.

Töö valmis..... juhendamisel

“.....”.....201….a.

Töö autor

..... allkiri

Töö vastab bakalaureusetööle esitatavatele nõuetele.

“.....”.....201….a.

Juhendaja

..... allkiri

Lubatud kaitsmisele.

..... õppekava kaitsmiskomisjoni esimees

“.....”.....201… a.

..... allkiri

TTÜ mehhatroonika instituut
Mehhatroonikasüsteemide õppetool

# *BAKALAUREUSETÖÖ ÜLESANNE*

2014. aasta kevadsemester

Üliõpilane: Madis Saks, 112345MAHB
Õppekava: MAHB02/09
Eriala: Mehhatroonika
Juhendaja: Nooremteadur, Märt Juurma
Konsultandid: ei ole

**BAKALAUREUSETÖÖ TEEMA:**

(eesti keeles) LED kuup reklaamimaterjali kuvamiseks
(inglise keeles) LED cube for displaying advertising information

**Lõputöös lahendatavad ülesanded ja nende täitmise ajakava:**

| **Nr** | **Ülesande kirjeldus** | **Täitmise tähtaeg** |
|---|---|---|
| **1.** | Turu-uuring. Olemasolevate lahenduste analüüs. | **04.04.2014** |
| **2.** | Pistikute lahenduse projekteerimine | **07.04.2014** |
| **3.** | Elektroonika osa projekteerimine ja koostamine | **14.04.2014** |
| **5.** | Korpuse projekteerimine ja koostamine | **28.04.2014** |
| **6.** | Kontrolleri programmeerimine | **09.05.2014** |
| **7.** | Majanduslik kalkulatsioon, maksumushinnang | **16.05.2014** |

**Lahendatavad insenertehnilised ja majanduslikud probleemid**: Töö eesmärgiks on luua reklaami otstarbel kasutatav kergesti hooldatav ja laiendatav moodulitest koosnev LED kuubik. Projekteerida moodulite ühendus pistikute abil, sellele vastav elektroonika ning kaubandusliku välimusega korpus. Panna kuubik kokku ja luua näiteprogramm.

**Täiendavad märkused ja nõuded:** ei ole

**Töö keel:** eesti keel

Kaitsmistaotlus esitada hiljemalt 12.05.2014 **Töö esitamise tähtaeg:** 22.05.2014

**Üliõpilane:** Madis Saks /allkiri/ …….. kuupäev……

**Juhendaja:** Märt Juurma /allkiri/ ……. kuupäev……

Konfidentsiaalsusnõuded ja muud ettevõttepoolsed tingimused formuleeritakse pöördel.

**SISUKORD**

# EESSÕNA

Lõputöö teema valiti lähtuvalt üliõpilase isiklikust huvist antud valdkonna vastu. Teemat aitas täpsustada ja suunata juhendaja Märt Juurma.

# SISSEJUHATUS

LED kuup on kolmemõõtmeline valgusdioodide (edaspidi diood) maatriks. Kontrolleri ja elektroonika abil juhitakse iga dioodi individuaalselt ning seeläbi on võimalik moodustada silmatorkavaid valgusmustreid. Seega võiks kuupe silmapaistvuse tõttu kasutada reklaam-materjalide kuvamiseks. Võimalik on kuvada lihtsamaid logosid ning tähthaaval ka lühikesi loosungeid ja koduleheküIgede aadresse.

Ebay.com e-kaubamajas müügil olevad kuubid on üldjuhul kokkujootmata „*Do it yourself*" komponentide komplektid. Kokkujootmata kujul algavad 8x8x8 kuubi hinnad 62 eurost. Odavaim kokkupandud 8x8x8 kuup maksab juba 144 €. Internetis leidub samuti mitmeid õpetusi kuni 8x8x8 kuupide tegemiseks, mille komponentide hinnaks oleks umbes 50 eurot. Pakutavad variandid on kõik sarnase ülesehitusega. Elektroonika põhimõtte saab neilt suures osas üle võtta [1, 2], kuid turul leiduvate lahenduste probleemiks on nende raske hooldatavus. Dioodidevahelised ühendused on püsivalt kokku joodetud.

Dioodid riknevad üsna tihti ning seetõttu peab kuupi sageli hooldama. Kui dioodid on omavahel kokku joodetud, siis kuubi keskel olevate dioodide vahetamine on väga ebamugav ja aeganõudev. Sellest tulenevalt seati eesmärgiks projekteerida modulaarne kuup kasutades pistikuid. Modulaarne kuup võimaldaks rikkis dioodidele kergema vaevaga ligipääseda. Tekib ka võimalus katkine moodul uue vastu vahetada, kui üksiku dioodi asendamiseks ei ole käepärast tööriistu või aega. Moodulite abil on võimalik kuupi laiendada, kui ülejäänud elektroonika on selleks sobiv. Vastavate moodulite olemasolu korral on võimalik vahetada ka kuubi värvust.

Eesmärgiks oli leida lahendus, mis on mehhaaniliselt piisavalt stabiilne, kuna pistikud ei luba kasutada jäikasid vertikaalseid tugitraate, mille külge tavaliselt anoodid joodetakse. Välja tuli selgitada ka optimaalne moodulite suurus. Projekteeritava kuubi hind peaks jääma alla turul pakutavate lahenduste maksumusi.

Olemasoleva programmi põhjal sooviti luua kood, mis võimaldaks tähemärke ükshaaval kuvada.

Elektroonika projekteerimisel on kasutatud EAGLE 6.4.0 programmi. Kontroller programmeeriti Arduino tarkvara kasutades.

# 1. TEOORIA

Valgusdiood on pooljuhtseade, mis vajaliku suurusega päripinge rakendumisel hakkab kiirgama valgust. Dioodil on kaks viiku: anood ja katood. Dioodile rakendub päripinge juhul, kui anood on ühendatud katoodist kõrgema potentsiaaliga.

LED kuupide puhul ühendatakse igal tasandil olevate dioodide katoodid kokku ja samuti üksteise kohal (sammastel) olevate dioodide anoodid. Dioodi süttimiseks peab sammas olema ühendatud pingeallika plussklemmiga ja tasand miinusklemmiga. Selleks, et korraga tunduksid põlemas mitu dioodi erinevatel tasanditel ja sammastel, ilma et süttiks mõni teine nendega samal sambal olev diood, tuleb tasandeid sisse lülitada kordamööda. Korraga on võimalus põleda vaid ühe tasandi dioodidel. See vähendab ka maksimaalset kuubi poolt tarbitavat voolu. Paralleelselt lülitatakse ka anoode, et vastavalt igale tasandile põleksid soovitud dioodid. Seega kaheksa tasandi puhul oleks iga diood põlemas maksimaalselt 1/8 kogu ajast. Tasanditevaheline ümberlülitumine käib nii kiiresti, et silmaga ei ole võimalik vilkumist märgata. Veidi kannatab vaid valguse intensiivsus, kuid mitte oluliselt.

Kuubi põlemismustreid juhitakse kontrolleriga. Väiksemaid kuupe saab juhtida vaid kontrolleri väljundeid kasutades, kuid 8x8x8 kuubi juhtimiseks läheks sellisel juhul vaja 64+8 sisend/väljund viiku. Tavaliselt kontrolleritel nii palju väljundeid ei ole ja seega kasutatakse nihkeregistreid, mis teevad järjestikinfo paralleelkujul infoks. Iga taktsignaali ajal siseneb nihkeregistrisse 1 bitt infot ning registris ees olnud info nihkub 1 biti võrra edasi järgmisesse väljundisse. Nihkeregisterid on võimalik ühendada üksteisega jadamisi ja nõnda saada nii palju väljundeid kui vaja. Ainus probleem on aja kulu. 64 biti järjestikinfo paralleelkujule viimiseks kulub 64 taktsignaali pulssi.

Kontrollerist järjestikinfo saatmiseks kasutatakse SPI (ingl.k. *Serial Peripheral Interface*) protseduure. Kui SPI kontrollregister on vastavalt seadistatud, saab SPI andmeregistrisse (SPI *data register* -SPDR) kirjutamisega saata soovitud baite järjestikkujul välisseadmetesse (nihkeregistritesse). Seadistused tuleb teha nihkeregistritega vastavaks.

Et dioodide tarbitav voolutugevus, mis võib ulatuda üle 1 ampri, ei läbiks kontrollerit, kasutatakse lülitusteks transistore. Kontrolleriga juhitakse transistoride baase. Kui baasi voolutugevus on piisavalt suur, muutub transistori kollektori ja emitteri vaheline takistus väga väikeseks ning ühendus hakkab elektrit juhtima. Lülituseks vajaliku baasivoolu suurus võib olla sadu kordi väiksem, kui kollektorit ja emitterit läbiv vool. Sammaste juhtimisel

ühendatakse kollektorid 5-voldise toitepingega ja emitterid dioodide anoodidega. Tasandite puhul on emitterid ühendatud pingeallika miinusklemmiga ning kollektorid katoodidega. (Lisa 1)

# 2. ELEKTROONIKA

## 2.1. Elektroonikakomponendid

Tabel 2.1. 8x8x8 suuruse kuubi jaoks vajalikud elektroonikakomponendid

| Komponent | kogus |
|---|---|
| Valge LED | 512 |
| 100 Ω takisti | 64 |
| 560 Ω takisti | 64 |
| 130 Ω takisti | 8 |
| 2N3904 transistor | 64 |
| TIP31C transistor | 8 |
| 74HC595 nihkeregister | 8 |

### 2.1.1. 100 Ω takisiti

Antud takisti ühendatakse dioodsammaste anoodidega jadamisi. Takisti ülesanne on piirata dioode läbivat voolu 20 mA peale. Valge dioodi pingelang on 3,2 V ja lubatav vool kuni 25 mA [3]. Takistus arvutatakse vastavalt valemile [4]:

$$R=(V-V_l)/I=(5-3,2)/0,02=90\ \Omega \quad (3.1)$$

R- takisti väärtus

V- toitepinge

$V_l$- valgusdioodi pingelang

I – kontuuri läbiv vool

Seega 5-voldise toitepinge korral peaks eeltakisti väärtus olema vähemalt 90 Ω. Kasutatavate 100 Ω taksitite korral on tegelik vool 18 mA.

### 2.1.2. 2N3904 npn transistor

2N3904 on madala koormuse kiireks lülitamiseks mõeldud transistor. Antud transistore kasutatakse dioodide anoodide ja nende toite ühenduse lülitamiseks. Valitud nihkeregistri väljundid taluvad üksikult maksimaalselt 35 mA ehk kannataksid ühe dioodi toitmisel tekkiva

voolu ära, kuid 8 bitil on lubatud korraga tarbida kuni 70 mA [5], mis ei rahulda võimalikku tekkivat 144 mA. Antud transistori lubatud kollektorvool on 200 mA ja seega kannatab ära ka terve samba korraga põlemisel tekkiva ca 40 mA.

Tabel 3.1. 2N3904 karakteristikud [6]

| Symbol | Parameter | Test Condition | Min. | Max. | Units |
|---|---|---|---|---|---|
| **ON CHARACTERISTICS** | | | | | |
| $h_{FE}$ | DC Current Gain | $I_C$ = 0.1mA, $V_{CE}$ = 1.0V<br>$I_C$ = 1.0mA, $V_{CE}$ = 1.0V<br>$I_C$ = 10mA, $V_{CE}$ = 1.0V<br>$I_C$ = 50mA, $V_{CE}$ = 1.0V<br>$I_C$ = 100mA, $V_{CE}$ = 1.0V | 40<br>70<br>100<br>60<br>30 | <br><br>300<br><br> | |
| $V_{CE(sat)}$ | Collector-Emitter Saturation Voltage | $I_C$ = 10mA, $I_B$ = 1.0mA<br>$I_C$ = 50mA, $I_B$ = 5.0mA | | 0.2<br>0.3 | V<br>V |
| $V_{BE(sat)}$ | Base-Emitter Saturation Voltage | $I_C$ = 10mA, $I_B$ = 1.0mA<br>$I_C$ = 50mA, $I_B$ = 5.0mA | 0.65 | 0.85<br>0.95 | V<br>V |

### 2.1.3. 560 Ω takisti

Antud takistid ühendatakse anoode juhtivate nihkeregistrite väljundite ja 2N3904 transistoride baaside vahele. Neid kasutatakse transistoride baasivoolu tõstmiseks küllastumiseks vajalikule tasemele. Vajalik kollektorit läbiv vool on 18 mA, transistori minimaalne võimendustegur $h_{FE}$ on 10 mA juures 100 ja 50 mA juures 60 (tabel 3.1). Baasivoolu arvutamiseks kasutatav võimendustegur on 70 ja kollektorvool 18 mA. Baasivool leitakse vastavalt valemile:

$$I_B = I_k / h_{FE} = 20/70 = 0{,}3 \text{ mA} \quad (3.2)$$

$I_B$ – baasivool

$h_{FE}$ – transistori võimendustegur

$I_K$ - soovitud kollektorvool [7]

Seega peab minimaalne baasivool küllastumiseks olema 0,3 mA. Praktikas on kindluse mõttes soovitatav kasutada 2 kuni 10 korda suuremat baasivoolu: $I_B$=0,3x5=1,5 mA. Nihkeregistri väljundite kõrge signaal on 5-voldine, baasi ja emitteri vaheline pingelang on antud tingimuste korral 0,65...0,90 volti. Seega baasi eeltakistile rakendub kõrge signaali korral minimaalselt: 5-0,9=4,1 V.

Eeltakisti leidmiseks kasutatakse Ohmi seadust vooluahela osa kohta [8]:

$$R_{maks} = U_R / I_B = 4{,}1/0{,}0015 = 2734\ \Omega \quad (3.3)$$

$R_{maks}$ – maksimaalne eeltakisti väärtus

$U_R$ – eeltakisti pingelang

Seega peab eeltakisti väärtus olema väiksem kui 2734 Ω. Minimaalne takistus on määratud nihkeregistri väljunditele lubatud maksimaalse vooluga, mis on (70/8) 8,75 mA.

$R_{min}=U_R/I_{maks}=4,1/0,00875=470\ \Omega$ (valem 3.3)

$R_{min}$ – minimaalne eeltakisti väärtus

$I_{maks}$ – maksimaalne nihkeregistri väljundile lubatud voolutugevus

### 2.1.4. TIP31C npn transistor

TIP31C on mõeldud keskmise suurusega koormuse lülitamiseks ja võimendamiseks. Kuubis kasutatakse seda lülitina tasandi katoodide ja pingeallika miinusklemmi vahel. Ühte transistori võib läbida korraga terve tasandi poolt tarbitav vool. Seega peavad transistorid taluma kuni (64x18 mA) 1,152 A suurust voolu. Valitud transistoride maksimaalne lubatud kollektori vool on kuni 3 A [9].

### 2.1.5. 130 Ω takisti

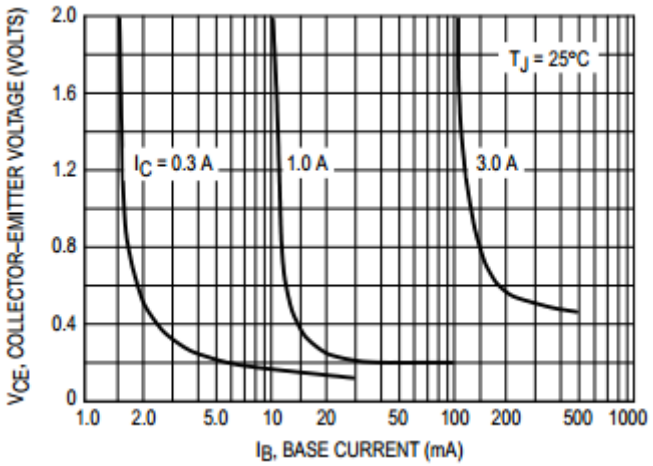
Antud takistid ühendatakse katoode juhtivate kontrolleri väljundite ja TIP31C transistoride baaside vahele. Takisteid kasutatakse TIP31C transistoride baasivoolu tõstmiseks küllastumiseks vajalikule tasemele. 1 A suuruse koormuse korral peaks baasivool olema küllastumiseks vähemalt 30 mA (sele 3.1).

$U_{BE}=0,9V$

$R_{maks}=4,1/0,03=137\ \Omega$ (valem 3.3)

Kasutatava kontrolleri sisend/väljund viikude maksimaalne vool võib olla 40 mA [10], mis on suurim lubatud baasivool. Sellest lähtuvalt arvutatakse minimaalne takisti väärtus.

$R_{min}=U/I_{maks}=4,1/0,04=103\ \Omega$ (valem 3.3)



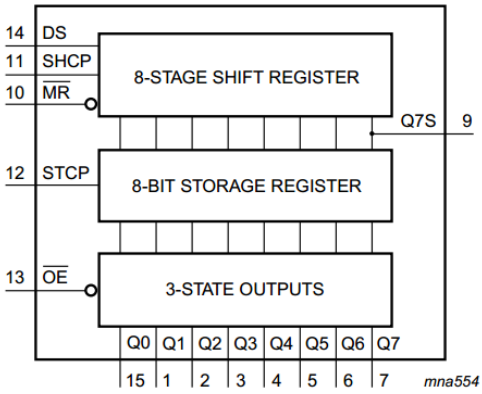

Sele 3.1. TIP31C küllastumise baasivoolud vastavalt kollektorvooludele [9]

### 2.1.6. 74HC595 nihkeregister

Nihkeregistrid muudavad järjestikinfo paralleelinfoks ning sellega annavad vajaliku arvu sisend/väljund viike. 74HC595 nihkeregistril on 8 paralleelset väljundit (sele 3.2). Nihe toimub taktsignaali (ingl.k. *Shift register clock* – SHCP) sisendi tõusva frondiga. Antud registritel on vahepuhverina lisa andmeregister. Seega nihutatud info ei lähe otse väljunditesse. Samal ajal, kui andmeregister väljastab etteantud infot, saab nihkeregistris uut infot nihutada ilma, et see otse väljunditesse läheks. Nihkeregistrist on võimalik andmed väljunditesse saata alles andmeregistri taktsignaali (ingl.k. *Storage register clock* - STCP) sisendi tõusva frondiga [5]. Nihkeregistreid on võimalik ühendada jadamisi nii palju kui vaja, et saada vajalik arv paralleelseid väljundeid. Seega kogu info käib igast registrist läbi ning kõikide antud tasandi dioodide oleku määramiseks on vaja saata 8 baiti (64 bitti) infot.

74HC595 põhilised parameetrid [5]:

- järjestik- ja 8-bitine paralleelväljund
- andmeregister
- 3 olekuga väljundid
- algoleku taastamise sisend
- 100 MHz (tüüpiline) nihke sagedus
- Toitepinge -0,5...7 V




Sele 3.2. 74HC595 ülesehitus ja viigud [5]

| Control | | | | Input | Output | | Function |
|---|---|---|---|---|---|---|---|
| SHCP | STCP | $\overline{OE}$ | $\overline{MR}$ | DS | Q7S | Qn | |
| X | X | L | L | X | L | NC | a LOW-level on $\overline{MR}$ only affects the shift registers |
| X | ↑ | L | L | X | L | L | empty shift register loaded into storage register |
| X | X | H | L | X | L | Z | shift register clear; parallel outputs in high-impedance OFF-state |
| ↑ | X | L | H | H | Q6S | NC | logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S). |
| X | ↑ | L | H | X | NC | QnS | contents of shift register stages (internal QnS) are transferred to the storage register and parallel output stages |
| ↑ | ↑ | L | H | X | Q6S | QnS | contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages |

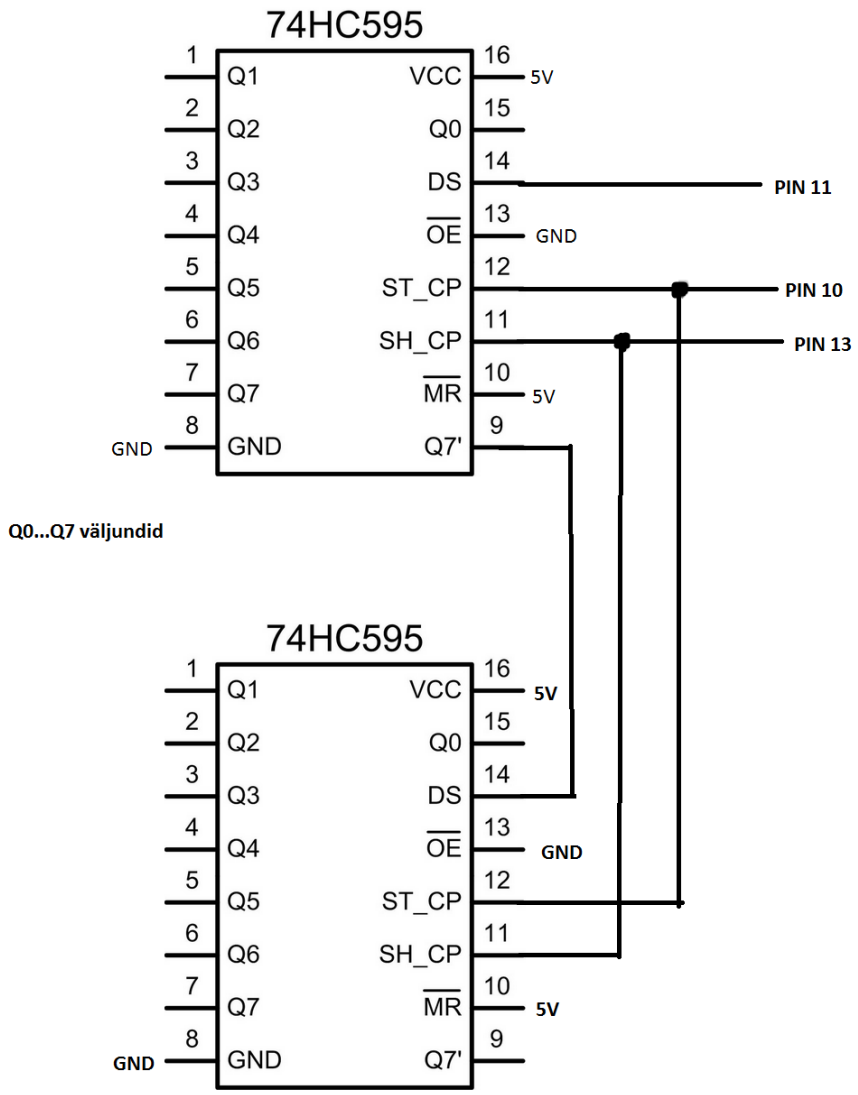
[1] H = HIGH voltage state;
L = LOW voltage state;
↑ = LOW-to-HIGH transition;
X = don't care;
NC = no change;
Z = high-impedance OFF-state.

Sele 3.3. 74HC595 funktsioonid vastavalt sisendiolekutele [5]

## 2.2. Ühendused

Nihkeregister saab kontrollerilt 3 erinevat signaali. DS (ingl.k. *serial data input*) on sisend järjestikinformatsiooni jaoks. Q7S on järjestikinfo väljundviik. Andmed jooksevad jadamisi läbi iga nihkeregistri. SHCP (*shift register clock*) on taktsignaali sisend, iga tõusva takti ajal toimub registris informatsiooni nihe. STCP (*storage register clock*) on sisend, mis väljastab tõusva takti ajal andmed nihkeregistrist andmeregistrisse, mis omakorda on ühendatud väljunditega. SHCP ja STCP viigud on nihkeregistritel ühendatud paralleelselt, signaal neile tuleb kontrollerist. Ühendused kontrolleriga on välja toodud tabelis 4.2. Lisaks on nihke-registril veel 4 sisendit. OE (ingl.k. *Output enable*) on viik, mis aktiveerib väljundid. OE on madalaktiivne ja seega ühendatud toite miinusklemmiga. MR (ingl.k. *Master Reclear*) on

samuti madalaktiivne [5]. Antud viik taastab madala signaali korral algoleku ehk saadab nihkeregistrisse nullid. Et seda ei juhtuks, on MR ühendatud 5-voldise pingega. VCC on toiteviik, mis on ühendatud 5-voldise pingeallikaga ja GND on ühendatud pingeallika miinusklemmiga. Nihkeregistri ühendused on väljatoodud sele 3.4.




Sele 3.4. Nihkeregistrite ühendused

Nihkeregistrite väljundite transistoride ja dioodide vahelised ühendused on välja toodud lisas 1.

## 2.3. Toide

Arduino kontrollerile sobiv toitepinge on 7...12 volti [10]. Pingeallikana kasutatakse 9-voldist patareid. Valgusdioodide ja nihkeregistrite toitepingena (piirväärtused -0,5..7 V [5]) kasutatakse 5 volti, mis saadakse 6-vatise alalisvoolu toiteadapteri abil. Adapterile lubatud

vool on kuni (6/5=1,2) 1,2 A, mis ületab kuubi poolt maksimaalselt tarbitava 1,152 A (ptk 2.1.4).

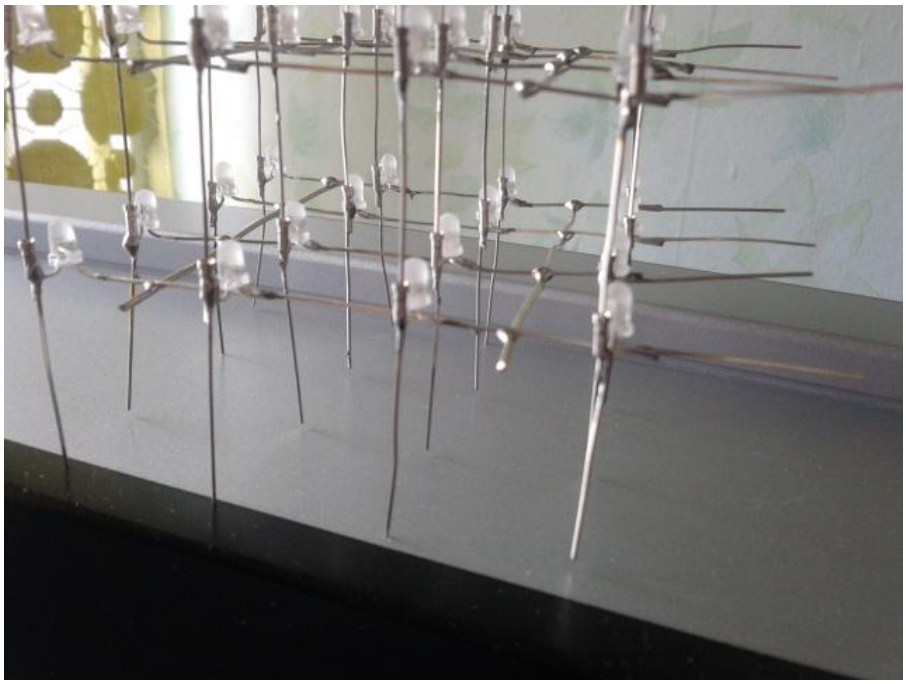
## 2.4. Laiendatavus

Valmistatakse platvorm ning moodulid kuni 8x8x8 suuruse kuubi jaoks. Antud platvormi on võimalik suhteliselt kerge vaevaga suurendada kõrgemale kuubile vastavaks. Iga tasandi kohta tuleks kasutada ühte lisa transistori, takistit ja umbes 20 cm juhet. Laiendamine oleks keerulisem, sest 12x12 põhjaga kuubi jaoks peaks elektroonikat rohkem kui kahekordistama: 64 asemel 144 dioodi tasandi kohta. Iga dioodiga kaasneks lisaks 2 takistit, transistor ja nihkeregistri väljund. Katoode juhtivad transistorid taluksid 144 valgusdioodiga kaasnevat voolu, kuid küllasrežiimi viimiseks oleks vaja väiksema väärtusega eeltakisteid. See tähendab, et kuup ei põleks olemasolevate takistitega maksimaalse eredusega.

# 3. MEHAANIKA

## 3.1. Moodulid

Juba kokkujoodetud kuubil on dioodide vahetamine väga ebamugav ja aeganõudev. Selle protsessi lihtsustamiseks saab kasutada pistikuid ja LED mooduleid. Pistikute abil on võimalik jõuda vähese vaevaga kuubi keskel olevate mooduliteni. Rikkis valgusdioodi on võimalik hõlpsamini asendada ning veel lihtsam on vahetada terve moodul uue vastu. Moodulite abil on võimalik ka kuupi laiendada, kui ülejäänud elektroonika on selleks vastav. Vastavate moodulite olemasolu korral on võimalik vahetada ka kuubi värvust üsna hõlpsasti. Moodulid tehakse suurusega 4x4 dioodi. Suuremate moodulite ühendamine oleks väga ebamugav ning väiksemad moodulid jätaksid kuubi tervikuna mehaaniliselt liiga nõrgaks ja ebastabiilseks. Dioodidevaheline kaugus on 22 mm. Iga moodul koosneb neljast reas. Iga rea dioodide katoodid on kokku joodetud. Read ühendatakse kahe tinatatud vasktraadiga (diameeter 0,75 mm) jootmise teel (sele 3.1). Kuna dioodide viigud on üsna pehmed, siis vasktraadid annavad moodulitele ka vajaliku jäikuse.

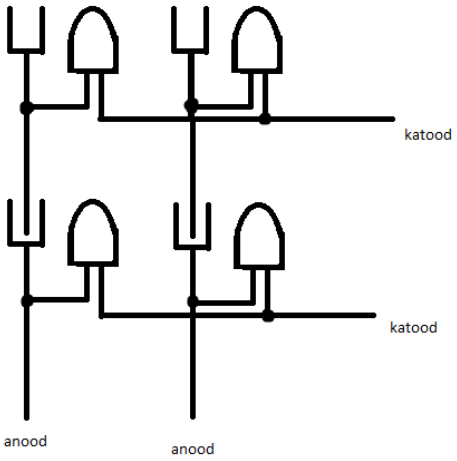


Sele 3.1. 4x4 moodulid üksteise kohal

## 3.2. Pistikud

Pistikute valmistamisel kasutatakse aukribasid (sele 3.2). Iga anoodi külge joodeti vertikaalses sihis aukriba element, mis oleks pistikupesaks selle kohale käiva dioodi anoodi jaoks (sele

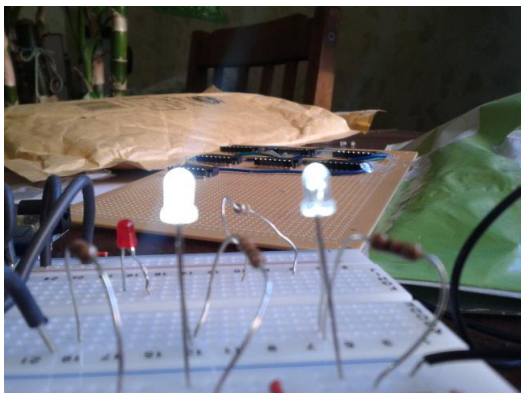
3.3). Samamoodi joodetakse ühe katoodi külge pistikupesa, millega ühendatakse ühel tasandil olevad moodulid. Parema visuaalse tulemuse jaoks eemaldatakse aukribade ümbert isolatsioon (sele 3.8). Aukribad valiti nende madala hinna (400 tk – 2,18 EUR [11]) tõttu ja nende pesa mõõtmed sobivad valgusdioodide jalgadega kokku. Aukribade ja dioodi viikude vaheline kontakt on piisavalt tugev, et kuup saavutaks vajaliku jäikuse ning moodulid ei hakkaks üksteise otsas logisema. Lubatud maksimaalne voolutugevus 4 A [12] on piisavalt suur. Pistikute mõõtmed on väikesed ja ei paista silma, kuid ühendus on piisavalt tugev. Aukribasid kasutatakse ka ülejäänud elektroonika pistikutena (sele 3.7).




Sele 3.2. Kasutatavad aukribad [12]  Sele 3.3. pistikupesad anoodide küljes

## 3.3. Dioodide tuhmistamine

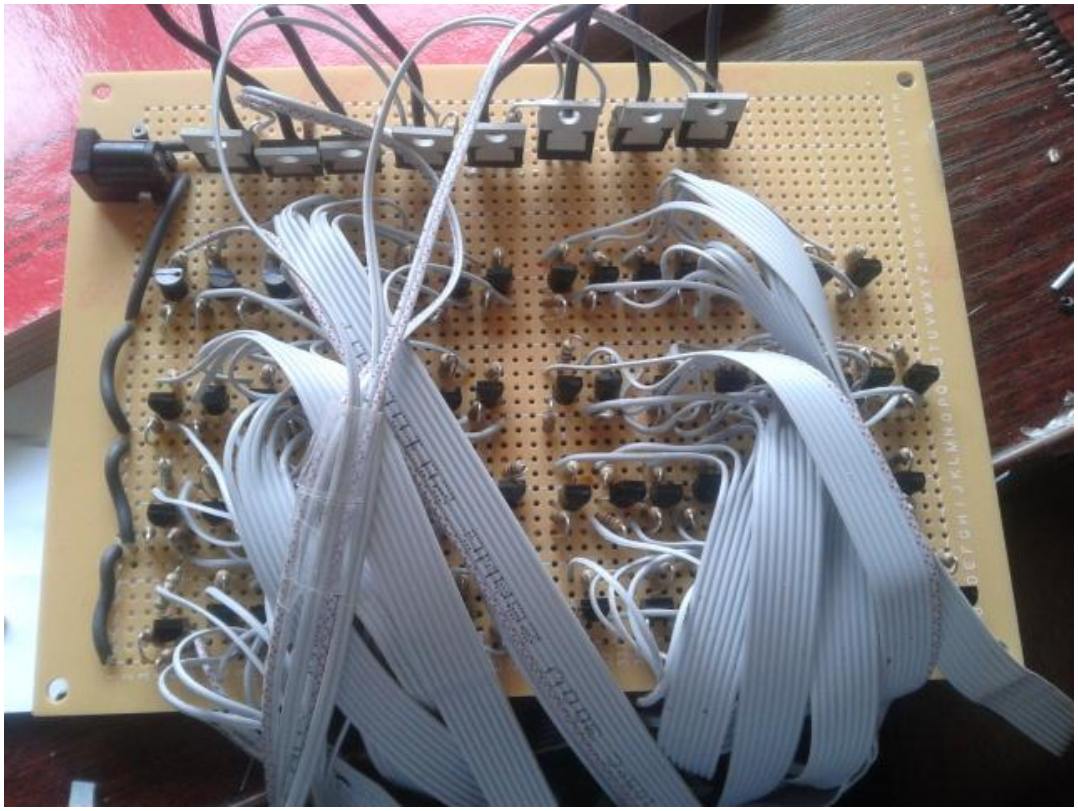
Dioodidel on algselt valgustamisnurk vaid 20 kraadi otse üles. Selle parandamiseks lihvitakse liivapaberiga nende klaasist kuplid tuhmiks ning valgus peegeldub ühtlaselt laiali (sele 3.4).



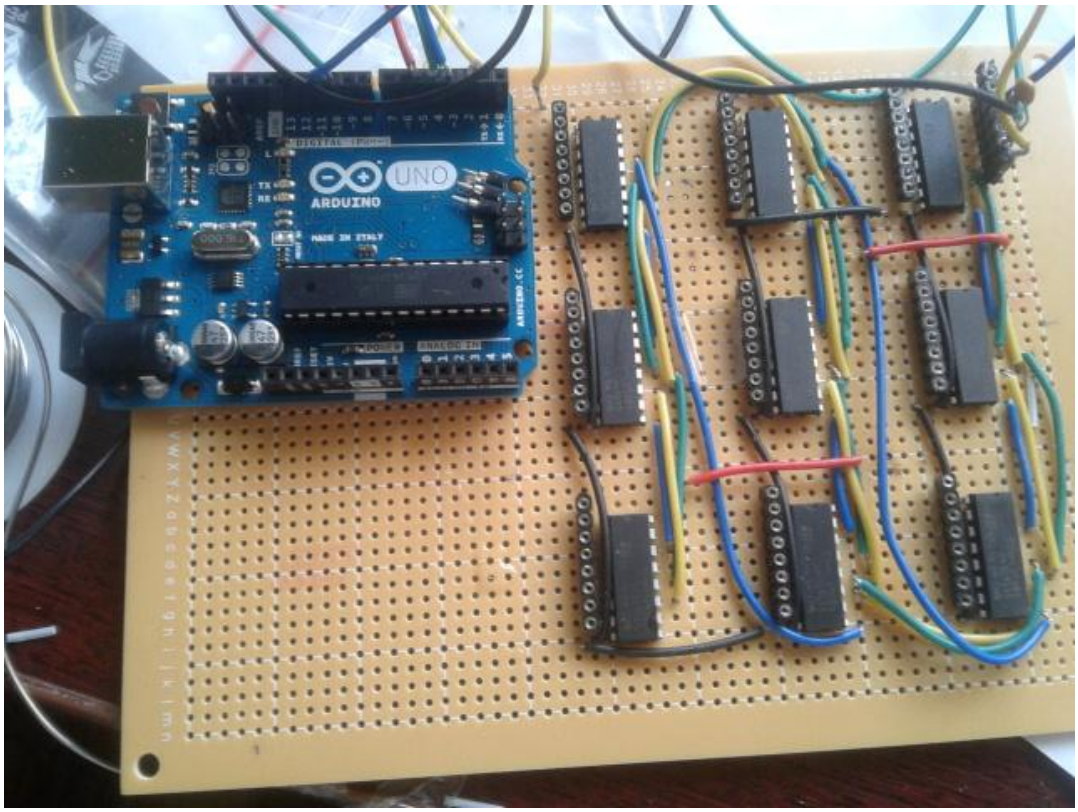
Sele 3.4. paremal diood enne ja vasakul pärast lihvimist.

## 3.4. Elektroonika paigutus

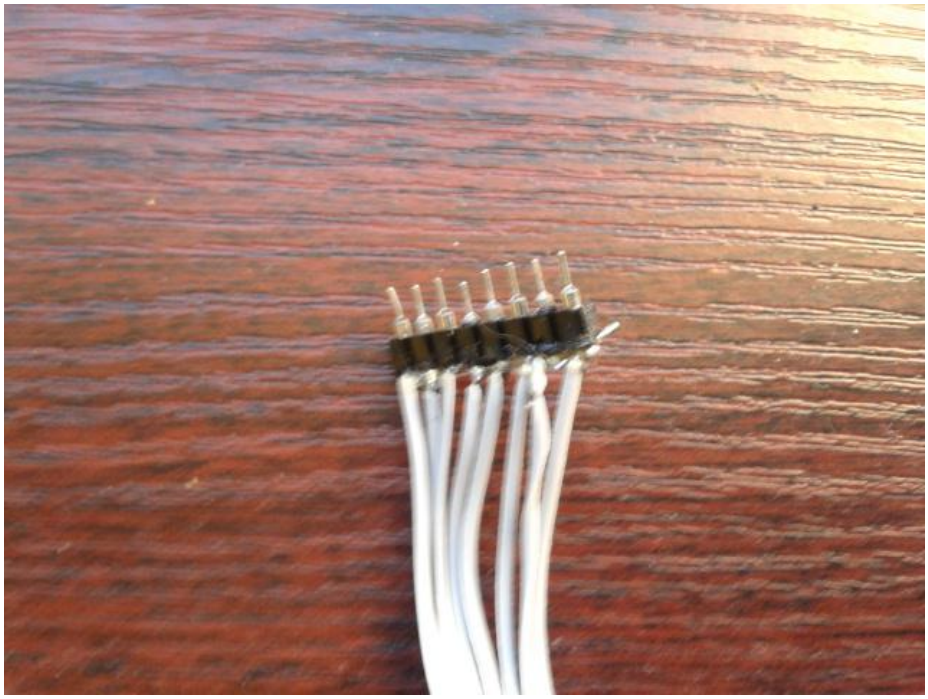
Elektroonika on mahutatud kahele 160x115 mm suurusele üheste (kokku 2200) aukudega makettplaadile. Väiksematele plaatidele poleks elektroonika mugavalt ära mahtunud ning ühele suurele plaadile mahutades oleks pidanud ka tunduvalt laiema korpuse tegema, mis halvendaks kuubi välisilmet. Üheseid auke on mugav vastavalt vajadusele kokku joota ning radasid teha. Ühel plaadil on nihkeregistrid ja teisel transistorid koos takistitega (sele 3.5). Nihkeregistrite kõrvale mahub plaadi peale ära ka Arduino kontroller (sele 3.6). Plaadid on paigutatud üksteise kohale M5 poltide abil. Elektroonika on ühendatud lintkaablitega, mille otsas on aukribadest tehtud pistikud (sele 3.7). Plaatidevaheline kaugus on 30mm.



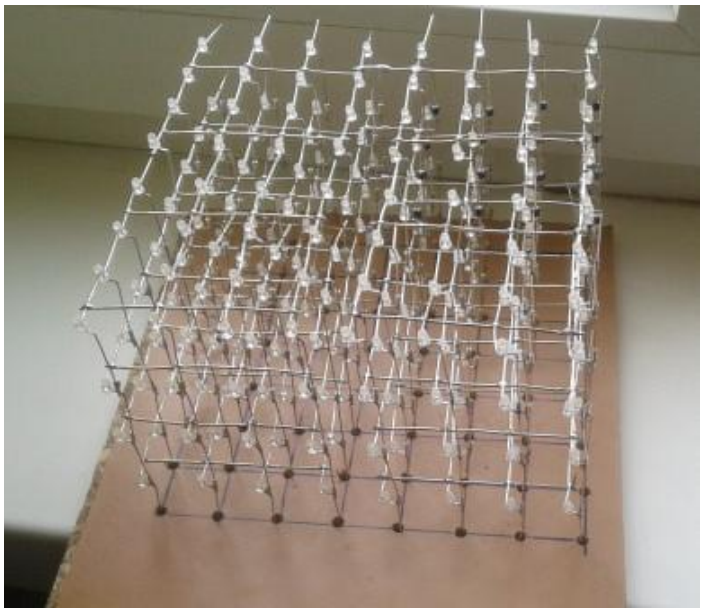
Sele 3.5. Transistoride plaat




Sele 3.6. Nihkeregistrite plaat



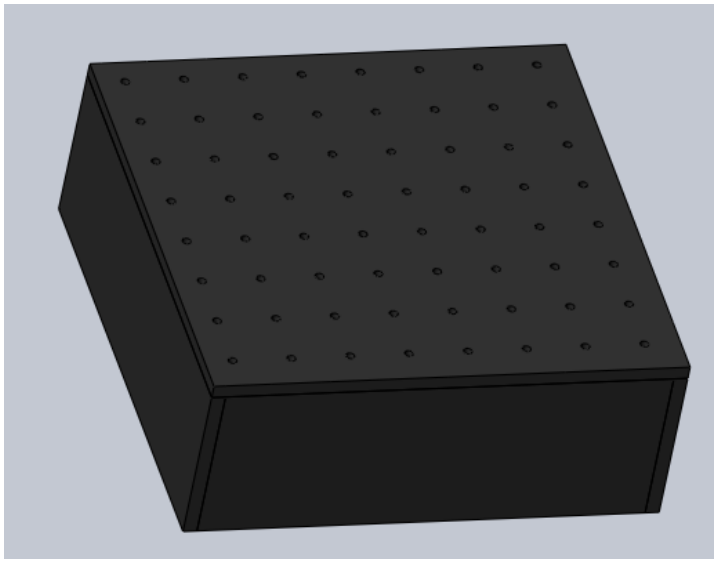
Sele 3.7. Pistikud lintkaabli otsas



Sele 3.8. 8x8x4 kuup

## 3.5. Korpus

Parema välimuse saavutamiseks kaetakse kuubi alla jääv elektroonika korpusega. Kaks üksteise kohal olevat elektroonika plaati katavad ära ruumi 160x115x60 mm. 8x8 suuruse tasandi mõõtmed on 154x154 mm. Korpusel jäetakse igast küljest 13 mm varu. Seega korpuse mõõtmed on 170x170x60 mm. Korpus valmistatakse puitmaterjalist plaatidest, mis ühendatakse kruvidega ning värvitakse mustaks (sele 3.9). Anoodide jaoks puuritakse 3 mm läbimõõduga avad, kust mahuvad läbi ka juhtmed tasandite katoodide jaoks. Ühele külgpinnale on tehtud ava toitepistiku ja USB kaabli jaoks.



Sele 3.9 Korpuse mudel

# 4. PROGRAMMEERIMINE

## 4.1. Kontrolleri valik

Kuubi juhtimiseks kasutatakse Arduino Uno Rev3 arendusplaati, millel on mikrokontroller Atmega328. Arduino Uno valiti sellepärast, et see on lihtsa kasutajaliidesega ja soodne arendusplaat ning samas vastab kõikidele vajadustele. Sellel on 14 digitaalset sisend/väljundit, 6 analoog sisendit ja 16 MHz taktsignaal. Arvutiga ühildumine käib USB kaabli abil. Laialdase leviku tõttu võib leida palju abimaterjale ning õpetusi erinevate plaadi funktsioonide kasutamise kohta. Arduinole oli olemas lähteprogramm kuubi juhtimise jaoks. Toide tuleb läbi USB kaabli arvutist, selleks on ka eraldi sisend viik ja ka 2.1/5.5 mm pistikupesa [10]. Arduino on võimalik soetada umbes 10 euroga [13].

Arduino Uno parameetrid [10]:

- Mikrokontroller — ATmega328
- Töö pinge — 5V
- Sisend pinge (soovituslik) — 7-12V
- Sisendpinge (piirväärtused) — 6-20V
- Digitaalsed sisend/väljund viike — 14 (6 neist võimaldavad pulsilaiusmodulatsiooni)
- Analoog sisend viike — 6
- Alalisvool sisend/väljund viigu kohta — 40 mA
- Alalisvool 3.3V viigu jaoks — 50 mA
- Välkmälu — 32 KB (ATmega328)
- Taktsignaal — 16 MHz

## 4.2. Andmeside protokoll

SPI (ingl.k. *Serial Peripheral Interface*) on sünkroonne järjestikinfo protseduur, mida mikrokontrollerid kasutavad teiste seadmetega kiireks infovahetuseks. Seda võib kasutada ka kahe mikrokontrolleri vaheliseks kommunikatsiooniks [14].

SPI ühendusel on alati üks ülem seade (tavaliselt mikrokontroller), mis kontrollib välisseadmeid ehk alamaid. Tavaliselt on 3 ühendust ühised kõigile seadmetele [14]:

- MISO (ingl.k. *Master In Slave Out*) – Liin alamalt ülemale info saatmiseks
- MOSI (ingl.k. *Master Out Slave In*) – Liin ülemalt alamale info saatmiseks
- SCK (ingl.k. *Serial Clock*) – Taktsignaali pulsid sünkroniseerivad ülema poolt saadetava info saatmise

1 liin on igal seadmel eraldi:

- SS (ingl.k. *Slave Select*) – Viik, millega ülem valib missuguse seadmega ta infot vahetab

Kui SS viik on madal, siis seade suhtleb ülemaga. Kui see on kõrge, siis seade ignoreerib ülemat. SS lubab mitmetel seadmetel jagada samu MISO, MOSI ja CLK liine.

Kõik SPI seaded on määratud SPI kontrollregistriga (ingl.k. *SPI Control Register* -SPCR). Register on ainult 1 bait mikrokontrolleri mälu, mida saab lugeda ja sinna kirjutada. Kontrollregistritega juhitakse erinevaid mikrokontrolleri funktsioone. Tavaliselt määrab iga registri bitt mingi konkreetse seadistuse, näiteks info saatmise kiiruse [15]. SPCP bittidele vastavad seaded on välja toodud tabelis 4.1.

Tabel 4.1. SPCR bittidele vastavad seaded [15]

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |

SPIE – Käivitab SPI katkestuse, kui väärtus on 1
SPE – Käivitab SPI, kui väärtus on 1
DORD – SPI saadab andmeid vähima kaaluga bitt esimesena, kui väärtus on 1. Kui väärtus on 0, saadetakse suurima kaaluga bitt esimesena.
MSTR – Seab Arduino ülem-seadmeks, kui väärtus on 1 ja alamaks, kui väärtus on 0
CPOL – Määrab taktsignaali polaarsuse
CPHA – Saadab infot langeva taktsignaali frondiga, kui väärtus on 1 ja tõusva frondiga, kui väärtus on 0
SPR1 ja SPR0 – Määravad SPI kiiruse, 00 on kiireim (4MHz) 11 on aeglaseim (250KHz)

Andmeregistrid hoiavad endas ühte baiti infot. Näiteks SPI andmeregister (ingl.k. *SPI data register* - SPDR) hoiab endas baiti, mis saadetakse MOSI viigu kaudu nihkeregistrisse [15].

Olekuregister (SPI *status register* – SPSR) muudab oma seisundit vastavalt mikrokontrolleri olekule antud hetkel. Näiteks SPSR 7. bitt muutub üheks, kui samal ajal toimub info vahetus SPI kaudu [12].

Tabel 4.2. Arduino Uno arendusplaadile vastavad SPI viigud [14]

| Arduino Board | MOSI | MISO | SCK | SS (slave) | SS (master) |
|---|---|---|---|---|---|
| Uno or Duemilanove | 11 or ICSP-4 | 12 or ICSP-1 | 13 or ICSP-3 | 10 | - |

## 4.3. Programmi kirjeldus

LED kuubi põlemismustrite kontrollimiseks on vaja juhtprogrammi. Kuubi eesmärk on kuvada reklaamimaterjale ning sellest tulenevalt pidi programm kuvama tähthaaval tekste. Töös kirjeldatav programm põhineb olemasolnud lahendusel [16]. Terviklik programm on välja toodud lisa 2.

Kõigepealt deklareeritakse kontrolleri väljunditele vastavad viigu numbrid, hetkel sisselülitatud tasandi number ning baitide massiiv, mis hoiab endas iga dioodi väärtust (sele 4.1).

```
#include <TimerOne.h>
#include <string.h>
#define AXIS_X 1
#define AXIS_Y 2
#define AXIS_Z 3

//viik, mis on ühendatud nihkeregistrite STCP-ga
int latchPin = 10;
//viik, mis on ühendatud nihkeregistrite SHCP-ga
int clockPin = 13;
//viik, mis on ühendatud esimese nihkeregistri DS-ga
int dataPin = 11;
//PORTB viigu number ühendatud STCP
int latchPinPORTB = latchPin - 8;
//baitide jada, kus hoitakse iga LED väärtust [x][y][z]
byte cube[8][8];

//hetkel sisselülitatud tasandi number
int current_layer = 0;
```

Sele 4.1. Deklaratsioonid [16]

*void iProcess()* on funktsioon, mis käivitatakse katkestusega. Funktsioon lülitab sisse järgmise tasandi ja kustutab eelmise. Lisaks toimub funktsiooni *spi_transfer* esilekutsumine ning sellele antakse argumendiks iga antud tasandi valgusdioodi väärtus. Lõpuks lülitatakse STCP viik madalaks ja siis uuesti kõrgeks, et tõusva frondiga saata andmed nihkeregistri väljunditesse (sele 4.2).

```
//katkestusega käivitatav funktsioon
void iProcess(){
  //sisselülitatud tasandile vastav viigu number kontrolleril
  int oldLayerBit = current_layer + 2;

  //tasandi numbri 1 võrra suurendamine
  current_layer++;
  if(current_layer >= 8){
    current_layer = 0;
  }

  //Kutsutakse esile spi_transfer funktsioon
  //ja määratakse selle argumentideks uue tasandi dioodide väärtused
  latchOff();
  for (int i = 0 ; i < 8 ; i++){
    spi_transfer(cube[current_layer][i]);
  }

  //vana tasand kustutatakse
  digitalWrite(oldLayerBit, LOW);
  //SPCP kõrgeks tõstmine ja nihkeregistrite andmete saatmine väljunditesse
  latchOn();
  //järgmise tasandi sisselülitamine
  digitalWrite(current_layer + 2, HIGH);
}

//STCP kõrgeks ja madalaks muutmise funktsioonid
void latchOn(){
  bitSet(PORTB,latchPinPORTB);
}
void latchOff(){
  bitClear(PORTB,latchPinPORTB);
}
```

Sele 4.2. funktsioon *iProcess()* [16]

*void setupSPI()* on SPI kontrollregistri seadistamise funktsioon (sele 4.3). Arduino kontroller tehakse ülem-seadmeks ning Käivitatakse SPI vastavate bittide kõrgeks tõstmisega. SPI kiirus seatakse 4 MHz peale. SPI2X kahekordistab SPI taktsignaali kiirust.

```
//SPI seadistus
void setupSPI(){
  byte clr;
  SPCR |= ( (1<<SPE) | (1<<MSTR) );//SPI sisselülitamine ja seab Arduino ülemseadmeks
  SPCR &= ~( (1<<SPR1) | (1<<SPR0) );//SPI kiiruse määramine (kõige kiirem)
  clr=SPSR;
  clr=SPDR;
  SPSR |= (1<<SPI2X); // taktsignaali kiirus kahekordistatakse
  delay(10);
}
```

Sele 4.3. SPI seadistus

*Byte spi_transfer(byte)* on funktsioon, mis kirjutab argumendiks määratud baidi SPI andmeregistrisse ning sealt saadetakse see edasi nihkeregistritesse (sele 4.4). Funktsioon töötab kuni SPI olekuregistris 7. bitt (ingl.k. SPI *Interrupt Flag* - SPIF) kõrgeks tõuseb. Antud bitt tõuseb kõrgeks kui SPI andmete ülekanne on lõppenud [17].

```
//SPI andmeregistrisse kirjutamine
byte spi_transfer(byte data)
{
  //argument kirjutatakse andmeregistrisse
  SPDR = data;
  //funktsioon töötab kuni andmeedastus on lõppenud
  loop_until_bit_is_set(SPSR, SPIF);
  return SPDR;
}
```

Sele 4.4. Andmeregistrisse kirjutamine [16]

*void Setup()* on seadistusfunktsioon, mis määrab, et kontroller saadab välja 9600 bitti sekundis. Tasandeid juhtivad viigud määratakse väljunditeks, andmeside väljundid muudetakse madalaks ning kutsutakse välja SPI seadistus funktsioon. Taimer 1 kutsub iga 1000 mikrosekundi järel katkestuse, mis omakorda kutsub esile *iProcess()* funktsiooni (sele 4.5). *Timer1.initialize()* funktsiooni argumendiga määratakse tasandi sisselülitusperiood mikrosekundites.

```
//seadistus funktsioon
void setup() {
  // kontroller saadab 9600 bitti sekundis
  Serial.begin(9600);

  //tasandi viikude määramine väljundiks
  for(int i = 2; i < 10; i++)
  {
    pinMode(i, OUTPUT);
  }
  //andmeside viikude määramine väljunditeks ja nende madalaks kirjutamine
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);

  digitalWrite(latchPin,LOW);
  digitalWrite(dataPin,LOW);
  digitalWrite(clockPin,LOW);

  //SPI seadistuse esilekutsumine
  setupSPI();

  //PWM tasandi viikude taimeri aktiveerimine
  //iga 1000 mikrosekundi tagand kutsutakse esile katkestus
  Timer1.initialize(1000);
 //katkestusega kutsutakse esile iProcess funktsioon
  Timer1.attachInterrupt(iProcess);
}
```

Sele 4.5. Seadistus funktsioon [16]

*char font_data[][]* on kahemõõtmeline massiiv, millesse salvestatakse iga tähemärgi kuvamiseks vajalik info (sele 4.6).

```
// tähtede andmeid sisaldav jada
char font_data[128][8] = {

  { 0x00,  0x3C,  0x66,  0x66,  0x7E,  0x66,  0x66,  0x66 },    // 65 : A
                //              |          |
                //              |   ****   |
                //              |  **  **  |
                //              |  **  **  |
                //              |  ******  |
                //              |  **  **  |
                //              |  **  **  |
                //              |  **  **  |

  { 0x00,  0x7C,  0x66,  0x66,  0x7C,  0x66,  0x66,  0x7C },    // 66 : B
                //              |          |
                //              |  *****   |
                //              |  **  **  |
                //              |  **  **  |
                //              |  *****   |
                //              |  **  **  |
                //              |  **  **  |
                //              |  *****   |
```

Sele 4.6. Tähemärkide info salvestamine [16]

Kuvatav tekst määratakse *int string[charNum]* jadasse kirjutamisega. *CharNum* määrab kuvatavate tähemärkide arvu. Igal tähemärgil on *char font_data[][]* massiivile vastav järjekorranumber. Soovitud tähemärkide järjekorranumbrid kirjutataksegi *int string[charNum]* jadasse. *void effect_text(int delay)* on funktsioon, mis kuvab määratud teksti kuubil. Funktsiooni argumendiga määratakse teksti kuvamise kiirus (sele 4.7).

```
// Kuvatav tekst
const int charNum = 10;//teksti tähemärkide arv
// valitakse tähemäkide järjekorranumbrid vastavalt soovitud tekstile
int string[charNum] = {76,69,68,0,75,85,85,80,33,0};//"LED KUUP! "
// Teksti kuvamis funktsioon
void effect_text(int delayt){
  fill(0x00);
  for (int ltr = 0; ltr < charNum; ltr++){//Iga tähemärgi jaoks
    for(int dist = 0; dist < 8; dist++) { //liigutab tähe mööda kuupi ette
      fill(0x00);
      int rev = 0;
      for (int rw = 7; rw >= 0; rw--) {//ridade kopeerimine
         // Igale dioodile omistatakse vastav väärtus, bitswap pöörab baidi tagurpidi
         cube[rev][dist] = bitswap(font_data[string[ltr]][rw]);
         rev++;
      }
      delay_ms(delayt);//teksti kuvamise kiirus
    }
  }
}
```

Sele 4.7. Teksti kuvamise funktsioon [16]

Lõpuks kutsutakse *void loop()*-is esile teksti kuvamise funktsioon *effect_text()* ning hakatakse seda kordama (sele 4.8).

```
void loop(){

  while (true)
  {
    effect_text(2000);
  }
}
```

Sele 4.8. *void loop()*

# 5. MAKSUMUSHINNANG

8x8x8 suuruse kuubi komponentide koguhind on 47,85 eurot. Kallimad osad on toiteadapter, mis maksab 12 eurot ja 10-eurone kontroller (tabel 5.1). Ebay.com e-poes maksab soodsaim 8x8x8 kuubi komponentide komplekt 62 eurot, sellele tuleb lisada 12-eurone toiteadapteri hind. Arvestades, et tegu on kokkupanemata kuubiga, võib otseselt kuupide maksumusi võrrelda. Projekteeritud kuubi hind moodustab 65 % müügil oleva hinnast. Erinevates õpetustes pakutavate lahenduste hinnad on umbes 50 eurot, mis on projekteeritud kuubiga sama suurusjärk.

Antud kuup ebay.com-ist maksab komplekteeritud kujul 144,4 €. Vajalike tööriistade olemasolul võtab projekteeritud kuubi koostamine aega umbes 40 tundi. Seega oleks koostamise tunni hind umbes 4 eurot. Silmas tuleb pidada seda, et kaubandusvõrgustikest ei pakuta modulaarseid kuupe.

Tabel 5.1. Komponendid ja nende hind

| komponent | kogus (tk) | Tüki hind (EUR) | Hind kokku (EUR) |
|---|---|---|---|
| Takisti 100 Ω | 64 | 0,016 | 1,28 |
| Takisti 560 Ω | 64 | 0,016 | 1,28 |
| Takisti 130 Ω | 8 | 0,016 | 0,13 |
| Transistor 2N3904 | 64 | 0,018 | 1,15 |
| Transistor TIP31C | 8 | 0,29 | 2,32 |
| Nihkeregister 74HC595 | 8 | 0,25 | 2,00 |
| 160x115 makettplaat | 2 | 2,90 | 5,80 |
| 3 mm Valge LED | 512 | 0,015 | 7,68 |
| 5 V, 6W AC/DC adapter | 1 | 12,00 | 12,00 |
| 2.1/5.5 pistikupesa | 1 | 0,30 | 0,30 |
| Arduino Uno arendusplaat | 1 | 10,16 | 10,16 |
| Aukriba | 664 | 0,0055 | 3,65 |
| | | | 47,85 |

# KOKKUVÕTE

Antud lõputöös antakse ülevaade LED kuubi toimimise põhimõttest, kirjeldatakse nihkeregistrite abil järjestikinfo muutmist paralleelinfoks, projekteeritakse elektroonika, selgitatakse komponentide valikut, nende funktsioone ja ühendusi.

Töö raames valmis elektrooniline platvorm 8x8x8 suuruse LED kuubi jaoks ning sellega ühilduvad 4x4 moodulid, mis võimaldavad kuupi paremini hooldada. Moodulite ühendamiseks leiti soodne ja lihtne lahendus aukribadest pistikute näol. Tinatatud vasktraadid aitasid saavutada eesmärgiks võetud vajalikku mehaanilist jäikust, mis hoiab kuupi piisavalt sümmeetrilisena. Mehaanilist tugevust aitaks parandada jäigemast materjalist viikudega dioodide kasutamine. Elektroonika katteks projekteeriti ka korpus, mis annab kuubile kaubandusliku välimuse.

Programmeerimise osas põhjendatakse mikrokontrolleri valikut ning selgitatakse nihkeregistrite ja mikrokontrolleri vaheliseks andmevahetuseks kasutatavaid SPI protseduure. Kuubi jaoks modifitseeriti reklaami eesmärgil töötav programm, mis kuvab tähthaaval etteantud tekste. Lõputöös on välja toodud kommenteeritud programm koos selle tööpõhimõttega.

Kuubi jaoks valiti võimalikult soodsad komponendid, mille maksumus oli kokku alla 50 euro. Sellega täideti eesmärk, mille kohaselt ei tohtinud kuubi maksumus olla suurem, kui olemasolevate lahenduste hinnad.

Kuubi võimalik arendussuund oleks RGB valgusdioodide kasutamine. RGB dioodid kiirgavad punast, rohelist ja sinist valgust. Nende värvide erinevate eredustega kombineerides on võimalik kiirata ka teiste värvitoonidega valgust. Seega oleks võimalik projekteerida värvimuutev kuup. Selleks peaks elektroonika olema ligi 3 korda mahukam ning juhtprogramm peaks võimaldama dioodide põlemisereduse reguleerimist.

Olen töö tulemustega rahul. Leitud lahendused ei olnud liiga keeruliselt teostatavad ning projekteeritud kuubi koduste vahenditega valmisehitamine andis rahuldava tulemuse.

## SUMMARY

This thesis provides a review of LED cube working principles. The conversion of serial to parallel information with shift registers is described, electronics is designed and the choice of components, their functions and connections are explained.

During this thesis an electronic platform for 8x8x8 LED cube and 4x4 modules for it were made. Modules are allowing easier maintainance for the cube. For connecting the modules there was a cheap and simple solution found by using female pin headers. Tinned copper wire helped to achieve required mechanical stiffness, which holds the cube fairly symmetrical. Mechanical strenght could be improved by using stiffer materials for diode's pins. A shell was designed for covering the electronics. The shell improves the appearance of the cube.

The choice of microcontroller is explained and the SPI procedures, which are used for the communication between shift registers and microcontroller are described in the programming section . The program was modified for advertising purposes. It displays given texts letter by letter. The commented program with working principles is showed.

The cheapest components were chosen for the cube. Total price of the components was under 50 euros. The objective was to design a cube which is not more expensive than existing solutions and it was fulfilled.

One development direction for the cube is a usage of RGB diodes. RGB diodes emit red, green and blue light. By combining different brightnesses of these colors it is possible to emit other colors as well. Therefore it is possible to design a color-changing cube. It requires almost 3 times more electronics and it has to be possible to regulate the birightnesses of diodes by the program.

I am satisfied with the results. The solutions were not too complicated to put into practice and built results of the designed cube with domestic tools were satisfactory.

# KASUTATUD KIRJANUDS

**Interneti leheküljed**:


1. Tehnika projektide keskkond, 7x7x7 kuubi elektroonika õpetus [*WWW*] http://www.instructables.com/id/Self-Contained-7x7x7-LED-Cube/step6/Wiring-Diagrams/ (25.04.2014)

2. Kevin Darrah kodulehekülg, 8x8x8 RGB kuubi elektroonika skeem [*WWW*] http://www.kevindarrah.com/download/8x8x8/Cube_sch_3_16_13.pdf (25.04.2014)

3. 3 mm valge valgusdioodi andmeleht [*WWW*] http://www.vishay.com/docs/85198/vlhw4100.pdf

4. Robootika Kodulabori kodulehekülg [*WWW*] http://home.roboticlab.eu/et/electronics/led_resistor (19.04.2014).

5. Nihkeregister 74HC595 andmeleht [*WWW*] http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf (18.04.2014)

6. Transistor 2N3904 andmeleht [*WWW*] http://www.fairchildsemi.com/ds/2N/2N3904.pdf (20.04.2014).

7. Amatöörrobootika kodulehekülg [*WWW*] http://robootika.tech-thing.org/failid/Skeemitehnika/04_transistorid.pdf (20.04.2014)

8. Elektrotehnika õppematerjalide lehekülg [*WWW*] http://helia.ee/andres2/?page_id=116 (20.04.2014)

9. Transistor TIP31C andmeleht [*WWW*] https://www.futurlec.com/Transistors/TIP31C.shtml (21.04.2014).

10. Arduino kodulehekülg [*WWW*] http://arduino.cc/en/Main/arduinoBoardUno (22.04.2014)

11. Aukribade müügileht [*WWW*] http://www.ebay.com/itm/10-pcs-1x40-Pin-Single-Row-2-54-mm-Round-Female-Header-DIY-/121322275537?pt=LH_DefaultDomain_0&hash=item1c3f5f02d1 (12.04.2014)

12. Aukribade müügileht [*WWW*] http://www.ebay.co.uk/itm/3x-40-pin-round-female-straight-SIL-header-2-54mm-spacing-/171257916015?ssPageName=ADME:L:OU:US:3160 (12.04.2014)

13. Arduino müügileht [*WWW*] http://www.ebay.com/itm/Arduino-UNO-R3-V3-0-ATMEGA328-FREE-USB-CABLE-Dev-Module-Ideal-4-Robotics-Rev-3-/230934013903?pt=UK_Computing_Other_Computing_Networking&hash=item35c4bd67cf (20.04.2014)

14. Arduino koduleheküg, SPI tutvustus [*WWW*] http://arduino.cc/en/Reference/SPI (04.05.2014)

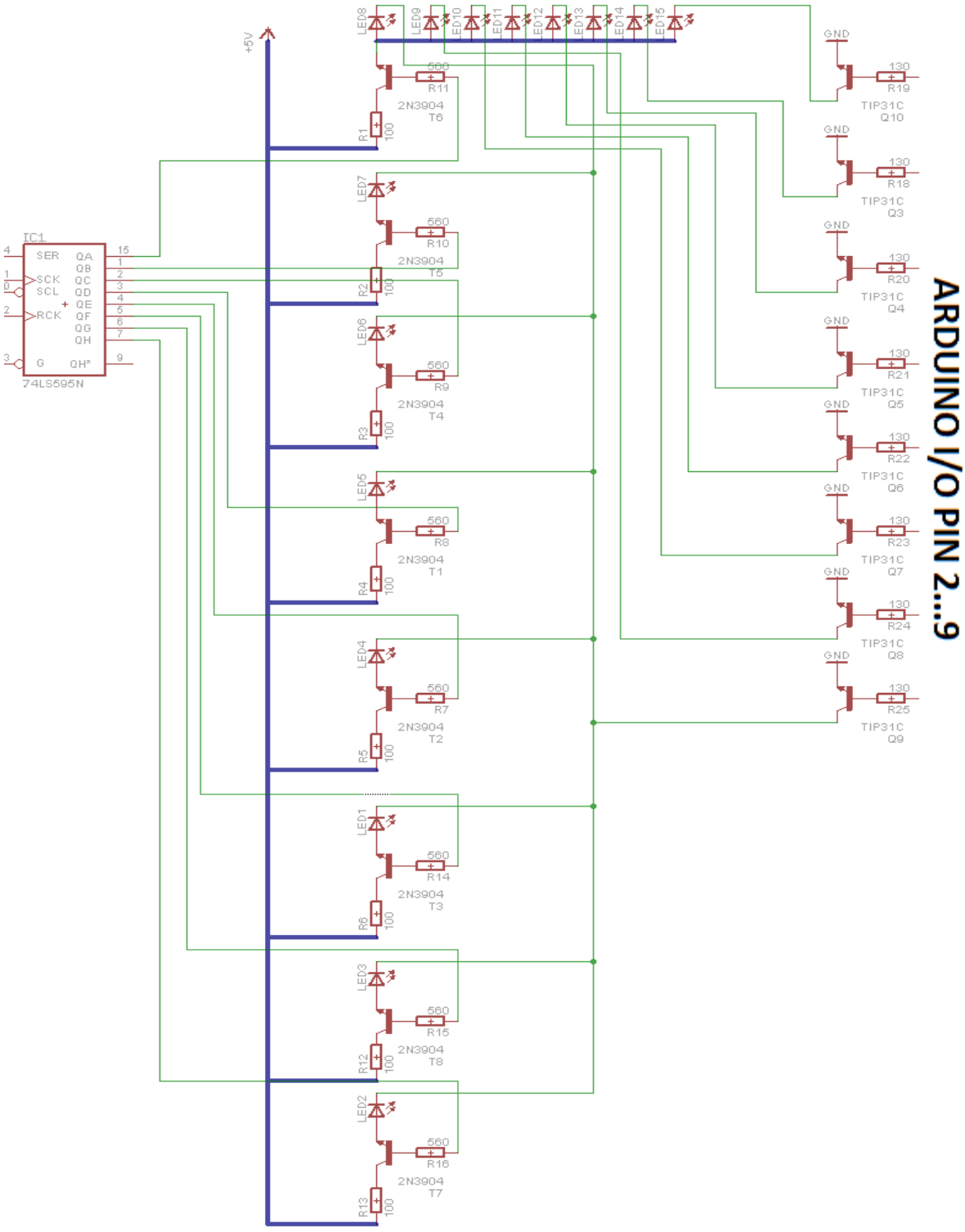
15. Arduino koduleheküg, SPI tutvustus [*WWW*] http://arduino.cc/en/Tutorial/SPIEEPROM (04.05.2014)

16. Failide jagamise keskkond, lähteprogramm [*WWW*] http://pastebin.com/75ityP4E (03.05.2014)

17. Mikrokontroller atmega328 andmeleht [*WWW*] http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet.pdf (03.05.2014)

## LISAD

## Lisa 1. Nihkeregistrite ühendus transistoride ja dioodidega



## Lisa 2. Juhtprogramm

```
#include <TimerOne.h>
#include <string.h>
#define AXIS_X 1
#define AXIS_Y 2
#define AXIS_Z 3

//viik, mis on ühendatud nihkeregistrite STCP-ga
int latchPin = 10;
//viik, mis on ühendatud nihkeregistrite SHCP-ga
int clockPin = 13;
//viik, mis on ühendatud esimese nihkeregistri DS-ga
int dataPin = 11;
//PORTB viigu number ühendatud STCP
int latchPinPORTB = latchPin - 8;
//baitide massiiv, kus hoitakse iga LED väärtust [x][y][z]
byte cube[8][8];

//hetkel sisselülitatud tasandi number
int current_layer = 0;

//katkestusega käivitatav funktsioon
void iProcess(){
  //sisselülitatud tasandile vastav viigu number kontrolleril
  int oldLayerBit = current_layer + 2;

  //tasandi numbri 1 võrra suurendamine
  current_layer++;
  if(current_layer >= 8){
    current_layer = 0;
  }
    //Kutsutakse esile spi_transfer funktsioon
  //ja määratakse selle argumentideks uue tasandi dioodide
väärtused
  latchOff();
  for (int i = 0 ; i < 8 ; i++){
    spi_transfer(cube[current_layer][i]);
  }

  //vana tasand kustutatakse
  digitalWrite(oldLayerBit, LOW);
  //SPCP kõrgeks tõstmine ja nihkeregistrite andmete saatmine
väljunditesse
  latchOn();
  //järgmise tasandi sisselülitamine
  digitalWrite(current_layer + 2, HIGH);
}

//STCP kõrgeks ja madalaks muutmise funktsioonid
```

```
void latchOn(){
  bitSet(PORTB,latchPinPORTB);
}
void latchOff(){
  bitClear(PORTB,latchPinPORTB);
}


//SPI seadistus
void setupSPI(){
  byte clr;
  SPCR |= ( (1<<SPE) | (1<<MSTR) );//SPI sisselülitamine ja
seab Arduino ülemseadmeks
  SPCR &= ~( (1<<SPR1) | (1<<SPR0) );//SPI kiiruse määramine
(kõige kiirem)
  clr=SPSR;
  clr=SPDR;
  SPSR |= (1<<SPI2X); // taktsignaali kiirus kahekordistatakse
  delay(10);
}

//SPI andmeregistrisse kirjutamine
byte spi_transfer(byte data)
{
  //argument kirjutatakse andmeregistrisse
  SPDR = data;
  //funktsioon töötab kuni andmeedastus on lõppenud
  loop_until_bit_is_set(SPSR, SPIF);
  return SPDR;
}
//seadistus funktsioon
void setup() {
  // kontroller saadab 9600 bitti sekundis
  Serial.begin(9600);

  //tasandi viikude määramine väljundiks
  for(int i = 2; i < 10; i++)
  {
    pinMode(i, OUTPUT);
  }
  //andmeside viikude määramine väljunditeks ja nende madalaks
kirjutamine
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);

  digitalWrite(latchPin,LOW);
  digitalWrite(dataPin,LOW);
  digitalWrite(clockPin,LOW);

  //SPI seadistuse esilekutsumine
```

```
  setupSPI();

  //PWM tasandi viikude taimeri aktiveerimine
  //iga 1000 mikrosekundi tagand kutsutakse esile katkestus
  Timer1.initialize(1000);
 //katkestusega kutsutakse esile iProcess funktsioon
  Timer1.attachInterrupt(iProcess);
}


void loop(){

  while (true)
  {
    effect_text(2000);
  }
}

// ================================================================
================================
//   TEXT Functions
// ================================================================
================================
// tähtede andmeid sisaldav massiiv
char font_data[128][8] = {
{ 0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00 },
// 0 :

//...

{ 0x00,  0x3C,  0x66,  0x66,  0x7E,  0x66,  0x66,  0x66 },
// 65 : A
             //           |          |
             //           |   ****   |
             //           |  **  **  |
             //           |  **  **  |
             //           |  ******  |
             //           |  **  **  |
             //           |  **  **  |
             //           |  **  **  |

  { 0x00,  0x7C,  0x66,  0x66,  0x7C,  0x66,  0x66,  0x7C },
  // 66 : B

  { 0x00,  0x3C,  0x66,  0x60,  0x60,  0x60,  0x66,  0x3C },
  // 67 : C

  { 0x00,  0x7C,  0x66,  0x66,  0x66,  0x66,  0x66,  0x7C },
  // 68 : D
```

```
{ 0x00,  0x7E,  0x60,  0x60,  0x7C,  0x60,  0x60,  0x7E },
// 69 : E

{ 0x00,  0x7E,  0x60,  0x60,  0x7C,  0x60,  0x60,  0x60 },
// 70 : F

{ 0x00,  0x3C,  0x66,  0x60,  0x60,  0x6E,  0x66,  0x3C },
// 71 : G

{ 0x00,  0x66,  0x66,  0x66,  0x7E,  0x66,  0x66,  0x66 },
// 72 : H

{ 0x00,  0x3C,  0x18,  0x18,  0x18,  0x18,  0x18,  0x3C },
// 73 : I

{ 0x00,  0x1E,  0x0C,  0x0C,  0x0C,  0x6C,  0x6C,  0x38 },
// 74 : J

{ 0x00,  0x66,  0x6C,  0x78,  0x70,  0x78,  0x6C,  0x66 },
// 75 : K

{ 0x00,  0x60,  0x60,  0x60,  0x60,  0x60,  0x60,  0x7E },
// 76 : L

{ 0x00,  0x63,  0x77,  0x7F,  0x6B,  0x63,  0x63,  0x63 },
// 77 : M

{ 0x00,  0x63,  0x73,  0x7B,  0x6F,  0x67,  0x63,  0x63 },
// 78 : N

{ 0x00,  0x3C,  0x66,  0x66,  0x66,  0x66,  0x66,  0x3C },
// 79 : O

{ 0x00,  0x7C,  0x66,  0x66,  0x66,  0x7C,  0x60,  0x60 },
// 80 : P

{ 0x00,  0x3C,  0x66,  0x66,  0x66,  0x6E,  0x3C,  0x06 },
// 81 : Q

{ 0x00,  0x7C,  0x66,  0x66,  0x7C,  0x78,  0x6C,  0x66 },
// 82 : R

{ 0x00,  0x3C,  0x66,  0x60,  0x3C,  0x06,  0x66,  0x3C },
// 83 : S

{ 0x00,  0x7E,  0x5A,  0x18,  0x18,  0x18,  0x18,  0x18 },
// 84 : T

{ 0x00,  0x66,  0x66,  0x66,  0x66,  0x66,  0x66,  0x3E },
// 85 : U
```

```
  { 0x00,  0x66,  0x66,  0x66,  0x66,  0x66,  0x3C,  0x18 },
  // 86 : V

  //...


  { 0x00,  0x00,  0x00,  0x3C,  0x0C,  0x18,  0x30,  0x3C },
  // 122 : z

  { 0x00,  0x0E,  0x18,  0x18,  0x30,  0x18,  0x18,  0x0E },
  // 123 : {

  { 0x00,  0x18,  0x18,  0x18,  0x00,  0x18,  0x18,  0x18 },
  // 124 : |

  { 0x00,  0x70,  0x18,  0x18,  0x0C,  0x18,  0x18,  0x70 },
  // 125 : }

  { 0x00,  0x00,  0x00,  0x3A,  0x6C,  0x00,  0x00,  0x00 },
  // 126 : ~

  { 0x00,  0x08,  0x1C,  0x36,  0x63,  0x41,  0x41,  0x7F }
 // 127 : □

};

// Kuvatav tekst
const int charNum = 10;//teksti tähemärkide arv
// valitakse tähemäkide järjekorranumbrid vastavalt soovitud t
ekstile
int string[charNum] = {76,69,68,0,75,85,85,80,33,0};//"LED
KUUP! "
// Teksti kuvamis funktsioon
void effect_text(int delayt){
  fill(0x00);
  for (int ltr = 0; ltr < charNum; ltr++){//Iga tähemärgi
jaoks
    for(int dist = 0; dist < 8; dist++) { //liigutab tähe
mööda kuupi ette
      fill(0x00);//blank last row
      int rev = 0;
      for (int rw = 7; rw >= 0; rw--) {//ridade kopeerimine
         // Igale dioodile omistatakse vastav väärtus, bitswap
pöörab baidi tagurpidi
         cube[rev][dist] = bitswap(font_data[string[ltr]][rw])
;
         rev++;
      }
      delay_ms(delayt);//teksti kuvamise kiirus
    }
  }
```

```
}

unsigned char bitswap (unsigned char x)//Reverses a byte (so
letters aren't backwards);
{  byte result;

    asm("mov __tmp_reg__, %[in] \n\t"
      "lsl __tmp_reg__  \n\t"   /* shift out high bit to carry
*/
      "ror %[out] \n\t"  /* rotate carry __tmp_reg__to low bit
(eventually) */
      "lsl __tmp_reg__  \n\t"   /* 2 */
      "ror %[out] \n\t"
      "lsl __tmp_reg__  \n\t"   /* 3 */
      "ror %[out] \n\t"
      "lsl __tmp_reg__  \n\t"   /* 4 */
      "ror %[out] \n\t"

      "lsl __tmp_reg__  \n\t"   /* 5 */
      "ror %[out] \n\t"
      "lsl __tmp_reg__  \n\t"   /* 6 */
      "ror %[out] \n\t"
      "lsl __tmp_reg__  \n\t"   /* 7 */
      "ror %[out] \n\t"
      "lsl __tmp_reg__  \n\t"   /* 8 */
      "ror %[out] \n\t"
      : [out] "=r" (result) : [in] "r" (x));
      return(result);
}
void fill (unsigned char pattern)
{
       int z;
       int y;
       for (z=0;z<8;z++)
       {
              for (y=0;y<8;y++)
              {
                     cube[z][y] = pattern;
              }
       }
}
void delay_ms(uint16_t x)
{
  uint8_t y, z;
  for ( ; x > 0 ; x--){
    for ( y = 0 ; y < 90 ; y++){
      for ( z = 0 ; z < 6 ; z++){
        asm volatile ("nop");
      }
    }
  }
}
```