Ilkka Saarelainen

KAUKO-OHJATTAVA MONIVÄRIVALAISIN

Tietotekniikan koulutusohjelma

2013

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KAUKO-OHJATTAVA MONIVÄRIVALAISIN

Saarelainen, Ilkka
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Toukokuu 2013
Ohjaaja: Ylikoski, Mauri
Sivumäärä: 49
Liitteitä: 15

Asiasanat: kauko-ohjattava, valaisin, bluetooth

Tämän opinnäytetyön tarkoituksena oli kehittää valaisin, joka on kilpailukykyinen markkinoilla olevien vastaavien tuotteiden kanssa. Markkinoilla oleviin tuotteisiin verraten tämän järjestelmän oli tarkoitus olla helposti laajennettavissa ja muokattavissa.

Järjestelmään oli tarkoituksena toteuttaa mahdollisimman yksinkertaisesti langaton yhteys ohjaavan mobiililaitteen ja valaisinyksikön välille. Bluetooth osoittautui helpoksi ja melko halvaksi tavaksi toteuttaa yksinkertainen langaton sarjaliikenne.

Järjestelmässä käytettiin hyväksi olemassa olevia komponentteja ja tekniikoita. Olemassa olevan tekniikan ja valmiiden komponenttien, kuten Bluetooth-moduulin, hyödyntäminen oli välttämätöntä järjestelmän pitämiseksi kohtuullisen yksinkertaisena.

Järjestelmä ei ole välttämättä vielä riittävän valmis myytäväksi tuotteeksi, mutta päätoiminnot siinä kuitenkin jo toimivat hyvin. Järjestelmän tarkoituksena olikin pysyä täysin avoimena ja kenen tahansa helposti rakennettavissa.

REMOTE CONTROLLED MULTI-COLOR LIGHT SYSTEM

Saarelainen, Ilkka
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
May 2013
Supervisor: Ylikoski, Mauri
Number of pages: 49
Appendices: 15

Keywords: remote control, light system, bluetooth

The purpose of this thesis was to develop a light system that is able to compete with other similar products on the market. Compared to other products on the market, this system was meant to be easily extensible and customizable.

The system was meant to be equipped with as simple as possible wireless connection between the controlling mobile device and the light device. Bluetooth saw to be easy, widely supported and quite cheap way to implement a simple wireless serial connection.

There were utilized existing components and techniques in this system. Utilizing existing techniques and components, like the Bluetooth module, were necessary to keep the system as simple as possible.

The system might not be complete enough to get it to the market, but all the main functions are already working well. The purpose of this system was to stay completely open and to be easily buildable by anyone.

SISÄLLYS

SYMBOLIT JA LYHENTEET

ASCII	American Standard Code for Information Interchange - 8-bittinen merkistö
ADT	Android Developer Tools - Android-kehitysympäristö
AVR	Atmel:n mikro-ohjainsarja
EEPROM	Electronically Erasable Programmable Read-Only Memory - Haihtumaton puolijohdemuisti
CIELAB	Kansainvälisen valaistuskomission standardoima väriavaruus
LED	Light Emitting Diode - hohtodiodi
PDIP	Plastic Dual In-line Package - elektroniikkakomponenttien kotelotyyppi
PWM	Pulse-Width Modulation - pulssinleveysmodulaatio
RGB	Red / Green / Blue - lyhenne kolmesta värikomponentista
TQFP	Thin Quad Flat Pack - elektroniikkakomponenttien kotelotyyppi
UART	Universal Asynchronous Receiver-Transmitter - asynkroninen sarjaliikenne-portti
USART	Universal Synchronous/Asynchronous Receiver-Transmitter - synkroninen tai asynkroninen sarjaliikenne-portti

1 JOHDANTO

LED-valojen valmistuskustannusten laskiessa markkinoille on tullut paljon erilaisia LED-valaisimia. Valkoisten valaisinten lisäksi LED-tekniikalla on helppo toteuttaa erivärisiä ja väriä vaihtavia tunnelma- ja koristevaloja.

Useimpia markkinoilla olevia väriä vaihtavia valaisimia ohjataan erillisillä kauko-ohjaimilla. Kauko-ohjaimiin liittyy kuitenkin paljon erilaisia rajoitteita. Jos esimerkiksi samassa tilassa on useita kauko-ohjaimella toimivia valaisimia, yhden tietyn valaisimen ohjaaminen saattaa olla hankalaa. Kauko-ohjaimella voidaan yleensä valita vain hyvin rajallinen määrä erilaisia värisävyjä. Usein valaisimista löytyy muutama automaattisesti väriä vaihtava ohjelma, joilla valon saa esimerkiksi vilkkumaan tai vaihtamaan väriä vähitellen.

Tämän työn tarkoituksena on kehittää valaisin, jossa edellä mainittuja rajoitteita ei ole, ja jonka toimintaan käyttäjä voi itse vaikuttaa omalla mobiililaitteellaan.

2 TOIMINNALLISET VAATIMUKSET

Valaisinta ohjataan mobiililaitteella, jossa on Android-käyttöjärjestelmä. Valaisimessa tulee olla kolmivärinen LED-valo. Nämä kolme värikomponenttia ovat punainen, vihreä ja sininen. Jokaisen värikomponentin kirkkautta voidaan ohjata erikseen, eri kirkkausasteiden yhdistelmistä voidaan luoda erilaisia värisävyjä.

Värin asettamisen lisäksi valaisimeen voidaan asettaa suoritettavaksi käyttäjän määrittelemiä valo-ohjelmia. Valo-ohjelma koostuu käskyistä, joita noudattamalla valaisin vaihtaa valon väriä. Valo-ohjelma määrittelee jokaisen värikomponentin kirkkauden muutokset erikseen toisista värikomponenteista riippumatta.

Puhelimen ja valaisimen välinen yhteys toteutetaan Bluetooth-yhteydellä. Valaisimessa olevan mikro-ohjaimen sovellus voidaan päivittää puhelimen sovelluksen avulla.

Valaisin tarvitsee tasavirtalähteen, jonka jännite on noin 4-12V. Valaisimen ohjainosa ja varsinainen valo voivat käyttää erillisiä virtalähteitä, eikä virtalähteiden jännite tarvitse olla sama.

Järjestelmän kaikkien osien on tarkoitus olla mahdollisimman helposti laajennettavissa ja muokattavissa.

3 KEHITYSYMPÄRISTÖT

Kauko-ohjainsovelluksen kehittämiseen käytin Eclipse Juno 4.2.1 kehitysympäristöä, johon olin asentanut Android-sovelluskehitykseen tarkoitetun ADT-lisäosan (Android Tools 2013). Sovelluksen toimivuutta testasin Samsung Galaxy S III -puhelimella.

Kauko-ohjainsovellus hyödyntää kolmansien osapuolten kehittämiä avoimen lähdekoodin Android Color Picker (Yukuku 2013) ja DragSortListView (Bauer 2013) -kirjastoja.

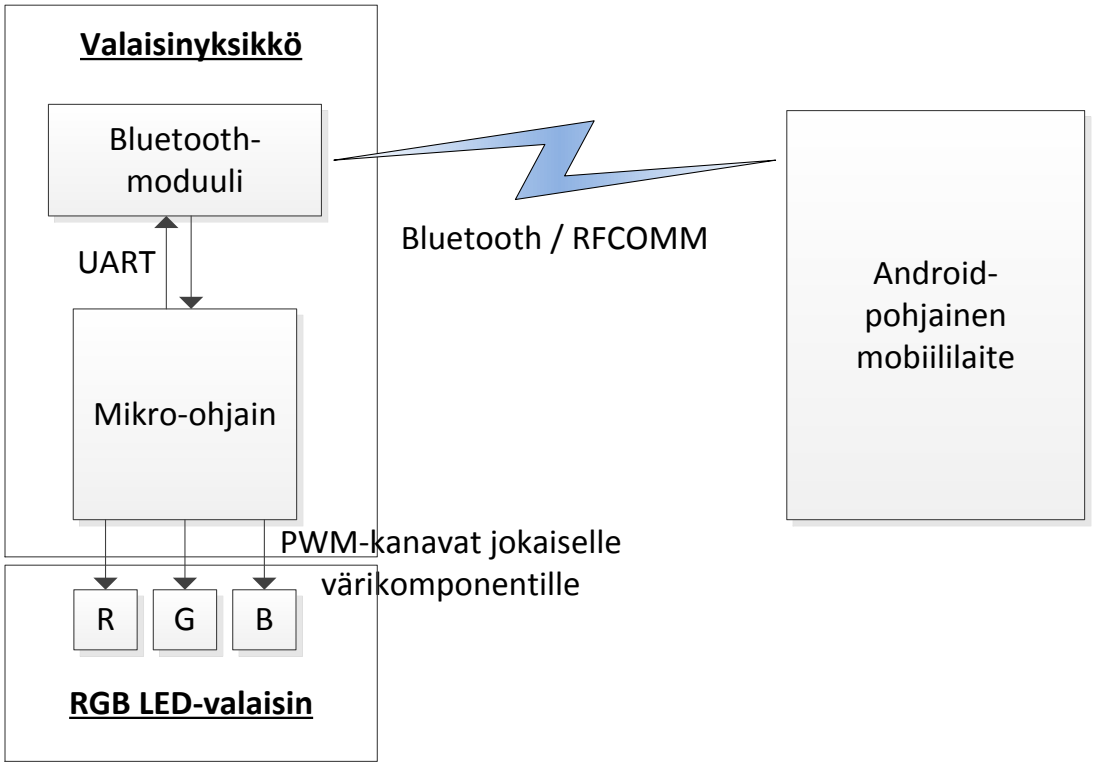
Mikro-ohjaimen sovelluksen kehittämiseen käytin Atmelin tarjoamaa AtmelStudio 6-kehitysympäristöä.

Piirilevyn suunnitteluun käytin Eagle 6.3.0 -suunnitteluohjelmaa.

4 VALAISINYKSIKÖN LAITTEISTO

4.1 Järjestelmän kokoonpano

Valaisinyksikkö koostuu mikro-ohjaimesta ja Bluetooth-moduulista. Järjestelmää ohjaava mobiililaite on yhteydessä valaisinyksikköön Bluetooth-yhteyden avulla. Kuva 1 esittää järjestelmän kokoonpanoa.



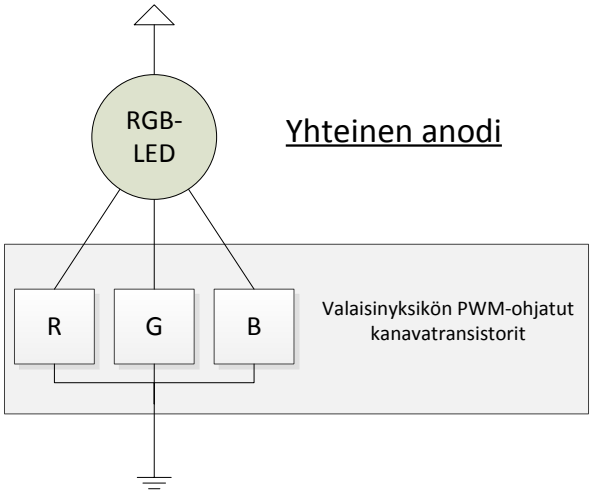
Kuva 1. Järjestelmän kokoonpano.

Valaisinyksikkö ei sisällä varsinaista valonlähdettä, vaan siihen pitää kytkeä erillinen monivärinen LED-valo, jossa on kolme värikomponenttia: punainen, vihreä ja sininen.

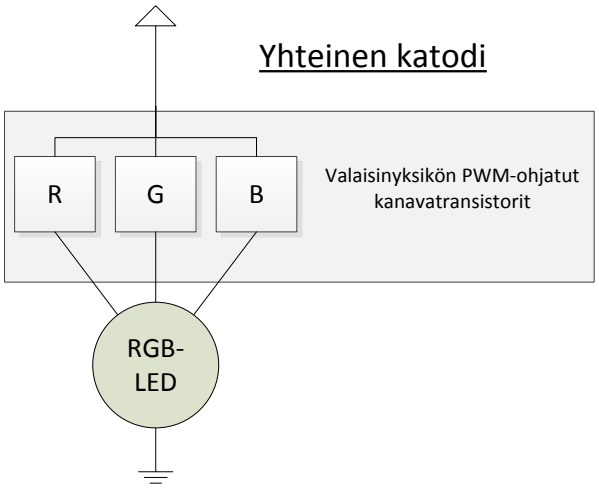
Kaikkien moniväristen LED-valojen sisäiset kytkennät eivät ole samanlaisia. Yleensä monivärisissä LED-valoissa värikomponentit on kytketty niin, että niillä on joko yhteinen anodi tai yhteinen katodi. Näitä kahta kytkentätapaa harvinaisempi on kytkentä, jossa on omat anodit ja katodit jokaiselle värikomponentille. Jälkimmäistä tapaa ei yleensä käytetä, koska se ei usein tuo mitään etua ja sen takia valoon tulee kaksi lii-

täntää lisää. Värikomponentit ovat käytännössä erillisiä LED-valoja, jotka on asetettu erittäin lähelle toisiaan.

Valaisinyksikköön voidaan kytkeä millä tahansa edellä mainituilla kytkentätavoilla toteutettu LED-valo. LED:n sisäinen kytkentä pitää ottaa huomioon liitettäessä valoa valaisinyksikköön. Jos värikomponenteilla on yhteinen anodi, valaisinyksikkö pitää kytkeä LED:n ja maapotentiaalin väliin (Kuva 2). Jos värikomponenteilla on yhteinen katodi, valaisinyksikkö pitää kytkeä virtalähteen ja LED:n väliin (Kuva 3). Jos jokaisella värikomponentilla on erilliset anodit ja katodit, valaisinyksikkö voidaan kytkeä kummalla tahansa edellä mainituista tavoista.



Kuva 2. LED:n kytkentä: yhteinen anodi.



Kuva 3. LED:n kytkentä: yhteinen katodi.

Valaisinyksikön ja LED-valon ei tarvitse käyttää samaa virtalähdettä, mutta virtalähteiden maapotentiaalit pitää kuitenkin kytkeä yhteen kanavatransistoreiden oikean toiminnan takaamiseksi. Valon virtalähteen jännite voi olla suurempi kuin valaisinyksikön virtalähteen.

Valaisinyksikön mikro-ohjaimessa on käytössä kolme PWM eli pulssileveysmoduloitua lähtöä, joista jokainen on kytketty erillisiin N-kanavatransistoreihin. Jokainen transistori ohjaa yhtä värikomponenttia. Kuvien 2 ja 3 R, G ja B kuvaavat näitä transistoreja.

Transistoreina käytetään IRLI-620 N-kanavatransistoreja. Kanavatransistori sopii tähän tarkoitukseen hyvin, koska sitä ohjataan jännitteellä (Inkinen, Manninen & Tuohi 2009, 581). Ohjattava virta voi olla maksimissaan 4 ampeeria (International Rectifier). 4 ampeeria värikomponenttia kohden riittää hyvin useimpien LED-valojen ohjaamiseen. Useimmat moniväriset LED-valonauhat tarvitsevat 12V jännitteen toimiakseen. Näistä tiedoista voidaan laskea 12V:n jännitteellä toimivan moniväri-LED:n suurin teho, jota tällä kokoonpanolla voidaan ohjata. Teho lasketaan seuraavalla yhtälöllä.

$$P = U \times I = 12V \times 4A = 48W$$

48 wattia on suurin teho yhtä värikomponenttia kohden. Suurin kokonaisteho on näin ollen $3 \times 48W = 144W$. Käytännössä 4A:n virtaa ei välttämättä pystytä ohjaamaan kanavatransistorien lämpiämisen vuoksi. Lämpiämisen haittojen minimoimiseksi kanavatransistoreihin voisi kiinnittää jäähdytyslevyn, tai vaihtaa kanavatransistorit suurempaa virtaa kestävään malliin.

Transistoreja käytetään ikään kuin nopeina kytkiminä niin, että ne ovat hetkellisesti joko kokonaan auki, tai kokonaan kiinni. Kanavatransistoreilla ohjauksesta on selitetty tarkemmin kappaleessa 4.3.3 PWM.

Valaisinyksikkö ei rajoita hetkellistä värikomponenttien läpi kulkevaa virtaa, paitsi transistorien sisäisen vastuksen verran. Tästä johtuen LED-valon täytyy itse rajoittaa

virran määrä esimerkiksi etuvastuksella. Yleensä valoissa onkin erillinen etuvastus jokaiselle värikomponentille.

4.2 Laitteiston kehitysvaiheet

Kehitin valaisinyksikön laitteiston ensin koekytkentälevylle, jossa varmistin laitteiston toimivuuden (Liite 3). Kun olin todennut laitteiston toimivaksi, tein kytkentöjen pohjalta kytkentäkaavion (Liite 1). Kytkentäkaavion pohjalta tein piirilevykaavion (Liite 2).

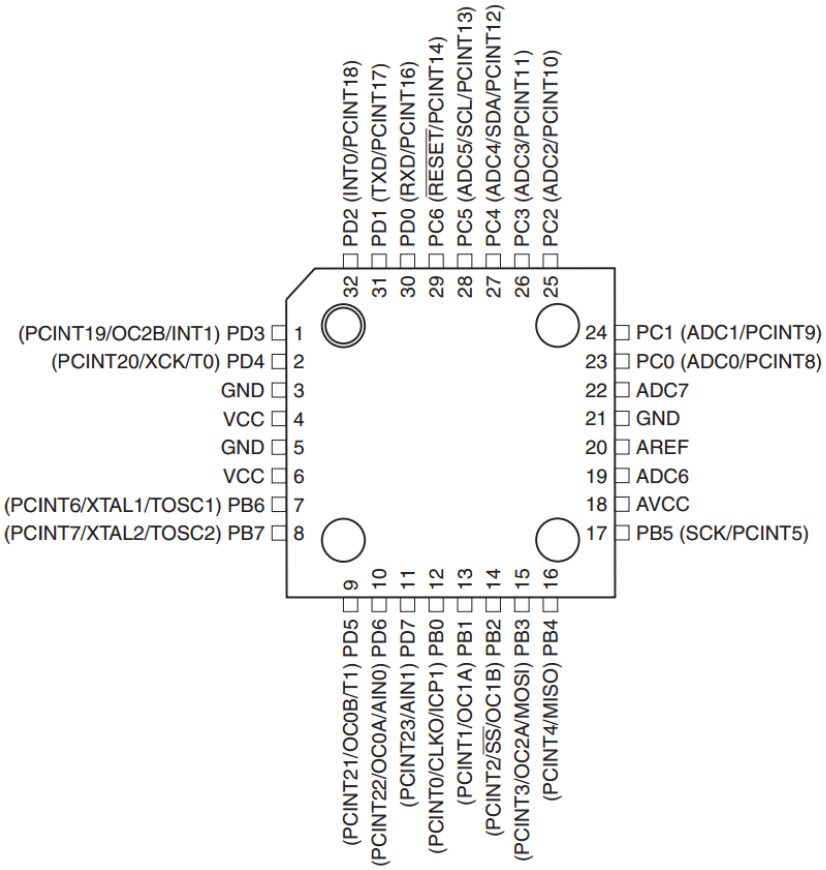
Koekytkentälevyllä käytin mikro-ohjaimen PDIP-koteloitua versiota, mutta lopulliseen piirilevyyn valitsin TQFP-koteloidun version, koska se mahtuu huomattavasti pienempään tilaan. Suunnittelin piirilevyn alun perin toteutettavaksi yksipuoliselle piirilevylle, mutta lopullinen toteutus on kuitenkin tehty kaksipuolisena piirilevynä. Tästä johtuen piirilevyn tilaa ei ole käytetty kovin hyvin hyödyksi.

Lopullisen piirilevyn tilasin Seeedstudio-verkkokaupasta, josta proto-piirilevyjen tilaus onnistuu vaivattomasti muutaman viikon toimitusajalla. Tilaamani piirilevy (Liite 4) ei vastaa täysin Liitteen 2 kaaviota, koska kaavioon on korjattu muutamia virheitä, joita tilaamassani piirilevyssä oli.

4.3 Mikro-ohjain

Valaisimen mikro-ohjaimena toimii Atmel AVR ATMega328. Valitsin tämän mikro-ohjaimen, koska siinä on kaikki tähän järjestelmään tarvittavat ominaisuudet. Seuraavassa listassa on lueteltuna valaisimen kannalta oleelliset mikro-ohjaimen ominaisuudet.

- USART (Universal Synchronous/Asynchronous Receiver-Transmitter), eli sarjaliikenne-portti
- 32kt flash-ohjelmamuistia
- 2kt käyttömuistia
- kuusi PWM-kanavaa
- kaksi 8-bittistä ja yksi 16-bittinen laskuri/ajastin
- 1,8 – 5,5 V käyttöjännite
- 0,2mA virrankulutus @ 1MHz, 1,85V
- sisäinen kalibroitu RC-oskillaattori
- 23 ohjelmoitavaa I/O–linjaa
- 32-jalkaa (TQFP-kotelo)
- itseohjelmointi ja erillinen käynnistysohjelma
- -40°C – 85°C toimintalämpötila



Kuva 4. ATMega328 TQFP-kotelon jalkajärjestys. (Atmel 2009, 2)

4.3.1 USART

USART:ia eli sarjaliikenneyhteyttä tarvitaan mikro-ohjaimen liittämiseen Bluetooth-moduuliin. Vaikka mikro-ohjaimessa onkin tuki synkroniselle sarjaliikenteelle, tässä järjestelmässä riittää asynkroninen yhteys, eli sarjaliikenneyhteydestä voidaan käyttää lyhennettä UART. UART on yleisesti käytössä oleva standardi rinnakkaismuotoisen datan sarjallistamiseen ja vastaavasti sarjamuotoisen datan rinnakkaistamiseen (Society of Robots 2007). Tämän standardin käyttäminen mahdollistaa sen, että valaisimen käyttämä Bluetooth-tiedonsiirtoyhteys voidaan helposti vaihtaa mihin tahansa tiedonsiirtotiehen, joka voidaan liittää mikro-ohjaimen UART-liitäntään. UART:n käyttäminen mahdollistaa järjestelmän helpon jatkokehityksen, muokattavuuden ja laajennettavuuden ilman isoja muutoksia.

UART-liitäntään liittyy kaksi pinniä mikro-ohjaimessa: TXD ja RXD (pinnit 31 ja 30, Kuva 4). TXD-pinniä käytetään datan lähettämiseen ja RXD-pinniä datan vastaanottoon.

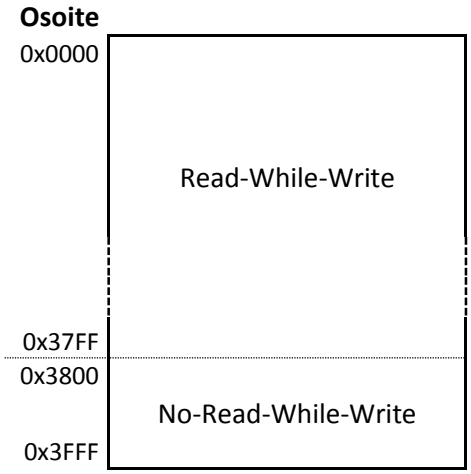
Tässä työssä sarjaliikenneyhteyttä käytetään 9600 bitin sekuntinopeudella. Valitsin näinkin hitaan nopeuden, koska yhteyden yli ei ole tarvetta siirtää suuria määriä dataa, ja toisaalta mikro-ohjaimen pitää tietyissä tapauksissa ehtiä suorittamaan melko paljon komentoja jokaisen vastaanotetun tavun välissä.

Mikro-ohjaimessa UART:n nopeus määritellään UBRR-rekisterin avulla. Mikro-ohjaimen datalehdessä on kattava lista rekisteriin kirjoitettavien arvojen ja käytettävien kellotaajuuksien yhdistelmistä (Atmel 2009, 197-200). Taulukosta on helppo valita oikea rekisterin arvo. Tässä tapauksessa kun kellotaajuus on 8 MHz ja nopeudeksi halutaan 9600 bittiä sekunnissa, UBRR-rekisterin arvoksi asetetaan 103. Tämän lisäksi UCSR0A-rekisteristä pitää asettaa U2X0-bitti ykköseksi, joka kaksinkertaistaa nopeuden. Toinen vaihtoehto olisi asettaa UBRR-rekisteriin arvo 51 ja jättää U2X0-bitti nollaksi, mutta tämä ei jostain syystä toiminut. Toimimattomuus saattaa johtua RC-oskillaattorin epätarkkuudesta.

4.3.2 Flash-muisti

Mikro-ohjaimen 32 kilotavun flash-muisti on kokonsa puolesta jopa tarpeettoman suuri valaisimeen. Käyttämättömästä ohjelmamuistista ei kuitenkaan ole mitään haittaa ja se tuo lisämahdollisuuksia laajennettavuuden ja jatkokehityksen kannalta.

Flash-ohjelmamuistin lopussa on neljän kilotavun muistialue, johon voidaan ohjelmoida käynnistyslataaja eli bootloader. Tästä muistialueesta käytetään datalehdessä nimeä NRWW (No-Read-While-Write) ja muusta muistialueesta RWW (Read-While-Write). Kuva 5 esittää muistin jaottelua RWW- ja NRWW-osioiden välillä.



Kuva 5. Mikro-ohjaimen muistin jako.

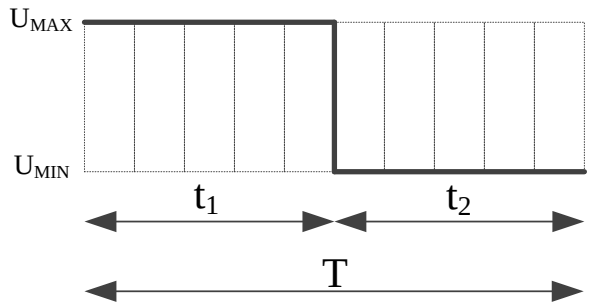
Käynnistyslataaja sijaitsee aina NRWW-muistialueella, jos käynnistyslataajaa halutaan käyttää. Käynnistyslataajan maksimikoko on neljä kilotavua. Käynnistyslataajan käyttöön varattava muistimäärä asetetaan mikro-ohjaimen BOOTSZ0- ja BOOTSZ1-Fuse-biteillä. Käynnistyslataajalle voidaan varata 512, 1024, 2048 tai 4096 kilotavua muistia. Käynnistyslataajalle varattava alue sijaitsee aina koko muistialueen lopussa.

Käynnistyslataaja otetaan käyttöön nollaamalla BOOTRST-Fuse-bitti. Tämä siirtää Reset-vektorin 0x0000-osoitteesta 0x3800-osoitteeseen. Reset-vektori on aina ensimmäinen suoritettava komento mikro-ohjaimen käynnistyksen tai uudelleenkäynnistyksen jälkeen.

Käynnistyslataajan muistialueen alussa on samanlaiset keskeytysvektorit kuin RWW-muistinkin alussa. Tämä mahdollistaa keskeytysten käytön käynnistyslataajassa. Käynnistyslataajan keskeytysvektorit voisi ottaa käyttöön asettamalla IVSEL-bitti MCUCR-rekisterissä. Tämän järjestelmän käynnistysohjelmassa ei kuitenkaan käytetä keskeytyksiä.

4.3.3 PWM

PWM eli pulssinleveysmodulaatio on signaalin modulointitapa, jossa signaalin tasoa muutetaan nopeasti minimijännitteen (U_{MIN}) ja maksimijännitteen (U_{MAX}) välillä. Taajuus on vakio, mutta pulssisuhdetta muutetaan. Pulssisuhteella tarkoitetaan puolijaksojen t_1 ja t_2 suhdetta. Kuva 6 esittää jakson T rakennetta.



Kuva 6. PWM-signaalin jakson rakenne.

Pulssisuhde ilmaistaan usein prosenttilukuna, joka kertoo t_1:n osuuden jaksosta T. Pulssisuhde D voidaan laskea seuraavalla yhtälöllä:

$$D = \frac{t_1}{T} \times 100\%$$

Hehkulampuista poiketen LED:ien kirkkautta ei voi säätää jännitettä muuttamalla, koska LED:n palaessa sen yli vaikuttaa aina sille ominainen vakio kynnysjännite. LED:n kirkkauteen voi vaikuttaa siis ainoastaan rajoittamalla sen läpi kulkevaa virtaa. Virran rajoittaminen onnistuu helposti vastuksella, mutta vastusarvon muuttaminen mikro-ohjaimella ei ole kovin helposti toteutettavissa.

Pulssinleveysmodulaation avulla hohtodiodin näennäistä kirkkautta voidaan kuitenkin säätää vaivattomasti. Kirkkauden säätö perustuu siihen, että LED:ä vilkutetaan niin nopeasti, että silmä ei erota vilkkumista (WaitingForFriday 2010). Pulssisuhteen pienentyessä valo näyttää himmenevän, koska valo palaa vähemmän aikaa, kuin se on pois päältä.

Mikro-ohjaimessa on sisäänrakennettu PWM-toiminto, eli pinnien tilaa ei tarvitse muuttaa ohjelmallisesti. Mikro-ohjaimen pulssigeneraattori perustuu laskuriin, jonka arvoa mikro-ohjain vertaa jatkuvasti ohjelmallisesti asetettuun referenssiarvoon. Laskurin arvo on TCTNn-rekisterissä ja siihen verrattava arvo asetetaan OCRnx-rekisteriin.

Edellä mainittujen rekisterien nimissä n = laskurin numero (0..2) ja x = vertailurekisterin kirjaintunnus (A tai B). Mikro-ohjaimen laskurit 0 ja 2 ovat 8-bittisiä, eli TCTNn-rekisterien maksimiarvo on 2^8. Laskuri 1 on 12-bittinen, mutta tässä tapauksessa sitäkin käytetään 8-bittisenä.

Tässä tapauksessa käytössä on Fast PWM Mode, jossa laskurin arvo kasvaa nollasta maksimiin, jonka jälkeen se automaattisesti aloittaa taas nollasta. Laskurin ollessa yhtä suuri kuin vertailuarvo, sitä vastaava pinni OCnx nollaantuu, eli jännite putoaa maatasoon, ja vastaavasti nousee käyttöjännitteen tasoon laskurin nollaantuessa.

Koska laskuri ja vertailuarvo ovat 8-bittisiä, erilaisia pulssisuhteita on olemassa $2^8 = 256$. Pulssisuhde on suoraan verrannollinen värikomponentin kirkkauteen. Koska jokaisen värikomponentin kirkkautta voidaan säätää erikseen, voidaan laskea erilaisten värisävyjen määrä.

$$Värisävyjen\ määrä = 2^{8+8+8} = 2^{24} = 16\,777\,216$$

Värikomponenttien kirkkauden säätö on lineaarinen niin, että OCRnx-rekisterin arvo 0 vastaa kokonaan pois päältä olevaa ja 255 vastaa täyttä kirkkautta. On kuitenkin huomattava, että ihmisen näköaistin herkkyys ei ole lineaarinen. Ihmissilmän valoherkkyys on suurimmillaan luminanssin ollessa pieni (Neuroelec 2012). Kirkkauden säätö saadaan näyttämään suunnilleen lineaariselta muuntamalla väriarvot CIELAB-

väriavaruuteen (Neuroelec 2012). Tässä järjestelmässä muunnoksesta käytetään nimitystä Color Correction, eli värin korjaus.
Muunnoksen yhteydessä kirkkauden tarkkuutta pitää laskea kahdeksasta bitistä viiteen bittiin, jotta lineaarisuus saadaan riittävästi poistettua. Muunnos tapahtuu seuraavalla yhtälöllä (Neuroelec 2012):

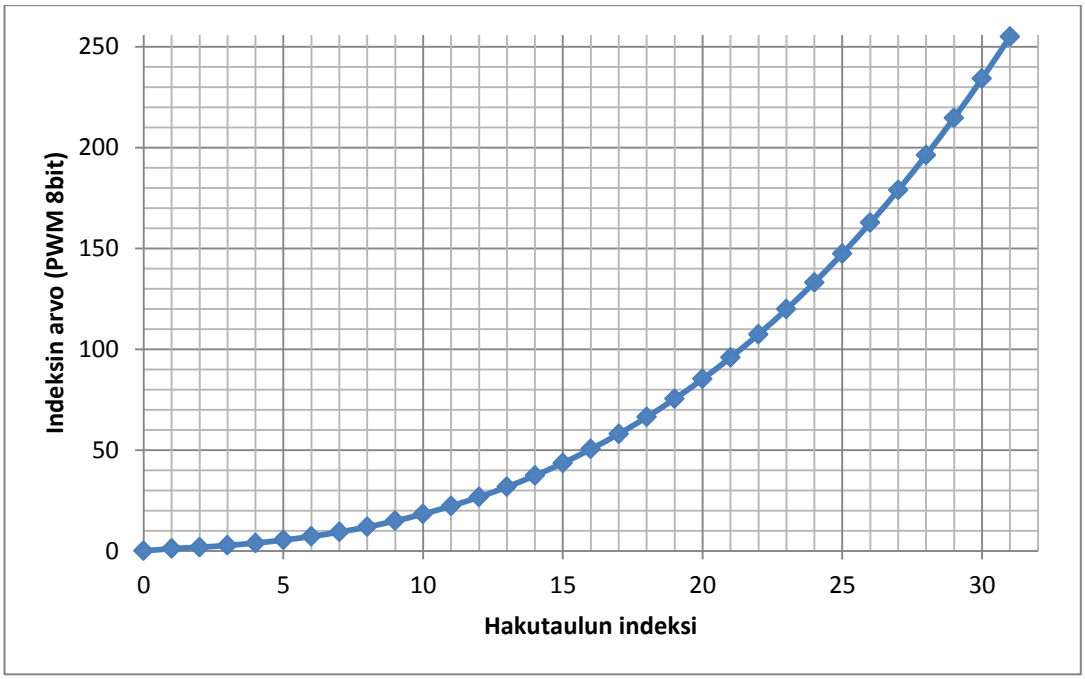
$$PWM8 = \left(\frac{PWM5 \times 100 + 16}{116}\right)^3 \times PWM8_{MAX}$$

Jossa PWM8 = OCRnx-rekisteriin kirjoitettava 8-bittinen arvo, joka määrää värikomponentin kirkkauden, PWM5 = viiden bitin tarkkuuteen pudotettu 8-bittinen arvo, $PWM8_{MAX}$ = OCRnx-rekisterin suurin mahdollinen arvo, eli 2^8-1.

Muunnosta ei ole järkevää tehdä mikro-ohjaimella yhtälön avulla, vaan huomattavasti nopeampi vaihtoehto on käyttää hakutaulua, johon muunnokset on laskettu valmiiksi. Käytännössä muunnos tapahtuu mikro-ohjaimella seuraavasti:

1. halutusta lineaariasteikon kirkkausarvosta pudotetaan 3 vähiten merkitsevää bittiä pois (siirto oikealle)
2. saatua tulosta käytetään hakutaulun indeksinä
3. kyseisestä indeksistä löytyvä 8-bittinen luku kirjoitetaan OCRnx-rekisteriin

Kuva 7 esittää hakutaulun sisältöä. Jokainen piste vastaa yhtä arvoa taulussa. Kuvan käyrä vastaa ihmissilmän herkkyyden epälineaarisuutta.



Kuva 7. CIELAB-hakutaulun arvot.

Koska muunnoksen yhteydessä tarkkuutta joudutaan laskemaan viiteen bittiin, värisävyjen määrä vähenee huomattavasti. Jokaista värikomponenttia kohden on siis käytössä vain 32 eri kirkkausastetta.

$$Värisävyjen\ määrä = 2^{5+5+5} = 2^{15} = 32\,768$$

4.3.4 Ajastetut keskeytykset

Valo-ohjelmien komentoja suoritetaan tasaisin väliajoin. Komentojen suorituksen välissä mikro-ohjaimen pitää pystyä suorittamaan muuta osaa ohjelmasta, joten keskeytykset on ainoa mahdollisuus toteuttaa tämä toiminnallisuus.

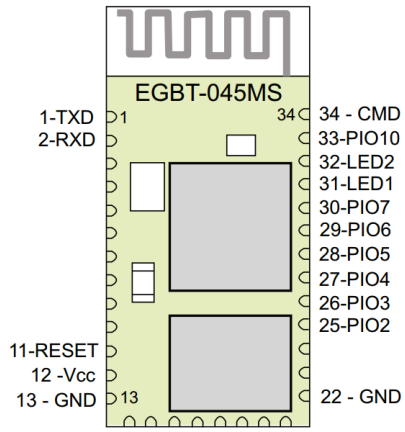
Valo-ohjaimen keskeytyksiin käytetään mikro-ohjaimen Timer/Counter2-laskuria. Laskuri kasvattaa automaattisesti TCNT2-rekisterin arvoa, ja vertaa sitä jatkuvasti vertailurekisterin OCR2x arvoon. Rekisterin nimessä x = vertailurekisterin kirjaintunnus (A tai B). Luvussa 6.2 (Ohjelman suoritus) on selostettu tarkemmin, mitä vertailuarvoa tässä tapauksessa käytetään.

Laskurin TCNT2-rekisteri on asetettu nollautumaan, kun sen arvo ja vertailuarvo ovat yhtäläisiä. Tämä tapahtuu asettamalla laskurin TCCR2A ohjausrekisterin WGM21-bitti.

TCNT2-rekisterin ja vertailuarvon ollessa yhtäläisiä laskuri laukaisee myös keskeytyksen, jonka avulla voidaan suorittaa ajastetusti erilaisia toimintoja; tässä tapauksessa valo-ohjelmien komentoja. Keskeytys otetaan käyttöön asettamalla TIMSK2-rekisterin OCIE2A-bitti.

4.4 Bluetooth-moduuli

Bluetooth-moduuli on monista elektroniikkatarvikkeita myyvästä verkkokaupasta löytyvä EGBT-045MS-moduuli.



Kuva 8. Bluetooth-moduulin jalkajärjestys. (e-Gizmo Mechatronix Central)

Moduuli vaatii toimiakseen 3,3V:n käyttöjännitteen. Moduulin TXD- ja RXD pinnit liitetään suoraan mikro-ohjaimen vastaaviin pinneihin. Moduuli voi toimia master- tai slave-tilassa, eli se voi joko itse muodostaa yhteyden, tai joku toinen laite voi muodostaa yhteyden siihen. Tässä tapauksessa moduuli toimii vain slave-tilassa.

Moduulin hallinta tapahtuu niin kutsutuilla AT-komennoilla. Ennen AT-komentojen käyttöä moduuli pitää asettaa komentoja hyväksyvään tilaan. Tämä tapahtuu syöttämällä CMD-pinniin (pinni 34) käyttöjännite.

Tässä työssä käytetään vain kahta AT-komentoa. Komentojen avulla käyttäjä voi vaihtaa moduulin Bluetooth-nimen ja -salasanan. Käytännössä mobiililaite lähettää halutun salasanan tai nimen mikro-ohjaimelle. Mikro-ohjaimen vastaanotettua uusi salasana tai nimi, se vetää Bluetooth-moduulin CMD-pinnin ylös ja lähettää UART-portin kautta AT-komennon, joka vaihtaa moduulin nimen tai salasanan. Jos mikro-ohjain ei vetäisi Bluetooth-moduulin CMD-pinniä ylös, lähetetty AT-komento lähtisi takaisin mobiililaitteelle.

Komento	**Vastaus**
AT+NAME=<nimi>\r\n	+NAME:<nimi>\r\nOK\r\n
AT+PSWD=<salasana>\r\n	+PSWD:<salasana>\r\nOK\r\n

Taulukko 1 on mikro-ohjaimen käyttämät AT-komennot ja Bluetooth-moduulin lähettämät vastaukset komentoihin.

Komento	**Vastaus**
AT+NAME=<nimi>\r\n	+NAME:<nimi>\r\nOK\r\n
AT+PSWD=<salasana>\r\n	+PSWD:<salasana>\r\nOK\r\n

Taulukko 1. Bluetooth-moduulin AT-komennot.

Komentojen lopussa ”\r\n” merkitsee rivinvaihtoa, joka on pakollinen jokaisen komennon jälkeen. Moduuli voi myös vastata erilaisilla virhekoodeilla, jos esimerkiksi salasana on liian pitkä. Mikro-ohjainta ei ole kuitenkaan ohjelmoitu vastaanottamaan mitään vastauksia, koska salasanan ja nimen oikeellisuus tarkistetaan jo mobiililaitteessa. Nimi voi olla maksimissaan 31 merkkiä pitkä ja sisältää vain ASCII-merkistön merkkejä. Salasana voi olla maksimissaan 16 merkkiä pitkä ja sisältää vain ASCII-merkkejä.

5 OHJAUSPROTOKOLLA

Valoyksikköä ohjataan tässä luvussa määritellyillä ohjauskomennoilla. Nämä komennot ovat käyttäjältä kokonaan piilossa, toisin kuin valo-ohjelmien komennot. Komennot on suunniteltu olemaan mahdollisimman lyhyitä. Varsinainen komento on yhden tavun pituinen. Kaikissa komennoissa kaksi eniten merkitsevää bittiä on ykkösiä. Nämä kaksi bittiä toimivat yksinkertaisena tunnisteena komennon alulle. Taulukossa 2 on esitettynä kaikki komennot, joilla ohjataan valaisinyksikköä. Taulukossa on listattuna komentojen englanninkieliset nimet, komentoja vastaavat arvot heksadesimaalimuodossa, sekä mikro-ohjaimen lähettämä vastaus komentoon.

Komento	HEX	Arg 1	Arg 2	Arg 3	Vastaus
Set red brightness	C1	Target value	-	-	-
Set green brightness	C2	Target value	-	-	-
Set blue brightness	C3	Target value	-	-	-
Set RGB	C4	Red value	Green value	Blue value	-
Upload program	C6	Program lenght in bytes		-	-
Reboot	C7	-	-	-	-
Get RGB	C9	-	-	-	Color values
Get color correction	CC	-	-	-	CC state
Toggle color correction	CD	-	-	-	CC state
Bluetooth control	CE	BT command	Lenght	-	-
Switch off	CF	-	-	-	-

Taulukko 2. Ohjauskomennot.

1. **Set red / green / blue brightness**

   Nämä komennot asettavat tietyn värikomponentin kirkkauden argumentin 1 mukaiseen kirkkauteen. Argumentti 1 on luku alueelta [0..255]. Valoyksikkö ei lähetä mitään vastausta komennon suorittamisen jälkeen.

2. **Set RGB**

   Tämä komento asettaa kaikkien värikomponenttien kirkkaudet kolmessa argumentissa määriteltyihin kirkkauksiin. Argumentti 1 määrittelee punaisen, argumentti 2 vihreän ja argumentti 3 sinisen värikomponentin kirkkauden. Argumentit ovat lukuja alueelta [0..255].

   Tämän komennon tarkoitus on vähentää siirrettävän datan määrää verrattuna kolmeen peräkkäiseen ”set brightness” -komentoon.

3. **Upload program**

   Tämä komento aloittaa valo-ohjelman lähetyksen mobiililaitteesta valoyksikölle. Argumentit 1 ja 2 on yhdistetty niin, että ne muodostavat yhden 16-bittisen luvun, joka on alueelta [0.. 65535]. Tämä luku kertoo lähetettävän valo-ohjelman pituuden tavuina.

4. **Reboot**

   Tämä komento uudelleenkäynnistää mikro-ohjaimen. Tämän komennon avulla mikro-ohjain voidaan käynnistää ohjelmiston päivitys -tilaan. Mikro-ohjain käynnistyy uudelleen noin 500 millisekunnin kuluttua komennon lähettämisestä.

5. **Get RGB**

   Tämä komento kysyy valoyksiköltä kaikkien värikomponenttien tämän hetkistä kirkkautta. Valoyksikkö lähettää vastauksena kolme tavua, jotka sisältävät kirkkausarvot. Arvot lähetetään tässä järjestyksessä: punainen, vihreä, sininen. Arvot ovat lukuja alueelta [0..255].

6. **Get color correction**

   Tämä komento kysyy valoyksiköltä, onko värinkorjaus päällä. Valoyksikkö vastaa yhdellä tavulla, joka on luku 0 tai 1. Vastaus on 0, jos värinkorjaus ei ole päällä, ja 1 jos värinkorjaus on päällä.

7. **Toggle color correction**

   Tämä komento vaihtaa värinkorjauksen tilaa, eli asettaa sen päälle jos se ei ole päällä, ja asettaa sen pois päältä jos se on päällä. Valoyksikkö vastaa yhdellä tavulla, joka on luku 0 tai 1. Vastaus on 0, jos värinkorjaus laitetaan pois päältä, ja 1 jos värinkorjaus laitetaan päälle.

8. Bluetooth control

Tämä komento suorittaa Bluetooth-moduulia koskevia komentoja. Argumentti 1 määrittelee suoritettavan Bluetooth-moduulia koskevan komennon ja argumentti 2 määrittelee Bluetooth-komentoa koskevan datan pituuden tavuina.

Argumentissa 1 käytettäviä Bluetooth-komentoja on kaksi: SET_PIN (0x01) ja SET_NAME (0x02). Suluissa on komentoja vastaavat heksadesimaaliarvot. SET_PIN asettaa Bluetooth-moduuliin uuden salasanan. SET_NAME asettaa Bluetooth-moduulille uuden nimen. Nimen tai salasanan pituus tavuina määritellään argumentissa 2.

9. Switch off

Tämä komento asettaa kaikkien värikomponenttien kirkkauden nollaan ja pysäyttää mahdollisesti suorituksessa olevan valo-ohjelman suorituksen. Tämä komento ei kuitenkaan katkaise valoyksiköstä virtaa, tai muutenkaan aseta sitä virransäästötilaan. Valoyksikkö toimii tämän komennon jälkeen normaalisti välittömästi, kun uusi valo-ohjelma lähetetään sille, tai jollekin värikomponentille asetetaan uusi kirkkausarvo.

6 VALO-OHJELMAT

Valo-ohjelmat ovat yksinkertaisia ohjelmia, joiden avulla valaisin voidaan ohjelmoida muuttamaan väriä automaattisesti ilman, että mikro-ohjaimen flash-muistia pitää ohjelmoida uudelleen. Tämä mahdollistaa uusien valo-ohjelmien luomisen helposti ja nopeasti mobiililaitteen avulla.

6.1 Valo-ohjelmien rakenne

Valo-ohjelmat ovat tallennettuna mobiililaitteeseen tekstitiedostoina, joiden tiedostotunniste on ”.lp”. Tiedosto on jaettu kolmelle riville, joista jokaisella on lueteltu yhden värikomponentin ohjauskomennot. Ensimmäisellä rivillä on punaisen, toisella vihreän ja kolmannella sinisen värikomponentin komennot. Valo-ohjelmissa yhden värikomponentin komentojonosta käytetään nimitystä ”sequence”.

Mikro-ohjaimessa on varattuna valo-ohjelmille taulukko, johon mahtuu yhteensä 600 komentoa. 600 komentoa voidaan jakaa vapaasti kolmen värikomponentin osioiden välillä, eli yhden värikomponentin ohjelman kokoa ei ole erikseen rajoitettu.

6.2 Ohjelman suoritus

Mikro-ohjaimessa on käytössä ajastettu keskeytys, jonka lauetessa mikro-ohjain suorittaa jokaisen värikomponentin ohjelmasta yhden käskyn, tai jatkaa edellisen käskyn suorittamista jos kyseessä on esimerkiksi ”wait”-käsky. Keskeytykset on ohjelmoitu laukeamaan noin 25ms välein. Keskeytyksistä muodostuu ikään kuin kellolähde, joka tahdistaa valo-ohjelmien suoritusta. Tämän kellon tarkka taajuus määräytyy seuraavalla yhtälöllä.

$$f_{valo-ohjelma} = \frac{f_{clk}}{n \times ref}$$

Jossa f_{clk} = mikro-ohjaimen kellotaajuus, n = laskurin esijakaja, ref = vertailuarvo, eli OCR2A-rekisterin arvo.

Mikro-ohjaimessa käytetään 8 MHz:n kellotaajuutta. Tästä syystä esijakajaksi valitsin 1024, koska sen avulla keskeytystaajuuden saa suunnilleen oikeaan kertaluokkaan. Vertailuarvoksi valitsin 195, eli tarkka taajuus on edellisen kaavan mukaisesti noin 40,0641 MHz.

$$f_{valo-ohjelma} = \frac{8\,000\,000\ Hz}{1024 \times 195} \sim 40{,}0641\ MHz$$

Todellinen jaksonaika jää siis hieman alle 25 millisekunnin, kuten seuraavasta yhtälöstä käy ilmi.

$$T_{valo-ohjelma} = \frac{1}{f} = \frac{1}{40{,}0641\ \ldots\ MHz} \sim 0{,}02496\ s$$

Valo-ohjelmissa tästä jaksosta käytetään nimitystä ”tick”.

6.3 Valo-ohjelman komennot

Seuraavassa listassa on kaikki komennot, joita käytetään valo-ohjelmissa. Taulukossa on komentojen englanninkieliset nimet, komentoja vastaavat arvot heksadesimaalimuodossa, sekä mahdollisten argumenttien selitys. Komennolla voi olla 0-2 argumenttia komennosta riippuen.

Komento	**HEX**	**Arg 1**	**Arg 2**
Red sequence start	1	-	-
Green sequence start	2	-	-
Blue sequence start	3	-	-
Fade	4	target value	size of step
Set brightness	5	target value	-
Wait color	6	colors to wait	-
Notify color	7	colors to notify	-
Wait	8	ticks to wait	

Taulukko 3. Valo-ohjelman komennot

1. **Red / Green / Blue sequence start**

   Nämä komennot kertovat mikro-ohjaimelle, mistä kohtaa tietyn värikomponentin jakso alkaa. Käyttäjän ei tarvitse välittää näistä komennoista, koska mobiilisovellus lisää ne automaattisesti jokaisen värikomponentin jakson alkuun. Näitä komentoja ei tallenneta tekstitiedostoon. Komennot vastaavat rivinvaihtoa tekstitiedostossa, eli ne erottavat värikomponenttien jaksot toisistaan.

2. **Fade**

   Tämä komento vaihtaa värikomponentin kirkkautta kohti argumentissa 1 määriteltyä kohdearvoa argumentissa 2 määritellyn kokoisilla askelilla. Mikro-ohjain lisää värikomponentin kirkkautta yhden askeleen verran jokaisella valo-ohjelman kellojaksolla, kunnes tavoitearvo on saavutettu. Tavoitearvo on luku alueelta [0..255] ja askeleen koko on luku alueelta [-128..127]. Askeleen koon pitää olla negatiivinen silloin, kun tavoitearvo on pienempi kuin lähtöarvo. Askeleen koon ei tarvitse olla jaollinen kokonaismuutoksen kanssa. Jos viimeinen askel menee tavoitearvon yli, kirkkaudeksi asetetaan tavoitearvo.

   Komentoon kuluva aika voidaan laskea seuraavalla yhtälöllä.

$$T = ceil\left(\frac{(t_2 - t_1)}{s_{size}}\right) \times \frac{1}{f_{valo-ohjelma}}$$

   Jossa T = komennon viemä aika, ceil = pyöristys ylöspäin seuraavaan tasalukuun, t_1 = kirkkaus ennen komennon suoritusta, t_2 = tavoitekirkkaus, S_{size} = askeleen koko, $f_{valo\text{-}ohjelma}$ = valo-ohjelman komentojen suoritustaajuus.

   Komennon suorittamiseen kuluu kuitenkin aina vähintään yksi suoritusjakso, vaikka lähtö ja tavoitearvot olisivat jo lähtötilanteessa samat. Jos askeleen koko on nolla, ohjelman suoritus jää suorittamaan tätä komentoa loputtomaksi ajaksi, koska tavoitearvoa ei koskaan saavuteta.

3. **Set brightness**

   Tämä komento asettaa värikomponentin kirkkauden argumentissa 1 määriteltyyn kirkkausarvoon. Kirkkausarvo on kokonaisluku alueelta [0..255]. Komennon suorittamiseen kuluu yhden suoritusjakson aika.

4. **Wait color**

   Tämä komento pysäyttää värikomponentin ohjelman suorituksen, kunnes se on saanut ”notify”-käskyn kaikilta argumentissa 1 määriteltyjen värikomponenttien ohjelmilta. Odotettavat värit määritellään argumentissa 1 asettamalla värejä vastaavat bitit ykkösiksi. Väreille varatut bitit ovat seuraavat:
   b_0 = punainen, b_1 = vihreä ja b_2 = sininen, jossa b_0 = vähiten merkitsevä bitti.

   Jos värikomponentin ohjelma jää odottamaan itseään, suoritusta ei jatketa enää koskaan, koska se ei voi lähettää itselleen ”notify”-käskyä odotustilassa. Tällainen tapaus on estetty luotaessa ohjelmia mobiilisovelluksella, mutta on käytännössä mahdollista tekstitiedostoa muokkaamalla.

   Esimerkki argumentista binäärimuodossa, joka asettaa ohjelman odottamaan punaista ja sinistä: 0000 0101.

5. **Notify color**

   Tämä komento vapauttaa argumentissa 1 määriteltyjen värikomponenttien ohjelmat kutsujaa koskevasta wait-tilasta. Vapautettavat värikomponentit määritellään samalla tavalla, kuin ”wait”-komennon odotettavat värit.

6. Wait

Tämä komento asettaa värikomponentin ohjelman odotustilaan argumentissa määritellyn pituiseksi ajaksi. Argumentit 1 ja 2 on yhdistetty niin, että ne muodostavat yhden 16-bittisen luvun. Argumentti 2 sisältää 16-bittisen luvun vähiten merkitsevät bitit ja argumentti 1 eniten merkitsevät. Luku määrittelee odotettavien suoritusjaksojen määrän. Odotettava aika voidaan laskea seuraavalla yhtälöllä. Argumentti on luku alueelta [0.. 65535].

$$t = n \times \frac{1}{f_{valo-ohjelma}}$$

Jossa t = odotusaika, n = odotettavien suoritusjaksojen määrä, $f_{valo\text{-}ohjelma}$ = valo-ohjelman komentojen suoritustaajuus.

Komennon suorittamiseen kuluu vähintään yksi suoritusjakso, eli odotusajan asettaminen nollaan vastaa samaa tilannetta, kun odotusajan asettaminen yhdeksi.

7 KAUKO-OHJAINSOVELLUS

Kauko-ohjainsovellus on suunniteltu Android-käyttöjärjestelmille, joiden versio on 3.0 tai uudempi. Android-sovelluksia kehitettäessä pitää aina alkuvaiheessa päättää, kuinka vanhoja Android-versioita aiotaan tukea. Mitä vanhempia versioita halutaan tukea, sitä vähemmän on käytössä uusien versioiden tuomia parannuksia ja helpotuksia ohjelman toteutukseen.

Päädyin tukemaan versiota 3.0 ja sitä uudempia, koska jo nyt yli puolet käytössä olevista Android-laitteista on varustettu tuolla versiolla tai sitä uudemmalla (Android Developers 2013). Lisäksi kauko-ohjainsovellus sisältää esimerkiksi Fragment-näkymiä, joita ei vanhemmissa versioissa vielä ole (Android API Guides 2013). Suositus on, että sovellukset tukisivat ainakin 90 % käytetyistä Android-laitteista (Android Developers 2013). Tämän sovelluksen osalta tuohon kattavuuteen ei valitettavasti vielä päästä.

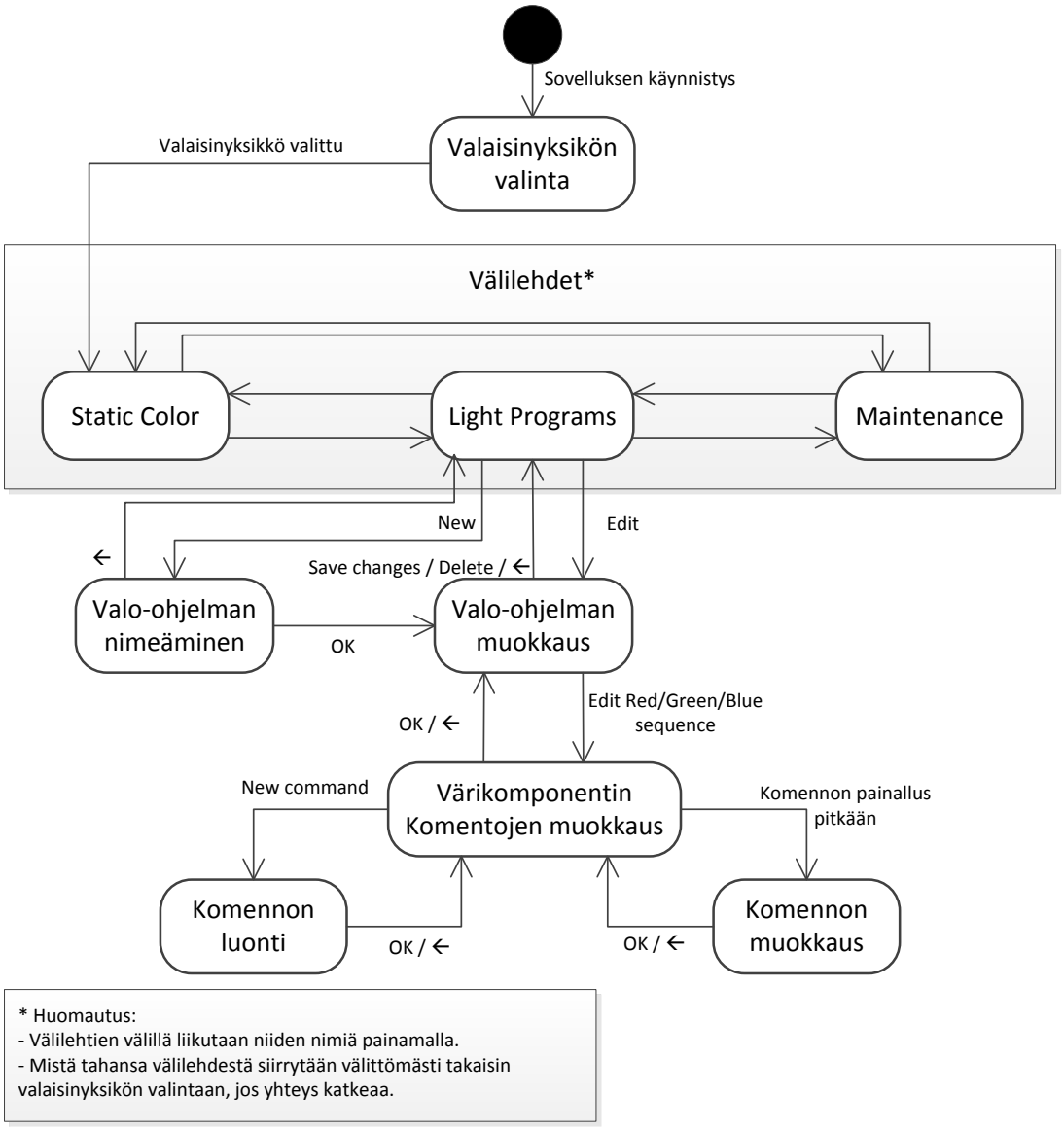
7.1 Ohjelman rakenne

Ohjelma on jaettu kolmeen pakettiin. Paketit ovat ”com.ile.moodlight”, ”com.ile.moodlight.activity” ja ”com.ile.moodlight.lightcommand”. Luokat on jaettu paketteihin seuraavasti:

- **com.ile.moodlight.activity:** Sisältää kaikki aktiviteettiluokat (Liite 10).
- **com.ile.moodlight.lightprogram:** Sisältää kaikki valo-ohjelmia koskevat luokat (Liite11).
- **com.ilemoodlight:** Sisältää kaikki muut luokat (Liite 9).

Ohjelman ylimpänä tasona toimii TopContainerActivity-niminen aktiviteetti, joka toimii ikään kuin koko ohjelman yhteen kasaajana. TopContainerActivity ei sisällä mitään näkyviä käyttöliittymäkomponentteja. TopContainerActivity sisältää FrameLayout-näkymän, joka jakaa näkymän kolmeen välilehteen.

Kaikkien aktiviteettien ja välilehtien ulkoasu on kuvattu erillisissä XML-tiedostoissa (Liite 12). Kaikki käyttöliittymässä näkyvät tekstit ovat määritelty strings.xml-tiedostossa.

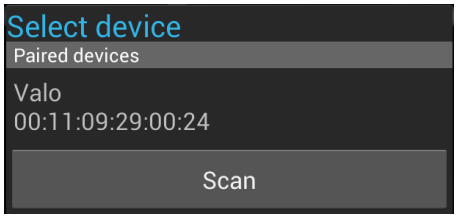


Kuva 9. Navigointi ohjelmassa.

Kuva 9 esittää ohjelman rakennetta. Kuvassa ”←”-merkki tarkoittaa Android-laitteiden ”ylös”-painiketta. Painikkeella siirrytään aina edelliseen ikkunaan ilman nykyisen ikkunan tietojen tallennusta.

Heti ohjelman käynnistyessä TopContainerActivity käynnistää SelectDeviceActivity-aktiviteetin, joka listaa Bluetooth-laitteet, ja pyytää käyttäjää valitsemaan valaisinyksikön, johon yhteys muodostetaan (Kuva 10). Jos käyttäjä ei valitse mitään va-

laisinyksikköä, käyttäjälle näytetään virheilmoitus ja ohjelma suljetaan, koska sitä ei voi käyttää ilman yhdistettyä valaisinyksikköä. Jos yhteyden muodostus ei onnistu, sama valinta-aktiviteetti aukeaa ja pyytää käyttäjää valitsemaan yhdistettävän valaisinyksikön uudelleen.



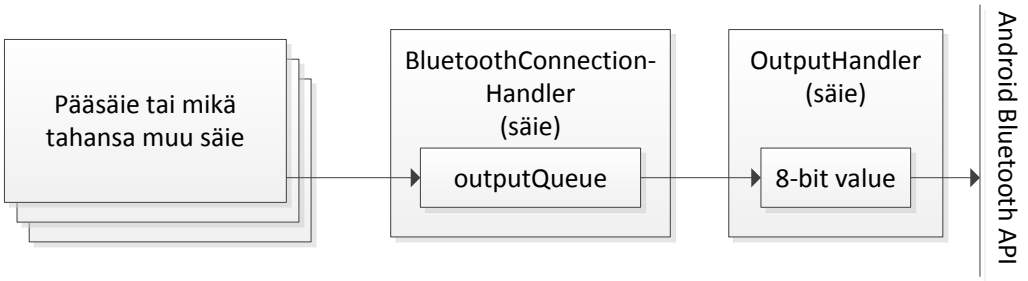
Kuva 10. Käytettävän valaisinyksikön valinta.

7.2 Bluetooth-yhteyden hallinta

Bluetooth-yhteyttä hallitsee BluetoothConnectionHandler-luokan instanssi. Luokka laajentaa Javan Thread-luokkaa, eli sen instanssi on tarkoitus ajaa omana säikeenä. Erillinen säie Bluetooth-yhteyden hallinnalle on pakollinen, koska ohjelman pääsäikeessä ei voi suorittaa mitään pitkäkestoista toimintoa (Android API Guides), ja yhteyden hallintaa pitää suorittaa koko ohjelman ajon ajan.

Luokka sisältää kaksi aliluokkaa: OutputHandler ja InputHandler. Näiden luokkien instanssit toimivat myös omina säikeinään. Nimiensä mukaisesti ne hallitsevat lähtevää ja tulevaa Bluetooth-liikennettä. Molemmille toiminnoille tarvitaan oma säie, koska toimintoja ajetaan samanaikaisesti.

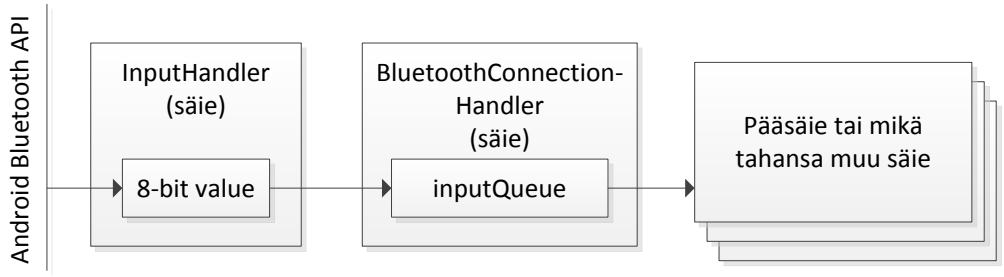
OutputHandler vahtii BluetoothConnectionHandler-luokan lähtevän datan puskuria. Heti kun jokin muu säie kirjoittaa puskuriin dataa, OutputHandler lähettää datan valaisinyksikölle. OutputHandler ei ole millään tapaa tietoinen siitä, minkä tyyppistä dataa se lähettää valaisinyksikölle. Data voi siis olla esimerkiksi ohjauskomentoja, valo-ohjelmia, jne.



Kuva 11. Datan lähetys valaisinyksikölle.

Kuva 11 esittää miten mikä tahansa säie voi lähettää dataa valaisinyksikölle. OutputQueue-olio on tyypiltään LinkedBlockingQueue. Alun perin käytin jonona oliota, jonka tyyppi oli ArrayBlockingQueue, mutta sen tyypin kanssa jono tuntui hidastavan lähetystä huomattavasti. Javan dokumentaatiosta kävikin ilmi, että LinkedBlockingQueue päästää lävitseen nopeammin dataa, kuin Array-pohjaiset jonot (Oracle 2011).

Javan BlockingQueue-interfacen toteuttavat jonot ovat kätevä tapa toteuttaa FIFO-jono kahden tai useamman säikeen välille. Yksi säie voi odottaa jonoon tulevaa dataa, ja toiset säikeet voivat kirjoittaa dataa jonoon.



Kuva 12. Datan vastaanotto valaisinyksiköltä.

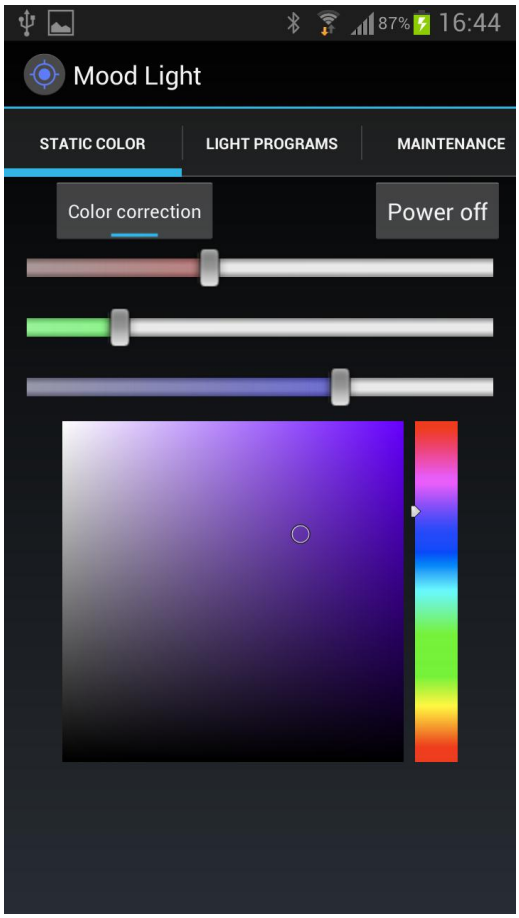
Kuva 12 esittää miten datan vastaanotto toimii. InputHandler toimii vastaavasti kuin OutputHandler, mutta se kuuntelee jatkuvasti tulevan datan virtaa ja kirjoittaa vastaanotetun datan tulevan datan puskuriin, josta mikä tahansa ohjelman muu säie voi lukea datan. Mekanismin heikkoutena on se, että dataa vastaanottavan säikeen pitää jäädä kuuntelemaan inputQueue-jonoa niin pitkäksi aikaa, kunnes data on vastaanotettu. Tämä ei ole normaalisti ongelma, jos yhteys toimii oikein, mutta yhteyshäiriöiden yhteydessä vastaanottava säie saattaa pysähtyä pitkäksi aikaa. Turha odotus

varsinkin pääsäikeessä ei ole koskaan toivottua, koska odotuksen aikana esimerkiksi käyttöliittymä jumiutuu.

Jos yhteys katkeaa jostain syystä, BluetoothConnectionHandler ilmoittaa asiasta TopContainerActivitylle Handler-mekanismin avulla. Tämän jälkeen TopContainerActivity pyytää käyttäjää valitsemaan uudelleen valaisinyksikön, johon yhteys muodostetaan (Kuva 10). BluetoothConnectionHandler ei voi suoraan avata tätä ikkunaa, koska ainoastaan pääsäie voi vaikuttaa ohjelman käyttäjälle näkyvään osaan (Android API Guides).

7.3 Värin asettaminen

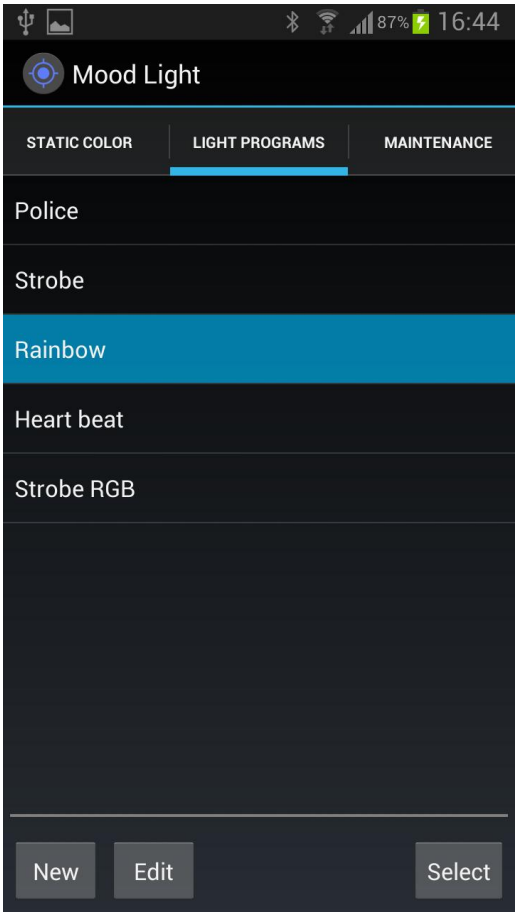
Värin asettaminen tapahtuu TopContainerActivityn vasemmanpuoleisesta välilehdestä. Tämän välilehden toteuttaa luokka StaticColorFragment ja sitä vastaava ulkoasumäärittely fragment_static_color.xml.



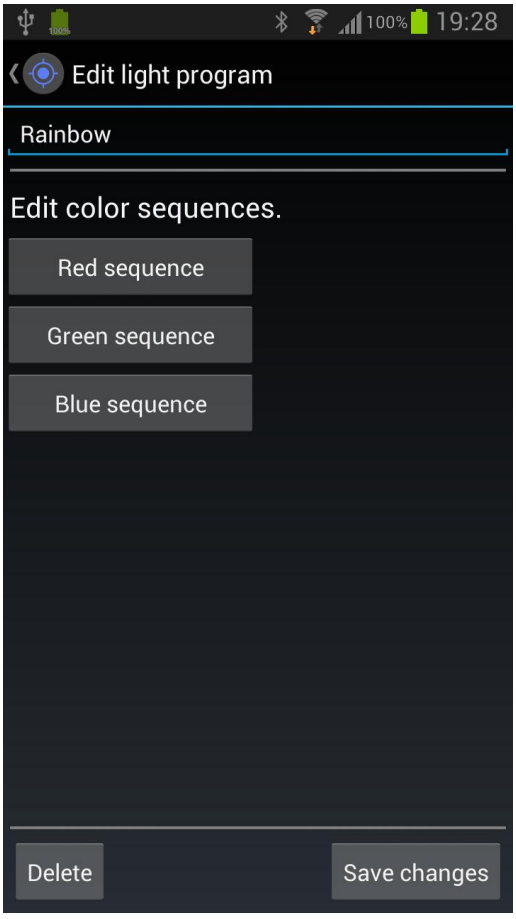
Kuva 13. Värin asettaminen.

Välilehdeltä voidaan asettaa väri joko kolmen liukupalkin avulla, tai palkkien alapuolella olevasta värivalitsimesta (Kuva 13). Liukupalkkien avulla jokaisen värikomponentin kirkkautta saa säädettyä erikseen, kun taas värivalitsimesta saa helpommin valittua minkä tahansa värin. Liukupalkit ja värivalitsin on synkronoitu keskenään niin, että ne vastaavat aina samaa väriä.

Värivalitsin pohjautuu avoimen lähdekoodin Android Color Picker -kirjastoon, jonka on tehnyt nimimerkki ”yukuku” (Yukuku 2013).
Välilehdeltä voi myös sammuttaa valon kokonaan ”Power off”-painikkeella ja ottaa värinkorjauksen käyttöön tai poistaa sen käytöstä ”Color correction”-painikkeella.



Kuva 14. Valo-ohjelmien hallinta.

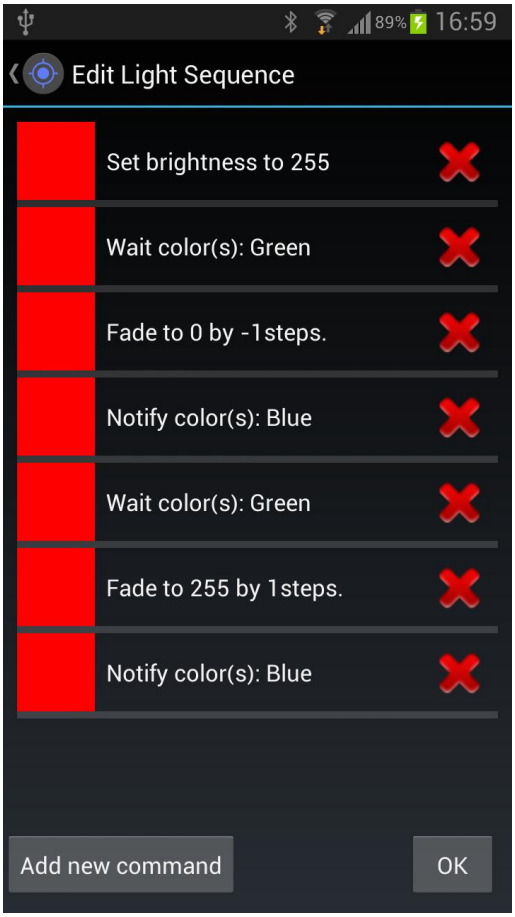


Kuva 15. Valo-ohjelman muokkauksen päävalikko.

7.4 Valo-ohjelmien hallinta

Valo-ohjelmien hallinta tapahtuu TopContainerActivity:n keskimmäisen välilehden kautta (Kuva 14). Tämän välilehden toteuttaa LightProgramFragment-luokka ja sitä vastaava ulkoasumäärittely fragment_light_program.xml.
Välilehdellä näkyy listattuna kaikki saatavilla olevat valo-ohjelmat. Ohjelman voi lähettää valaisinyksikölle ensin valitsemalla sen listalta ja painamalla ”Select”-nappia. Ohjelmaa voi muokata ensin valitsemalla sen listalta ja painamalla ”Edit”-nappia. ”Edit”-napin painallus avaa EditLightProgramActivity-aktiviteetin (Kuva 15).

EditLightProgramActivity:n ulkoasun toteuttaa ulkoasumäärittely activity_edit_light_program.xml. Tämän aktiviteetin kautta voidaan muokata valo-ohjelman toiminnallisutta, vaihtaa valo-ohjelman nimeä tai poistaa valo-ohjelma. Uudelleennimeäminen tapahtuu kirjoittamalla uusi nimi yläreunassa olevaan nimi-kenttään vanhan nimen tilalle. Valo-ohjelman poisto tapahtuu painamalla ”Delete”-nappia. Valo-ohjelman värikomponenttien ohjelmaosia voi muokata painamalla ”Red / Green / Blue sequence” –nappia. Kaikkien tehtyjen muutosten jälkeen pitää painaa ”Save changes”-nappia, jotta muutokset astuvat voimaan. Näistä napeista aukeaa EditLightSequenceActivity-aktiviteetti, josta varsinainen ohjelman editointi tapahtuu (Kuva 16).



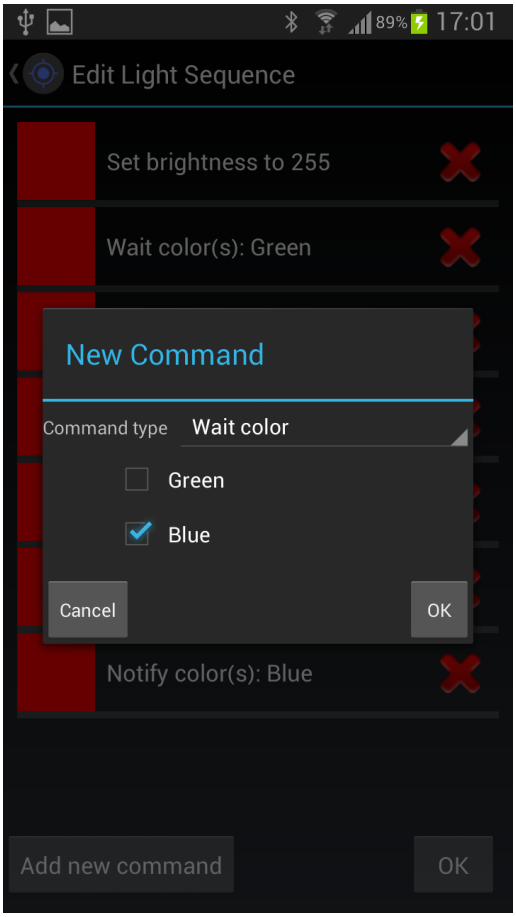
Kuva 16. Punaisen valo-ohjelman muokkaus

EditLightSequenceActivity:n ulkoasun toteuttaa ulkoasumäärittely activity_edit_light_sequence.xml. Kuva 16 on esimerkki punaisen värikomponentin ohjelmasta. Komentojen nimien vasemmalla puolella olevan laatikon väri kertoo parhaillaan muokattavan värikomponentin.

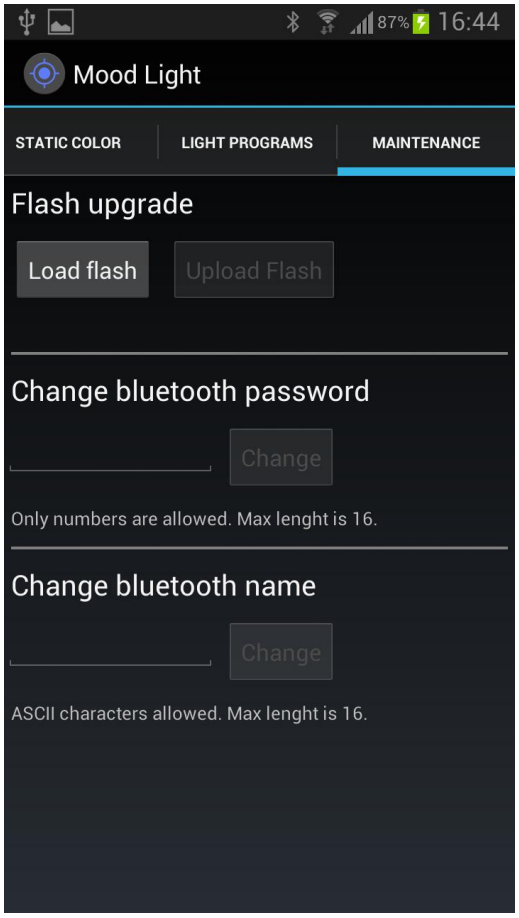
Komentojen järjestystä voi muuttaa raahaamalla komento haluttuun paikkaan. Raahaaminen tapahtuu vetämällä komentoa edellä mainitusta laatikosta. Komentoa voi muokata painamalla pitkään (noin yksi sekunti) komennon nimen kohdalta. Komennon voi poistaa punaisesta rastista. Uusia komentoja voi luoda ”Add new command”-nappia painamalla. Napin painallus avaa uuden AddNewCommand-aktiviteetin (Kuva 18). Tehdyt muutokset hyväksytään painamalla OK-nappia. Jos muutoksia ei halua ottaa käyttöön, edelliseen aktiviteettiin pääsee mobiililaitteen takaisin-napilla.

NewCommandActivityn ulkoasun toteuttaa ulkoasumäärittely activity_new_command.xml. Aktiviteetista valitaan komennon tyyppi ja asetetaan sitä koskevat argumentit. Tätä samaa aktiviteettia käytetään komentojen muokkaamiseen.

EditLightSequenceActivity:ssä käytetty listanäkymä käyttää hyväkseen avoimen lähdekoodin DragSortListView-kirjastoa. Kirjasto mahdollistaa listan, jonka elementtien järjestystä voidaan helposti muuttaa (Bauer 2013).



Kuva 18. Uuden komennon luonti.



Kuva 17. Valaisinyksikön hallinta.

7.5 Valaisinyksikön hallinta

Valaisinyksikön hallinta tapahtuu TopContainerActivity:n kolmannelta välilehdeltä. Tämän välilehden toteuttaa MaintenanceFragment-luokka ja sitä vastaava ulkoasumäärittely fragment_maintenance.xml (Kuva 17).

Tämän välilehden kautta voidaan päivittää valaisinyksikön mikro-ohjaimen ohjelmisto, sekä vaihtaa valaisinyksikön Bluetooth-nimi ja -salasana.

Mikro-ohjaimen uusi ohjelmisto pitää sijaita mobiililaitteen massamuistilla ”MoodLight”-kansiossa. Painamalla ”Load flash”-painiketta sovellus tarkistaa ensin tiedoston oikeellisuuden. Jos tiedosto on tyypiltään oikea ja siinä ei ole havaittavia virheitä, varsinainen ohjelman päivitys voidaan aloittaa.

Päivitys tapahtuu painamalla valaisinyksikön päivitysnappia pohjassa ja samanaikaisesti painamalla sovelluksen ”Upload flash”-nappia. Kun päivitys on alkanut, valaisinyksikön päivitysnappia ei tarvitse enää painaa. Päivitysoperaation tila ilmestyy ”Load flash” ja ”Upload flash” -nappien alapuolelle. Kun päivitys on valmis, valaisinyksikkö uudelleenkäynnistyy automaattisesti.

Päivitystoiminnossa on käytetty hyväksi Atmelin kehittämää avoimen lähdekoodin AVR OSP -päivitysohjelman rakennetta (Atmel AVR911 2013). Alkuperäinen ohjelma on tehty C++ -kielellä, eli minun piti kirjoittaa tarvitsemani toiminnot uudelleen Java-kielelle. Koodia ei ole siis suoraan kopioitu, mutta toiminnallisuus on lähes identtinen alkuperäisen koodin kanssa. Ohjelmiston päivitykseen liittyy luokat FlashUpdater ja HEXFile.

FlashUpdater laajentaa Androidin AsyncTask-luokkaa, joka on eräs tapa toteuttaa pääsäikeen ja toisen säikeen välinen kommunikaatio. AsyncTask-luokkaan voidaan kirjoittaa onPreExecute-metodi, joka suoritetaan pääsäikeessä ennen toisen säikeen suorittamista. Tämä metodi voi sisältää koodia, joka on yhteydessä käyttöliittymäkomponentteihin. doInBackground-metodi suoritetaan erillisessä säikeessä, eli se ei voi sisältää käyttöliittymää muokkaavaa koodia. Metodi voi kuitenkin sisältää publishProgress-kutsun, jonka seurauksena pääsäikeessä suoritetaan tämän luokan on-

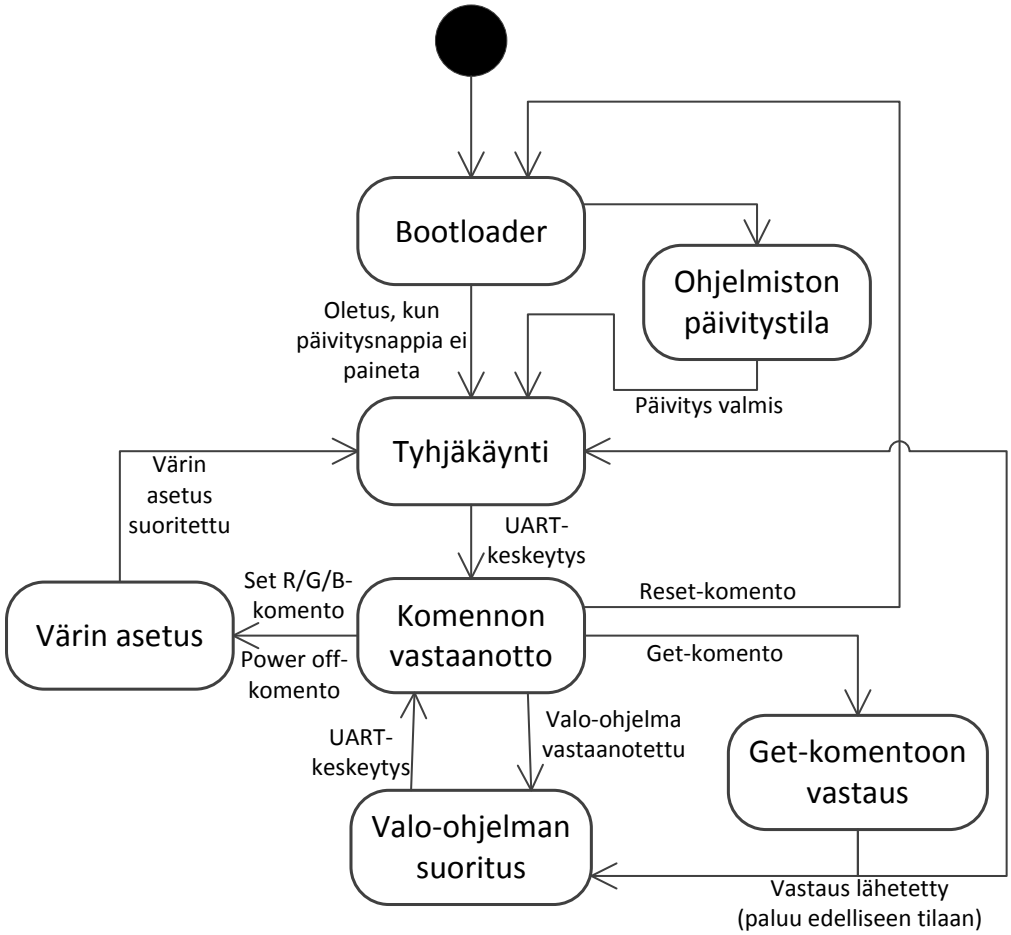
ProgressUpdate-metodi, joka voi olla yhteydessä käyttöliittymäkomponentteihin. Kun säie on suoritettu loppuun, pääsäikeessä suoritetaan vielä tämän luokan onPostExecute-metodi, jonka kautta luokka voi taas olla yhteydessä käyttöliittymäkomponentteihin.

Toteutin ohjelmiston päivitystoiminnon AsyncTask-luokan pohjalta, koska sen avulla on helppo päivittää käyttöliittymäkomponentteja säiettä ajettaessa. Tässä tapauksessa komponenttien päivittämistä käytetään ohjelmistopäivityksen tilan kertomiseen. Näin käyttäjä saa tietää reaaliajassa, mitä parhaillaan tapahtuu. Pitkäkestoisen toiminnon aikana käyttäjä saattaa helposti luulla toiminnon jumiutuneen, jos hänelle ei anneta mitään merkkiä edistymisestä.

Toisena vaihtoehtona olisi ollut toteuttaa päivitys Javan Thread-luokkaa laajentamalla ja käyttämällä Androidin Handler-mekanismia käyttöliittymän päivittämiseen, kuten tässäkin ohjelmassa on toisissa paikoissa käytetty. Handler-mekanismilla toteutus olisi ollut kuitenkin jonkin verran monimutkaisempi.

8 MIKRO-OHJAIMEN SOVELLUS

Mikro-ohjaimen sovellus koostuu kahdesta osasta: käynnistyslataajasta ja varsinaisesta valoa ohjaavasta pääohjelmasta. Kehitin näitä osia toisistaan riippumattomina projekteina AVR-studiossa. Kuva 19 esittää koko mikro-ohjaimen ohjelmiston tilamuutokset.



Kuva 19. Mikro-ohjaimen sovelluksen tilakaavio.

8.1 Pääohjelma

Pääohjelma koostuu kolmesta tiedostosta: RGBLight.h (Liite 6), RGBLight.cpp (Liite 7) ja ColorComponent.h (Liite 8). RGBLight.h sisältää funktioiden ja globaalien muuttujien esittelyt, sekä erilaiset määritykset, kuten ohjauskomentojen arvot ja argumenttien määrät. RGBLight.cpp sisältää pääohjelman ja siihen liittyvät funktiot, kuten UART-lähetys- ja -vastaanottofunktiot, sekä mikro-ohjaimen käynnistyessä

tehtävät alustustoiminnot. ColorComponent.h sisältää ColorComponent-luokan, jota käytetään värikomponenttien valo-ohjelmaosien suorittamisessa. Luokasta on olemassa aina kolme instanssia; yksi instanssi jokaiselle värikomponentille.

8.2 Käynnistyslataaja

Käynnistyslataaja suoritetaan aina kun valaisinyksikköön kytketän virta ja Reboot-komennon jälkeen. Käynnistyslataaja siirtyy ohjelmistonpäivitystilaan, jos käyttäjä painaa käynnistyksen aikana päivitysnappia. Muussa tapauksessa käynnistyslataaja ei tee mitään ja mikro-ohjain siirtyy suorittamaan pääohjelmaa.

Käynnistyslataaja on lähes suora kopio Atmelin tekemästä avoimen lähdekoodin käynnistyslataajasta (Atmel AVR109 2004)(Liite 15).

Käynnistyslataaja ei voi päivittää itseään, eli se on ainut osa valaisinyksikön ohjelmasta, jonka päivitys ei onnistu ilman ohjelmointiliitäntää.

8.3 Lähetys ja vastaanotto sarjaportilla

Vastaanotto sarjaportilla tapahtuu keskeytysohjatusti, eli aina kun mikro-ohjain on vastaanottanut yhden tavun, se laukaisee keskeytyksen, jonka jälkeen mikro-ohjain tulkitsee saamansa datan sisällön ja toimii vaaditulla tavalla. Toiminto pitää olla keskeytysohjattu, koska mikro-ohjain ei voi ennalta tietää, milloin sille lähetetään dataa.

Mikro-ohjain ei lähetä paljoakaan dataa mobiililaitteelle, joten päätin toteuttaa lähetyksen yksinkertaisesti ilman keskeytyksiä. Toisin sanoen ohjelman suoritus pysähtyy siksi aikaa, kun mikro-ohjain lähettää dataa.

8.4 Ohjauskomennot

Mikro-ohjaimen vastaanottaessa ohjauskomennon, se tarkistaa mistä komennosta on kyse, ja jää odottamaan sille komennolle määriteltyä määrää argumentteja. Kun komento ja mahdolliset argumentit on vastaanotettu, mikro-ohjain suorittaa ohjauskomennon. Ohjauskomennoista on kerrottu tarkemmin kappaleessa 5 - Ohjausprotokolla.

8.5 Valo-ohjelmien vastaanotto

Valo-ohjelmaa vastaanottaessa mikro-ohjain tulkitsee valo-ohjelman komennot lennossa heti kun komento on vastaanotettu kokonaisuudessaan. Tämän jälkeen valo-ohjelman komento tallennetaan komentotauluun. Kun kaikki komennot on vastaanotettu ja tallennettu komentotauluun, mikro-ohjain alkaa automaattisesti suorittaa kyseistä ohjelmaa.

8.6 Bluetooth-moduulin hallinta

Bluetooth-moduulia koskevan komennon vastaanottaessa mikro-ohjain muodostaa komennosta AT-komennon, jolla ohjataan Bluetooth-moduulia. Käytettävät AT-komennot on listattu ”Bluetooth-moduuli”-kappaleessa. Ennen komennon lähettämistä mikro-ohjain vetää Bluetooth-moduulin CMD-pinnin ylös, jotta Bluetooth-moduulin hallinta onnistuu.

9 JÄRJESTELMÄN TESTAUS

9.1 Kauko-ohjainsovelluksen testaus

Kauko-ohjainsovelluksen testaus tapahtui pääasiallisesti niin, että yhden toiminnon implementoituani varmistin sen toimivuuden ennen seuraavan toiminnon implementointiin siirtymistä.

Mobiililaitteen ja mikro-ohjaimen välistä yhteyttä tarkkailin tietokoneen avulla. Tarkkailuun käytin USB-porttiin kytkettävää sarjaliikenne-sovitinta, jonka kytkin Bluetooth-moduulin rinnalle mikro-ohjaimen RXD- tai TXD-pinniin riippuen siitä, halusinko nähdä mikro-ohjaimen vastaanottaman vai lähettämän datan. Datan vastaanotto onnistui tietokoneella esimerkiksi Hyperterminal-ohjelmalla.

Mikro-ohjaimen sovelluksen päivitykseen käytettävän osan ohjelmasta toteutin ensin normaalina Java-ohjelmana, jota voi ajaa tietokoneella. Näin pystyin helpommin todentamaan sen toiminnallisuuden. Kun päivitys toimi tietokoneella, voisin kopioida ohjelman toiminnallisen osan suoraan kauko-ohjainsovelluksen käyttöön. Testauksen oheistuotteena syntyi siis myös yksinkertainen Java-ohjelma, jolla voi päivittää AVR OSP-yhteensopivan mikro-ohjaimen ohjelmiston sarjaportin kautta.

Ohjelman kehityksen aikana vastaan tuli paljon sellaisia tilanteita, joissa käyttäjä pystyi tekemään jotain sellaista, joka olisi kaatanut sovelluksen. Suurimmaksi tällaiseksi ongelmaksi muodostui Bluetooth-yhteyden tilan varmistaminen. Jos yhteyttä ei ole muodostettu ja käyttäjä yrittää esimerkiksi asettaa jonkin värin, sovelluksen lähtevän datan puskuri täyttyy äkkiä ohjauskomennoista, kun niitä ei voida lähettää. Puskurin täyttyessä sovellus kaatuu. Yksinkertaisena ratkaisuna kehitin mekanismin, joka estää ohjelman käytön, jos yhteyttä ei ole muodostettu. Edelleen on silti mahdollista, että puskuri täyttyy, jos yhteys katkeaa kesken ohjelman käytön ja Bluetooth-yhteyttä hallinnoiva BluetoothConnectionHandler ei ehdi huomata katkennutta yhteyttä ennen puskurin täyttymistä. Tilannetta helpottaa se, että suurensin puskurin kokoa niin paljon, että sitä on lähes mahdoton täyttää muutamassa sekunnissa, jonka aikana katkennut yhteys jo ehditään havaitsemaan.

Kauko-ohjainsovelluksen testausta ja virheiden etsimistä helpottaa huomattavasti Androidin logcat-lokiohjelma, jonka avulla koodiin voi sijoittaa lokiin kirjoituskäskyjä (Android Tools 2013). Nämä lokimerkinnät tulevat näkymiin reaaliajassa tietokoneelle Eclipsen logcat-ikkunaan.

Android-sovellukset toimivat teoriassa kaikissa laitteissa joissa on minimivaatimusten mukainen Android-versio tai sitä uudempi. Käytännössä näin ei välttämättä kuitenkaan aina ole. Suurimman ongelman sovellusten toimivuuteen aiheuttaa laitteiden erikokoiset näytöt. Tässä tapauksessa testaus on suoritettu vain yhdellä puhelimella, eikä yhdelläkään tablet-laitteella. Tältä osin testaus jäi ehdottomasti liian vähäiseksi.

9.2 Mikro-ohjaimen sovelluksen testaus

Mikro-ohjaimen sovelluksen testaus eteni pääpiirteissään hyvin samaan tapaan, kuin kauko-ohjainsovelluksenkin, eli yhden toiminnon implementoituani testasin sen toiminnan. Kehitysvaiheessa, kun en ollut vielä tehnyt kauko-ohjainsovellusta, testasin mikro-ohjaimen ohjelman toiminnallisuutta kytkemällä sen tietokoneeseen USB-porttiin kytkettyyn sarjaliikenne adapteriin. Näin pystyin lähettämään tietokoneelta haluamiani ohjauskomentoja suoraan mikro-ohjaimelle ja varmistamaan niiden oikean toiminnan.

Kun koodi ei toiminut halutulla tavalla, lisäsin koodiin ylimääräisiä uartPutChar()-fuktioita, joiden avulla pystyin seuraamaan ohjelman kulkua tai muuttujien arvoja.

10PARANNUSEHDOTUKSET

Työtä tehdessä mieleeni tuli paljon uusia ideoita, joita järjestelmään voisi implementoida ja osan alkuperäisistä ideoistakin jouduin jättämään pois, koska työn laajuus olisi muuten paisunut liikaa.

Kauko-ohjainsovelluksen komentojen lähettämiseen ja vastausten vastaanottoon voisi tehdä oman luokan. Nykyinen toimintamalli on melko alkeellinen, koska jos komennolta odotetaan vastausta, koodin suorituksen pitää jumiutua niin pitkäksi aikaa, että vastaus on saatu, tai odotus aikakatkaistaan. Muutenkin komentojen kutsuminen tapahtuu niin alhaisella abstraktiotasolla, että uusien komentojen lisääminen tai vanhojen muokkaaminen koodiin on melko työlästä ja saattaa helposti rikkoa ohjelman toiminnallisuuden.

Mikro-ohjaimen sovellukseen voisi lisätä toiminnon, joka mahdollistaisi valo-ohjelmien tallentamisen EEPROM-muistiin.

Valo-ohjelmien käyttöön varattu taulukko käyttää tällä hetkellä ylimääräistä muistia, koska jokaiselle komennolle on varattu kolme tavua muistia, vaikka suurin osa komennoista ei käytä kaikkia kolmea tavua. Muistin voisi siis käyttää järkevämminkin.

Valaisin olisi kätevä saada sammumaan automaattisesti tietyn ajan kuluttua, jos käyttäjä näin haluaa. Mahdollisesti myös ajastettu päälle kytkentä voisi olla hyvä lisäominaisuus.

Suunnittelin piirilevyn alun perin niin, että siitä voisi tehdä yksipuolisen. Tästä johtuen komponenttien sijoittelu ei ole paras mahdollinen. Komponentit voisi siis sijoitella paljon pienempään tilaan.

Mobiilisovelluksen käyttöliittymä vaatisi tarkempaa suunnittelua, jotta siitä tulisi käyttäjäystävällisempi. Mobiilisovelluksen graafinen ilme vaatisi muutenkin parantelua, kuten paremman ikonin. Nykyinen ikoni nimittäin pohjautuu yhteen Androidin vakiokuvakkeeseen.

11YHTEENVETO

Ennen työn aloittamista tiedostin jo sen, että työn laajuus kasvaa helposti todella suureksi, kun työtä tehdessä mieleen tulee uusia toimintoja, joita työhön voisi implementoida. Tästä syystä asetin itselleni tarkat rajoitukset, mitä toimintoja työhön teen. Tästäkin huolimatta aliarvioin työn laajuuden etukäteen ja jouduin jättämään toimintoja pois lopullisesta versiosta. Poisjääneitä toimintoja ovat valo-ohjelmien tallennus EEPROM:iin ja viivästetty valon sammutus.

Muuten olen tyytyväinen tekemääni työhön ja siihen, että työstä tuli juuri sellainen, kuin oli alun perin tarkoituskin tulla. Järjestelmästä tuli melko helposti muokattava ja laajennettava, jota itse pidin tärkeänä ominaisuutena alusta lähtien.

Järjestelmä saa varmasti jatkokehitystä myöhemmin. Tarkoituksenani on myös julkaista lähdekoodit ja piirilevykaaviot helposti saataville internetiin.

LÄHTEET

Android API Guides 2013. Fragments. Viitattu 1.5.2013. http://developer.android.com/guide/components/fragments.html

Android API Guides. Processes and Threads. Viitattu 28.4.2013. http://developer.android.com/guide/components/processes-and-threads.html

Android Developers 2013. Dashboards. Viitattu 1.5.2013. http://developer.android.com/about/dashboards/index.html

Android Developers 2013. Supporting Different Platform Versions. Viitattu 1.5.2013. http://developer.android.com/training/basics/supporting-devices/platforms.html

Android Tools 2013. Get the Android SDK. Viitattu 28.4.2013. http://developer.android.com/sdk/

Android Tools 2013. logcat. Viitattu 28.4.2013. http://developer.android.com/tools/help/logcat.html

Atmel 2009. ATMega 328 Datasheet. Viitattu 10.4.2013. http://www.atmel.com/Images/doc8161.pdf

Atmel AVR109 2004. Selfprogramming. Viitattu 28.4.2013. http://www.atmel.com/Images/doc1644.pdf

Atmel AVR911 2013. AVR Open Source Programmer. Viitattu 28.4.2013. http://www.atmel.com/Images/doc2568.pdf

Bauer, C. 2013. Android ListView with drag and drop reordering. Viitattu 3.4.2013. https://github.com/bauerca/drag-sort-listview

e-Gizmo Mechatronix Central. EGBT-046S/EGBT-045MS Bluetooth Module. Viitattu 10.4.2013. http://www.rasmicro.com/Bluetooth/EGBT-045MS-046S%20Bluetooth%20Module%20Manual%20rev%201r0.pdf

Inkinen, P., Manninen, R., Tuohi, J. 2009. Momentti 2 Insinöörifysiikka 3. painos. Helsinki: Otava.

International Rectifier. IRLI620G Datasheet. Viitattu 16.4.2013. http://www.datasheetcatalog.org/datasheets2/34/347938_1.pdf

Neuroelec 2012. LED Brightness to your eye, Gamma correction – No! http://neuroelec.com/2011/04/led-brightness-to-your-eye-gamma-correction-no/

Oracle 2011. Java SE 6 API Specification. Viitattu 29.8.2013. http://docs.oracle.com/javase/6/docs/api/java/util/concurrent/LinkedBlockingQueue.html

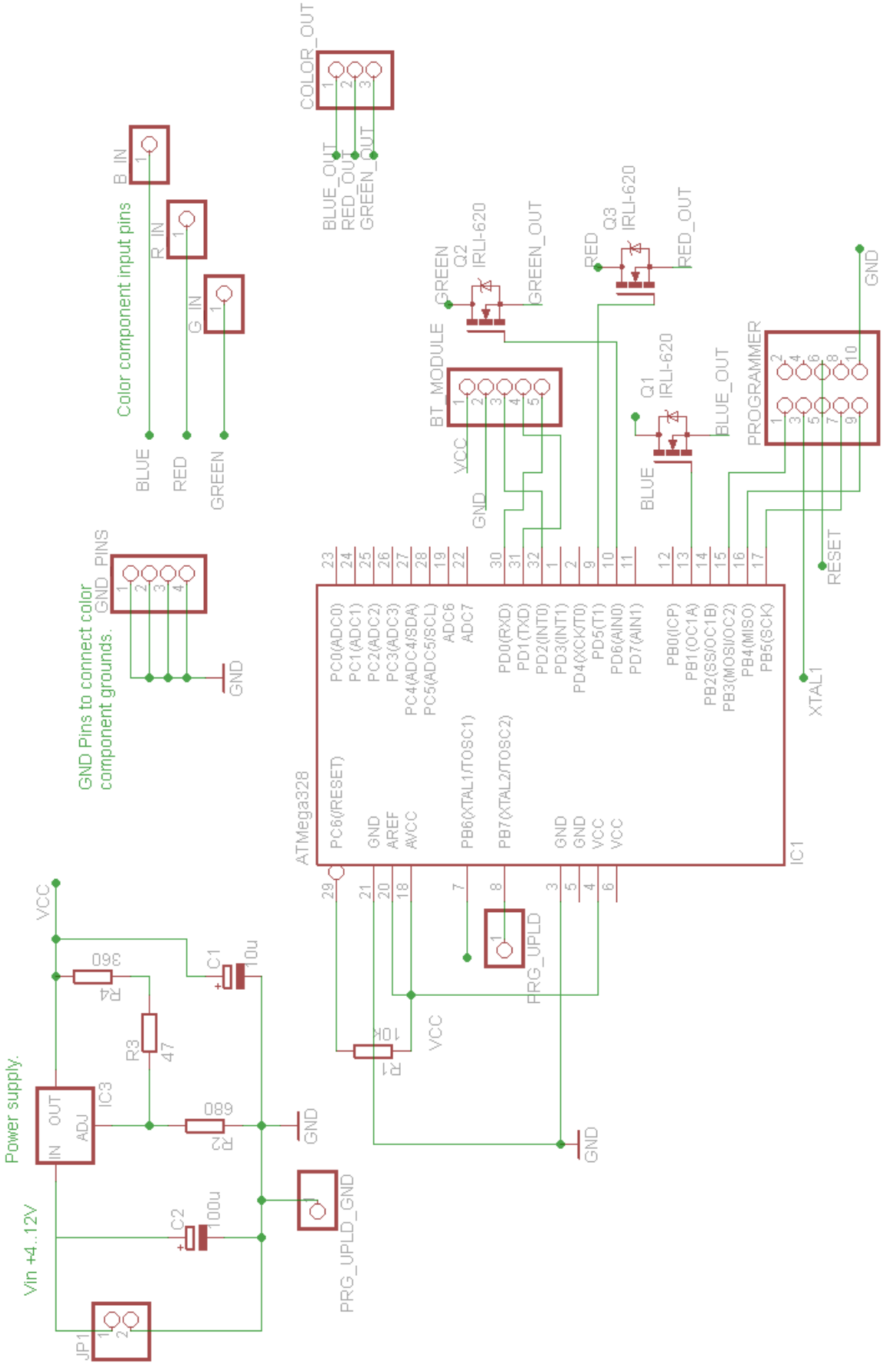
Society of Robots 2007. MICROCONTROLLER UART TUTORIAL. Viitattu 31.3.2013. http://www.societyofrobots.com/microcontroller_uart.shtml

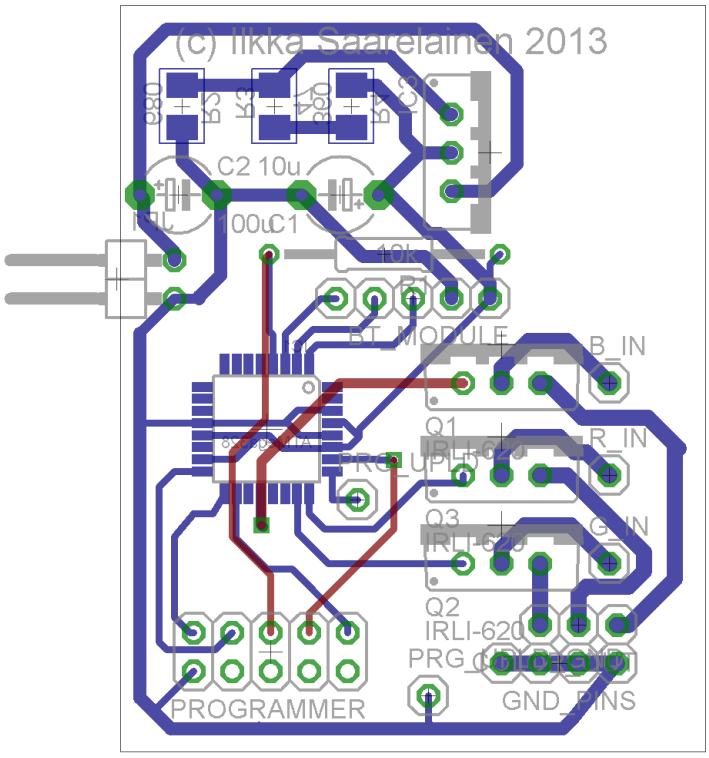
WaitingForFriday 2010. Controlling LED brightness using PWM. http://www.waitingforfriday.com/index.php/Controlling_LED_brightness_using_PWM

Yukuku 2013. Android Color Picker. Viitattu 28.4.2013. https://code.google.com/p/android-color-picker/

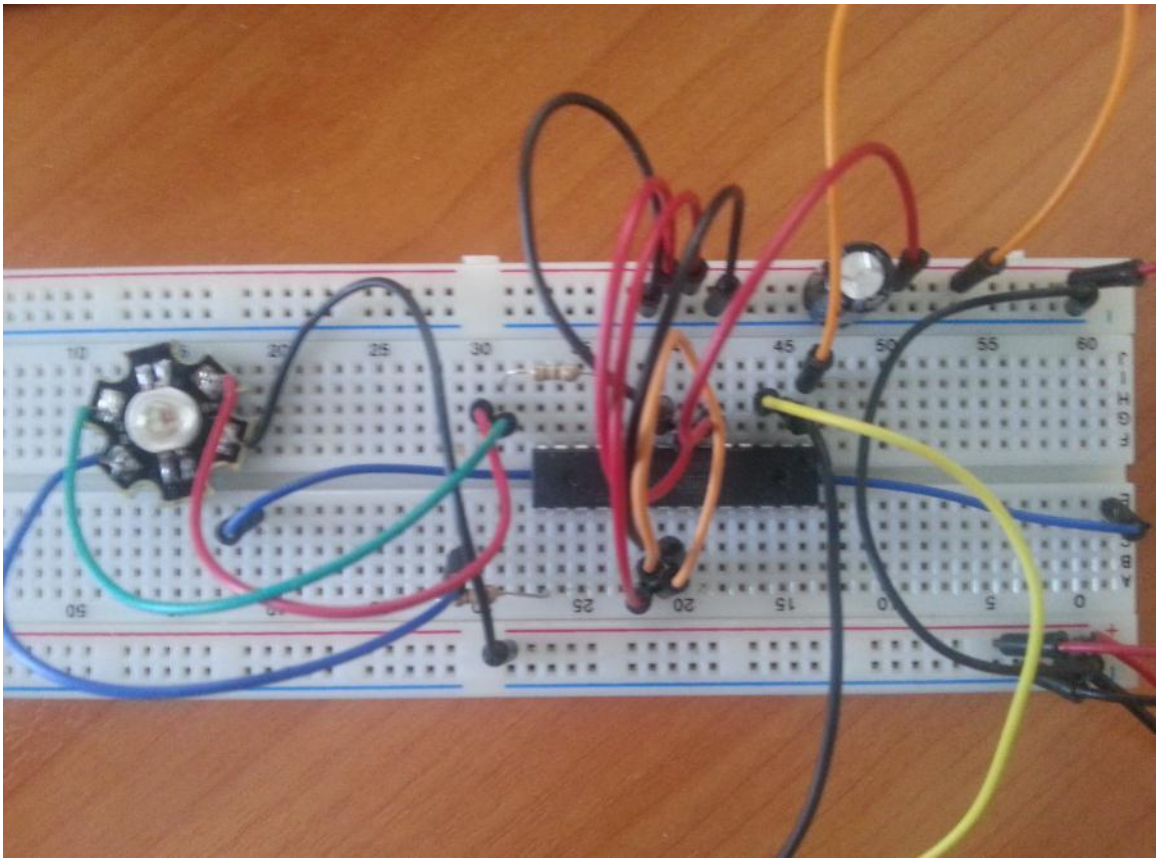
LIITTEET

LIITE 1

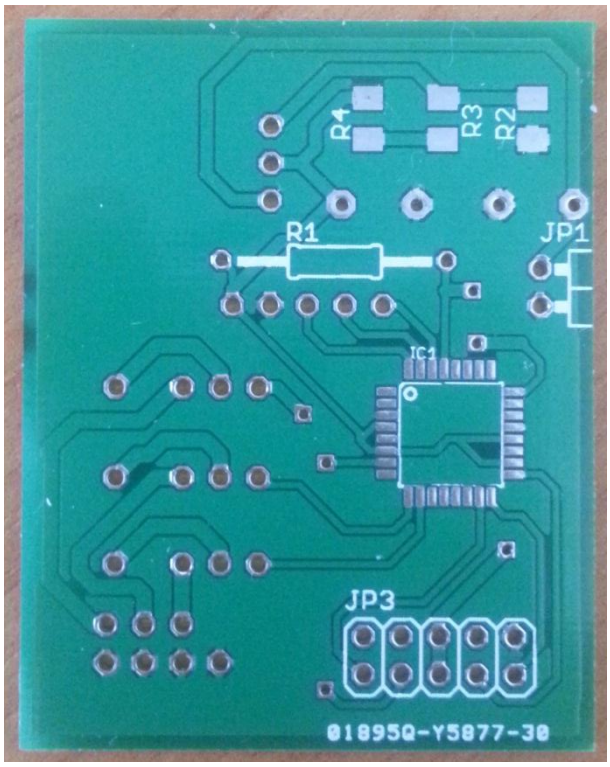

COLOR_OUT
BLUE_OUT
RED_OUT
GREEN_OUT
B_IN
R_IN
G_IN
Color component input pins
BLUE
RED
GREEN
GND_PINS
GND Pins to connect color component grounds.
GND
BT_MODULE
VCC
GND
GREEN
Q2
IRLI-620
GREEN_OUT
RED
Q3
IRLI-620
RED_OUT
Q1
IRLI-620
BLUE
BLUE_OUT
PROGRAMMER
GND
RESET
XTAL1
ATMega328
PC6(/RESET)
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
ADC6
ADC7
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
GND
AREF
AVCC
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
GND
GND
VCC
VCC
IC1
PRG_UPLD
R1
10K
VCC
GND
Power supply.
VCC
R4
360
R3
47
C1
10u
IC3
IN
OUT
ADJ
R2
680
GND
C2
100u
Vin +4..12V
JP1
PRG_UPLD_GND

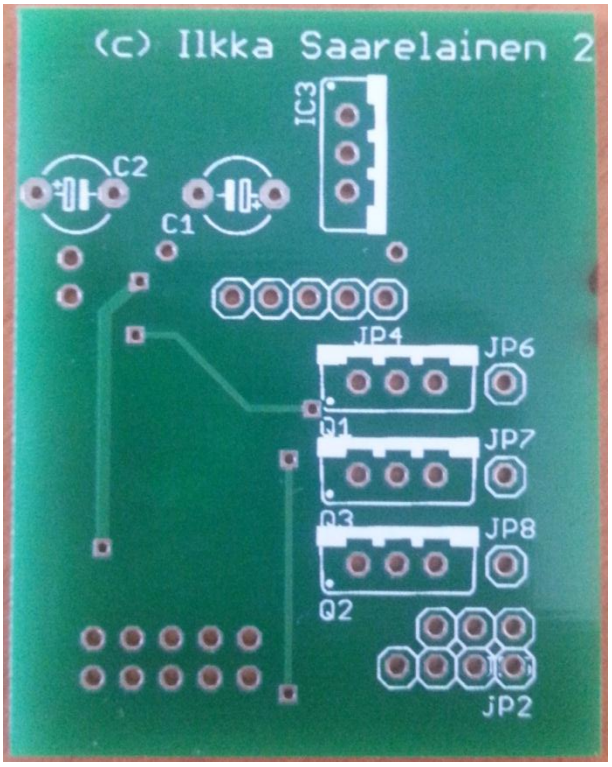

(c) Ilkka Saarelainen 2013
C2 10u
100uC1
10k
BT_MODULE
B_IN
R_IN
G_IN
Q1
IRLI-620
Q3
IRLI-620
Q2
IRLI-620
PROGRAMMER
GND_PINS

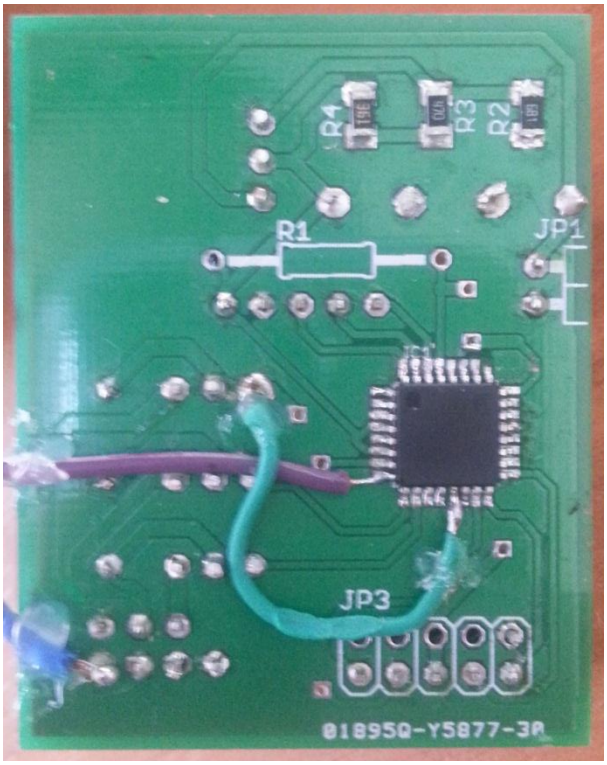
LIITE 3

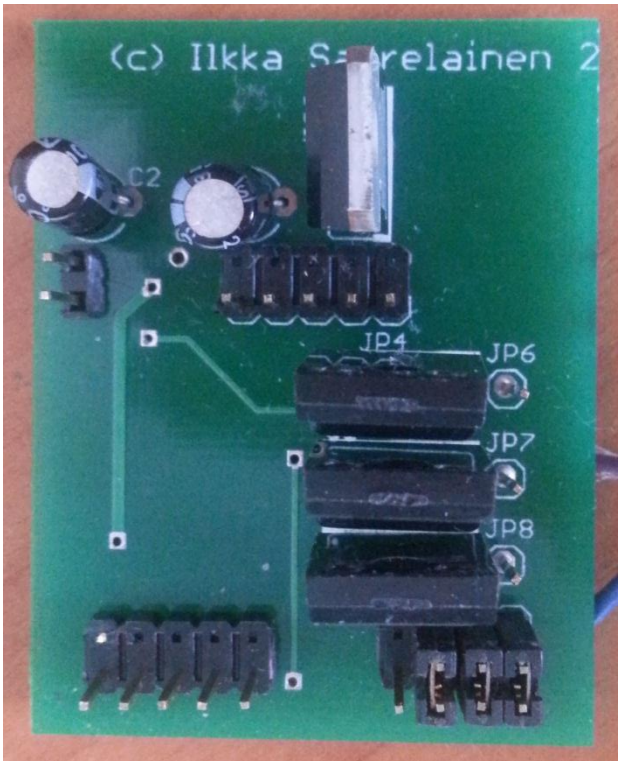


LIITE 4

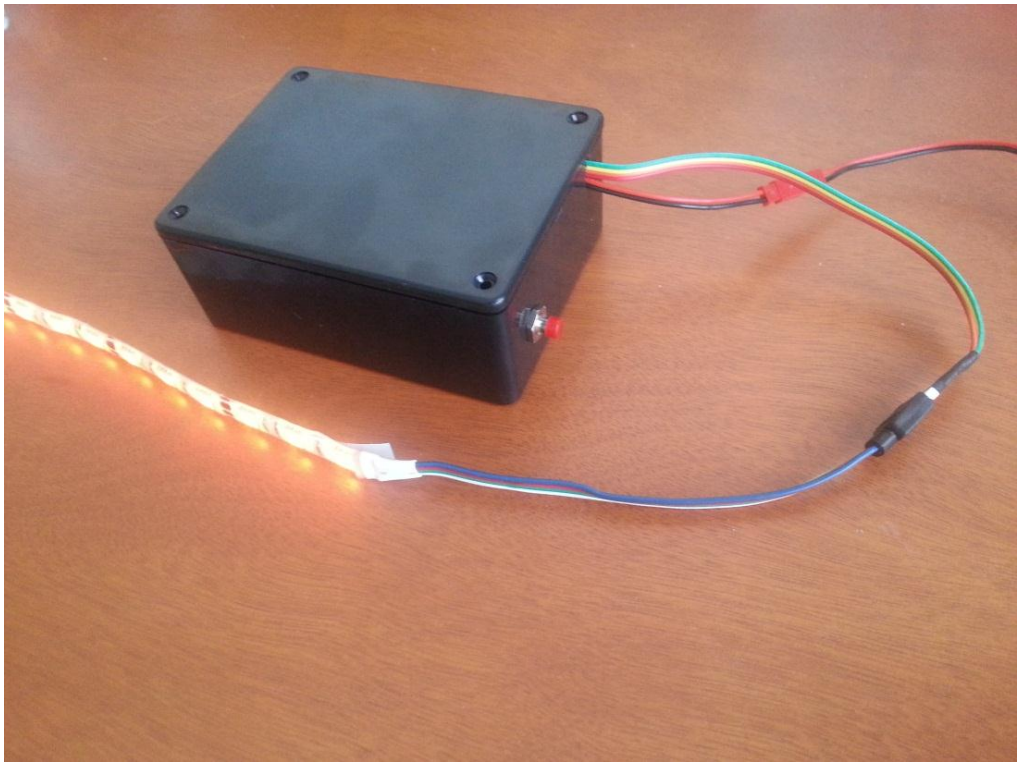

R4
R3
R2
R1
JP1
IC1
JP3
01895Q-Y5877-30


(c) Ilkka Saarelainen 2
IC3
C2
C1
JP4
JP6
Q1
JP7
Q3
JP8
Q2
JP2


R4
R3
R2
R1
JP1
JP3
01895Q-Y5877-30


C2
JP4
JP6
JP7
JP8

LIITE 5



```
/*
 * RGBLight.h
 *
 * Created: 16.3.2013 17:18:26
 * Author: Ilkka Saarelainen
 */

#ifndef RGBVALAISIN_H_
#define RGBVALAISIN_H_

// Define the CPU-freuency.
#define F_CPU 8000000UL

// Define UART port and used pins.
#define UART_PORT_DDR DDRD
#define UART_RX_PIN PIN0
#define UART_TX_PIN PIN1

/*****************************************************************/
/* Type definitions                                              */
/*****************************************************************/
typedef unsigned char uint_8;
typedef signed char   int_8;
typedef unsigned int  uint_16;
typedef signed int    int_16;

// Include general header files.
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/io.h>
#include <avr/wdt.h>
#include <avr/pgmspace.h>

// Function definitions.
static void uartPutChar(uint8_t);
static void uart_init();
static void pwm_init();
void putByte(uint_8);
void finishCommandWrite();
void initCommandWriter();
uint_8 numberOfArguments(uint_8);
void switchOff();
void uartSendFlashString(const char*, uint_8);
void uartSendString(char*, uint_8);
void setRedPWM(uint_8);
void setGreenPWM(uint_8);
void setBluePWM(uint_8);

/*****************************************************************/
/* Global values of color components                             */
/*****************************************************************/
#define RED   0
#define GREEN 1
#define BLUE  2

#define RED_BIT   0x01
#define GREEN_BIT 0x02
#define BLUE_BIT  0x04

/*****************************************************************/
/* Registers to set color-values                                 */
/*****************************************************************/
```

```
#define RED_PWM_PIN    PORTD6
#define GREEN_PWM_PIN  PORTD6
#define BLUE_PWM_PIN   PORTB1

#define RED_PWM_PORT   PORTD
#define GREEN_PWM_PORT PORTD
#define BLUE_PWM_PORT  PORTB

#define RED_PWM_DDR    DDRD
#define GREEN_PWM_DDR  DDRD
#define BLUE_PWM_DDR   DDRB

uint_8 gColorCorrection = 1;

// Macro to easily access the CIEL-table.
#define CIELPWM(A) (pgm_read_byte(CIEL + A))

// Pre-calculated lookup table for color correction.
const uint_8 CIEL[] PROGMEM = {
0,    1,    2,    3,    4,    5,    7,    9,    12,
15,   18,   22,   27,   32,   38,   44,   51,   58,
67,   76,   86,   96,   108,  120,  134,  148,  163,
180,  197,  216,  235,  255};

#define RED_PWM_   OCR0A
#define GREEN_PWM_ OCR0B
#define BLUE_PWM_  OCR1AL

/*************************************************************************/
/* General global variables and defines.                                 */
/*************************************************************************/

uint_16 gArgumentsRemaining = 0; // Number of not yet received arguments.
uint_8  gCommand = 0;   // Last received command will be stored here
uint_8  arg[64];        // Arguments will be stored here.

// List of all control commands.
#define SET_R              0xC1
#define SET_G              0xC2
#define SET_B              0xC3
#define SET_RGB            0xC4
#define DOWNLOAD_PROGRAM   0xC6
#define RESET              0xC7
#define GET_RGB            0xC9
#define GET_COLOR_CORR     0xCC
#define TOGGLE_COLOR_CORR  0xCD
#define SWICH_OFF          0xCF
#define BLUETOOTH_CONTROL  0xCE

// Number of expected arguments per control command.
#define SET_X_ARGS             1
#define SET_RGB_ARGS           3
#define DOWNLOAD_PROGRAM_ARGS  2
#define SWICH_OFF_ARGS         0
#define BLUETOOTH_CONTROL_ARGS 2
#define TOGGLE_COLOR_CORR_ARGS 0
#define GET_COLOR_CORR_ARGS    0
#define GET_RGB_ARGS           0

// Mask to detect a command.
#define COMMAND_START_MASK 0b11000000

/*************************************************************************/
/* Light program related definitions and variables                       */
/*************************************************************************/
```

```
// Light program command types enumeration.
enum CommandType
{
    RED_SEQ_START   = 1,
    GREEN_SEQ_START = 2,
    BLUE_SEQ_START  = 3,
    FADE_LINEAR = 4,
    SET_COLOR   = 5,
    WAIT_COLOR  = 6,
    NOTIFY_COLOR= 7,
    WAIT = 8
};
// Light program command structure.
struct Command
{
    uint8_t command;
    uint8_t arg1;
    int8_t arg2;
};

// Number of arguments per light program command.
#define FADE_LINEAR_ARGS  2
#define WAIT_ARGS         2
#define SET_COLOR_ARGS    1
#define WAIT_COLOR_ARGS   1
#define NOTIFY_COLOR_ARGS 1

uint_8 remainingBytes = 0;
Command* commandWriter_p = 0;

// Global command table
Command commandTable[600];
// Remaining commands while receiving light program.
uint_16 gCommandsRemaining = 0;
// Current color while writing to the command table.
uint_8 gCommandTableCurrentColor = 0;
// Flag to set light program download state.
uint_8  gDownloadStarted = 0;
// Flag to set the state of the light program.
uint_8 gProgramRunning = 0;

// Characters remaining of a Bluetooth command.
uint_8  gBluetoothCharsRemaining = 0;
// Total number of characters of a Bluetooth command.
uint_8  gBluetoothTotalChars = 0;
// Bluetooth command identifier.
uint_8  gBluetoothCommand = 0;

/*********************************************************************/
/* Bluetoot control related global variables                         */
/*********************************************************************/

// Bluetooth module control-mode pin
#define CONTROL_PIN PORTB7

// Bluetooth module control commands (saved on the flash)
const char BLUETOOTH_SET_PIN_STRING[] PROGMEM  = "AT+PSWD=";
const char BLUETOOTH_SET_NAME_STRING[] PROGMEM  = "AT+NAME=";

// Bluetooth commands
#define BLUETOOTH_SET_PIN  0x01
#define BLUETOOTH_SET_NAME 0x02

// Include project related header files.
#include "ColorComponent.h"
#endif /* RGBVALAISIN_H_ */
```

```
/*
 * RGBLight.cpp
 *
 * Created: 16.3.2013 17:00:11
 *  Author: Ilkka Saarelainen
 */

#include "RGBLight.h"

void uart_init()
{
    UBRR0L = 103;
    // Use the double speed.
    UCSR0A |= (1 << U2X0);

    // Enable RX and TX.
    UCSR0B |= (1 << RXEN0) | (1 << TXEN0);

    // 8-bit, 1 stop bit, no parity, asynchronous UART
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00) | (0 << USBS0) |
        (0 << UPM01) | (0 << UPM00) | (0 << UMSEL01) |
        (0 << UMSEL00);

    // Enable receive complete interrupts.
    UCSR0B |= (1 << RXCIE0);

    // Set the TX pin as output.
    UART_PORT_DDR |= (1 << UART_TX_PIN);
    // Set the RX pin as input.
    UART_PORT_DDR &= ~(1 << UART_RX_PIN);

    sei();
}

static void pwm_init()
{
    //Note: PWM will output to pin "OC0A" (PD6) and "OC0B (PD5).
    //Set pin to output
    RED_PWM_PORT &= ~(1 << RED_PWM_PIN);
    RED_PWM_DDR  &= ~(1 << RED_PWM_PIN);

    GREEN_PWM_PORT &= ~(1 << GREEN_PWM_PIN);
    GREEN_PWM_DDR  &= ~(1 << GREEN_PWM_PIN);

    BLUE_PWM_PORT &= ~(1 << BLUE_PWM_PIN);
    BLUE_PWM_DDR  &= ~(1 << BLUE_PWM_PIN);

    GREEN_PWM_PORT &= ~(1 << GREEN_PWM_PIN);
    DDRD |= (1 << PORTD6) | (1 << PORTD5);

    PORTB &= ~(1 << PORTB1);
    DDRB |= (1 << PORTB1);

    ////////////////////////////////////////////////////////////////////
    // timer 0
    // Set compare A-mode to inverting (COM0A*) and Waweform generaion mode
    // to Fast PWM (WGM0*).
    TCCR0A |= (1 << COM0A1) |(1 << WGM01) | (1 << WGM00);
    TCCR0A |= (1 << WGM01) | (1 << WGM00);
    //Set compare B-mode to inverting (COM0B*)
    TCCR0A |= (1 << COM0B1);
    //8-prescaler
    TCCR0B |=  (1 << CS00);
```

```
    ////////////////////////////////////////////////////////////////////
    // timer 1
    // Set compare A-mode to inverting (COM1A*) and Waweform generaion mode
    TCCR1A |= (1 << COM1A1) | (1 << WGM12) | (1 << WGM10);
    TCCR1A |= (1 << WGM12) | (1 << WGM10);
    TCCR1B |= (1 << CS10);

    TCNT0 = 0x00;
    TCNT1 = 0x00;

    OCR0A = 0x00;
    OCR0B = 0x00;
    OCR1A = 0x00;
    // Turn off all PWM-outputs
    setRedPWM(0);
    setGreenPWM(0);
    setBluePWM(0);

    ////////////////////////////////////////////////////////////////////
    // Timer 2 - used as a tick generator.
    // - Compare mode.
    // - 1024 prescaler.
    TCCR2A |= (1 << WGM21);
    TCCR2B |= (1 << CS22)| (1 << CS21) | (1 << CS20);
    TIMSK2 |= (1 << OCIE2A);

    // Clear counter on compare match.
    TIFR2 |= (1 << OCF2A);

    TCNT2 = 0x00;

    // 195 = about 0,025s interrupt interval.
    OCR2A = 195;
}
/*********************************************************************/
/* Interrupt-function for UART receiving                             */
/*********************************************************************/
ISR(USART_RX_vect)
{
    // Disable interrupts.
    cli();

    uint_8 receivedByte;
    receivedByte = UDR0;

    // If previous command is completed, check if message
    // contains a new command.
    if(gArgumentsRemaining == 0 &&
    (receivedByte & COMMAND_START_MASK) == COMMAND_START_MASK)
    {
        gArgumentsRemaining = numberOfArguments(receivedByte);
        gCommand = receivedByte;
        //uartPutChar(receivedByte);

        // Disable light program if static color is set.
        if(gCommand == SET_R || gCommand == SET_G || gCommand == SET_B
            || gCommand == SET_RGB)
        {
            gProgramRunning = 0;
        }
    }
    // In case of program download.
    else if(gCommand == DOWNLOAD_PROGRAM &&
            !gDownloadStarted)
    {
```

```
    // Get high bits first.
    if(gArgumentsRemaining == 2)
    {
        gCommandsRemaining = (receivedByte << 8);
        gArgumentsRemaining--;
    }
    // Then get low bits.
    else
    {
        gCommandsRemaining |= receivedByte;
        // Set program download to started.
        gDownloadStarted = 1;

        // Set arguments remaining to commands remaining + 1
        // because of the checksum.
        gArgumentsRemaining = gCommandsRemaining;

        // Init command writer
        initCommandWriter();

        // Stop running program
        gProgramRunning = 0;
    }
}
else if(gCommand == DOWNLOAD_PROGRAM &&
        gCommandsRemaining)
{
    putByte(receivedByte);
    gCommandsRemaining--;
    gArgumentsRemaining--;

    // Set download started to 0
    if(gCommandsRemaining == 0)
    {
        gDownloadStarted = 0;
        // Finalize the command writing.
        finishCommandWrite();
    }
}
// Read all characters of the bluetooth control command.
else if (gCommand == BLUETOOTH_CONTROL &&
        gBluetoothCharsRemaining)
{
    arg[gBluetoothTotalChars - gBluetoothCharsRemaining]
        = receivedByte;
    gBluetoothCharsRemaining--;
}
else if(gCommand == BLUETOOTH_CONTROL)
{
    //uartPutChar('f');
    // The first byte contains the command identifier.
    if(gArgumentsRemaining == 2)
    {
        gBluetoothCommand = receivedByte;
    }
    // Second byte contains number of remaining characters.
    else
    {
        gBluetoothCharsRemaining = receivedByte;
        gBluetoothTotalChars     = receivedByte;
    }
    gArgumentsRemaining--;
}

// All received bytes should be arguments as long as
// "gArgumentsRemaining > 0".
```

```
    else if(gArgumentsRemaining > 0)
    {
        gArgumentsRemaining--;
        arg[gArgumentsRemaining] = receivedByte;
    }

    // Enable interrupts.
    sei();
}

ColorComponent r(&setRedPWM, RED_BIT);
ColorComponent g(&setGreenPWM, GREEN_BIT);
ColorComponent b(&setBluePWM, BLUE_BIT);

ISR(TIMER2_COMPA_vect)
{
    if(gProgramRunning)
    {
        r.execute();
        g.execute();
        b.execute();
    }
}

void uartPutChar(uint8_t data)
{
    // Make sure that the UART buffer is empty
    while (!(UCSR0A & (1 << UDRE0)));
    UDR0 = data;
}

/*
 * This function resets the controller via watch dog.
 */
void reset()
{
    cli();
    wdt_enable(WDTO_500MS);
    while(true);
}

static void executeLastCommand()
{
    switch(gCommand)
    {
    case SET_R:
        setRedPWM(arg[0]);
        if(!gArgumentsRemaining)gCommand = 0;
        break;
    case SET_G:
        setGreenPWM(arg[0]);
        if(!gArgumentsRemaining)gCommand = 0;
        break;
    case SET_B:
        setBluePWM(arg[0]);
        if(!gArgumentsRemaining)gCommand = 0;
        break;
    case SET_RGB:
        setRedPWM(arg[2]);
        setGreenPWM(arg[1]);
        setBluePWM(arg[0]);
        if(!gArgumentsRemaining)gCommand = 0;
        break;
    case DOWNLOAD_PROGRAM:
        // All functionality is implemented in the UART-
        // receive function.
```

```
            break;
        case TOGGLE_COLOR_CORR:
            gColorCorrection = !gColorCorrection;
            uartPutChar(gColorCorrection);
            if(!gArgumentsRemaining)gCommand = 0;
            break;
        case RESET:
            reset();
            break;
        case GET_COLOR_CORR:
            uartPutChar(gColorCorrection);
            if(!gArgumentsRemaining)gCommand = 0;
            break;
        case GET_RGB:
            uartPutChar(RED_PWM_);
            uartPutChar(GREEN_PWM_);
            uartPutChar(BLUE_PWM_);
            if(!gArgumentsRemaining)gCommand = 0;
            break;
        case SWICH_OFF:
            if(!gArgumentsRemaining)gCommand = 0;
            switchOff();
            break;
        case BLUETOOTH_CONTROL:
            // Check if all command characters are already received
            if(!gBluetoothCharsRemaining)
            {
                // Set the control pin to high.
                PORTB |= (1 << CONTROL_PIN);
                // Wait bluetooth module to enter command mode.
                _delay_ms(10);
                switch(gBluetoothCommand)
                {
                case BLUETOOTH_SET_PIN:
                    uartSendFlashString(BLUETOOTH_SET_PIN_STRING, 8);
                    uartSendString((char *)arg, gBluetoothTotalChars);
                    break;
                case BLUETOOTH_SET_NAME:
                    uartSendFlashString(BLUETOOTH_SET_NAME_STRING, 8);
                    uartSendString((char *)arg, gBluetoothTotalChars);
                    break;
                }
                // Clear the control pin.
                _delay_ms(10);
                PORTB &= ~(1 << CONTROL_PIN);
                // Clear the command
                gCommand = 0;
                gBluetoothCommand = 0;
                gBluetoothTotalChars = 0;
            }
            break;
        default:
            break;
        }
}

void uartSendFlashString(const char* aString, uint_8 aLenght)
{
    for(int i = 0; i < aLenght; i++)
    {
        uartPutChar(pgm_read_byte(aString++));
    }
}

void uartSendString(char* aString, uint_8 aLenght)
{
```

```
    for(int i = 0; i < aLenght; i++)
    {
        uartPutChar(aString[i]);
    }
}

/*
* Turns off all the PWM-outputs.
*/
void switchOff()
{
    // Stop light program
    gProgramRunning = 0;

    setRedPWM(0);
    setGreenPWM(0);
    setBluePWM(0);
}

/*
 * This function is used to generate new commands byte
 * by byte and push them to the table.
 */
void putByte(uint_8 aInput)
{
    // If remainingBytes is 0, must be a beginning of a
    // new command.
    if(remainingBytes == 0)
    {
        // Save the command.
        commandWriter_p->command = aInput;
        commandWriter_p->arg1 = 0;
        commandWriter_p->arg2 = 0;

        // Define amount of arguments expected based on
        // command type.
        switch(aInput)
        {
        case FADE_LINEAR:
            remainingBytes = FADE_LINEAR_ARGS;
            break;
        case SET_COLOR:
            remainingBytes = SET_COLOR_ARGS;
            break;
        case WAIT_COLOR:
            remainingBytes = WAIT_COLOR_ARGS;
            break;
        case NOTIFY_COLOR:
            remainingBytes = NOTIFY_COLOR_ARGS;
            break;
        case WAIT:
            remainingBytes = WAIT_ARGS;
            break;
        case RED_SEQ_START:
            gCommandTableCurrentColor = RED_SEQ_START;
            remainingBytes = 0;
            // Set the start point to the next item in the command table.
            r.mStart_p = commandWriter_p + 1;
            break;
        case GREEN_SEQ_START:
            gCommandTableCurrentColor = GREEN_SEQ_START;
            remainingBytes = 0;
            g.mStart_p = commandWriter_p + 1;
            break;
        case BLUE_SEQ_START:
            gCommandTableCurrentColor = BLUE_SEQ_START;
```

```
            remainingBytes = 0;
            b.mStart_p = commandWriter_p + 1;
            break;
        case DOWNLOAD_PROGRAM:
            return;
        }

        // Set the end pointers for color.
        switch(gCommandTableCurrentColor)
        {
        case RED_SEQ_START:
            r.mEnd_p = commandWriter_p;
            break;
        case GREEN_SEQ_START:
            g.mEnd_p = commandWriter_p;
            break;
        case BLUE_SEQ_START:
            b.mEnd_p = commandWriter_p;
            break;
        }

        //If the command does not have any arguments, increase the writer pointer
        if(!remainingBytes)
        {
            commandWriter_p++;
        }
    }
    // Else save arguments.
    else
    {
        switch(remainingBytes)
        {
        case 2:
            commandWriter_p->arg2 = aInput;

            break;
        case 1:
            commandWriter_p->arg1 = aInput;
            // No more arguments -> move writer to the next
            // cell in the table.
            commandWriter_p++;
            break;
        }
        remainingBytes--;
    }
}

/*
 * This function must be called after all the commands are wrote
 * to the command table.
 */
void finishCommandWrite()
{
    r.mCp = r.mStart_p;
    g.mCp = g.mStart_p;
    b.mCp = b.mStart_p;

    //Start the program.
    gProgramRunning = 1;
}

/*
 * This function resets the command writer. Must be called
 * before every use of the command writer.
 */
void initCommandWriter()
```

```
{
    commandWriter_p = commandTable;

    r.reset();
    g.reset();
    b.reset();
}

/*
 * Returns the number of arguments of given Connection-
 * protocol command.
 * NOTE: This function does NOT work with light program commands!
 * @param[in] aCommand Input command.
 * @return The number of arguments of the given command.
 */
uint_8 numberOfArguments(uint_8 aCommand)
{
    switch(aCommand)
    {
    case SET_G:
        return SET_X_ARGS;
    case SET_R:
        return SET_X_ARGS;
    case SET_B:
        return SET_X_ARGS;
    case SET_RGB:
        return SET_RGB_ARGS;
    case DOWNLOAD_PROGRAM:
        return DOWNLOAD_PROGRAM_ARGS;
    case SWICH_OFF:
        return SWICH_OFF_ARGS;
    case BLUETOOTH_CONTROL:
        return BLUETOOTH_CONTROL_ARGS;
    case TOGGLE_COLOR_CORR:
        return TOGGLE_COLOR_CORR_ARGS;
    case GET_COLOR_CORR:
        return GET_COLOR_CORR_ARGS;
    case GET_RGB:
        return GET_RGB_ARGS;
    default:
        return 0;
    }
}

/*
 * This function initializes the color components.
 * This function must be called before running
 * the first light program.
 */
void initColorComponents()
{
    r.setGreen(&g);
    r.setBlue(&b);

    g.setRed(&r);
    g.setBlue(&b);

    b.setRed(&r);
    b.setGreen(&g);
}

/*
 * Sets the PWM-output of the red color component brightness to the
 * value given in the argument.
 * If the color correction is enabled, the real brightness is
 * read from the CIELPWM-lookup table.
```

```
 */
void setRedPWM(uint_8 aValue)
{
    if(aValue == 0)
    {
        // Disconnect the pin from PWM-generation.
        TCCR0A &= ~((1 << COM0A1) | (1 << COM0A0));
        RED_PWM_PORT &= ~(1 << RED_PWM_PIN);
    }
    else
    {
        // Enable non-inverting PWM for this pin.
        TCCR0A |= (1 << COM0A1);
    }

    if(gColorCorrection)
        RED_PWM_ = CIELPWM((aValue >> 3));
    else
        RED_PWM_ = aValue;
}

/*
 * Sets the PWM-output of the green color component brightness to the
 * value given in the argument.
 * If the color correction is enabled, the real brightness is
 * read from the CIELPWM-lookup table.
 */
void setGreenPWM(uint_8 aValue)
{
    if(aValue == 0)
    {
        // Disconnect the pin from PWM-generation.
        TCCR0A &= ~(1 << COM0B1);
        GREEN_PWM_PORT &=  ~(1 << GREEN_PWM_PIN);
    }
    else
    {
        // Enable non-inverting PWM for this pin.
        TCCR0A |= (1 << COM0B1);
    }
    if(gColorCorrection)
        GREEN_PWM_ = CIELPWM((aValue >> 3));
    else
        GREEN_PWM_ = aValue;
}

/*
 * Sets the PWM-output of the blue color component brightness to the
 * value given in the argument.
 * If the color correction is enabled, the real brightness is
 * read from the CIELPWM-lookup table.
 */
void setBluePWM(uint_8 aValue)
{
    if(aValue == 0)
    {
        // Disconnect the pin from PWM-generation.
        TCCR1A &= ~(1 << COM1A1);// | (1 << COM1A0));
        BLUE_PWM_PORT &= ~(1 << BLUE_PWM_PIN);
    }
    else
    {
        // Enable non-inverting PWM for this pin.
        TCCR1A |= (1 << COM1A1);
    }
    if(gColorCorrection)
```

```
        BLUE_PWM_ = CIELPWM((aValue >> 3));
    else BLUE_PWM_ = aValue;
}

/*
 * Main function.
 */
int main()
{
    // Initialize colorComponents.
    initColorComponents();

    gCommand = 0;
    // Init UART
    uart_init();
    // Init command writer.
    initCommandWriter();
    // Init PWM.
    pwm_init();

    //Set Bluetooth module control pin.
    PORTB = 0;
    DDRB |= (1 << CONTROL_PIN);

    while(true)
    {
        if(gArgumentsRemaining == 0 && gCommand != 0)
        {
            executeLastCommand();
        }
        // Weird bug: the above if-clause can not be alone in
        // the while loop. If it's alone, the if condition is
        // newer true.
        asm("nop");
    }
}
```

```
/*
 * ColorComponent.h
 *
 * Created: 16.3.2013 17:31:33
 * Author: Ilkka Saarelainen
 */
#ifndef COLORCOMPONENT_H_
#define COLORCOMPONENT_H_

class ColorComponent
{
public:
    // Constructor.
    ColorComponent(
         void (*pwmSetFunc)(uint_8), uint_8 aColorBit):
         setPWM(pwmSetFunc),
         mMyColorBit(aColorBit)
    {
        mCp = 0;
        mEnd_p = 0;
        mStart_p = 0;
    }

    void execute()
    {
        bool commandFinished = false;
        // Do nothing if any of the wait flags are set.
        if(mColorsToWait)
        {
            return;
        }
        switch(mCp->command)
        {
            case FADE_LINEAR:
                commandFinished = addToColor(mCp->arg2,
                        mCp->arg1);
                break;
            case SET_COLOR:
                mCurrentColor = mCp->arg1;
                commandFinished = true;
                break;
            case NOTIFY_COLOR:

                if(mCp->arg1 & RED_BIT)
                {
                    mRed->mColorsToWait &= ~mMyColorBit;
                }
                if(mCp->arg1 & GREEN_BIT)
                {
                    mGreen->mColorsToWait &= ~mMyColorBit;
                }
                if(mCp->arg1 & BLUE_BIT)
                {
                    mBlue->mColorsToWait &= ~mMyColorBit;
                }
                commandFinished = true;
                break;
            case WAIT_COLOR:
                // Set colors to wait.
                mColorsToWait = mCp->arg1;
                commandFinished = true;
                break;
```

```
            case WAIT:
                commandFinished = wait();
            default:
                break;
        }
        // Update the color.
        setPWM(((uint_8)mCurrentColor));
        // Load next command.
        if(commandFinished)
        {
            loadNextCommand();
        }
    }

    /*
     * Resets all member variables to 0,
     * except pointers to other colors.
     */
    void reset()
    {
        mCp      = 0;
        mEnd_p   = 0;
        mStart_p = 0;
        mColorsToWait = 0;
        mCurrentColor = 0;
        mWaitCounter  = 0;
    }

    void setRed(ColorComponent *aColor)
    {
        mRed = aColor;
    }

    void setGreen(ColorComponent *aColor)
    {
        mGreen = aColor;
    }

    void setBlue(ColorComponent *aColor)
    {
        mBlue = aColor;
    }
    /*
     * Command table pointers.
     */
    // Pointing to the start of command table.
    Command* mStart_p;
    // Pointing to the end of the table.
    Command* mEnd_p;
    // Command pointer.
    Command* mCp;

    // Colors to wait.
    uint_8 mColorsToWait;

protected:

    int_16 mCurrentColor;
    // Wait counter.
    uint_16 mWaitCounter;
    void (*setPWM)(uint_8);

    // Other colors.
    ColorComponent* mRed;
    ColorComponent* mGreen;
    ColorComponent* mBlue;
```

```
    // The bit mask of this color component.
    uint_8 mMyColorBit;

    void loadNextCommand()
    {
        // If command pointer is already pointing to the end
        // of the table, jump to the beginning.
        if(mCp == mEnd_p)
        {
            // + 1 because the first command is the sequence start command and should
not be executed.
            mCp = mStart_p;
            return;
        }
        // Move command pointer to the next command.
        mCp++;
    }

    /*
     * Implements the wait function.
     * Note: wait time should be at least 1. 0 will cause malfunction.
     * @return 1 if waiting is completed, otherwise 0.
     */
    uint_8 wait()
    {
        // If wait counter is 0, this is the first tick for this
        // command.
        if(mWaitCounter == 0)
        {
            // Set value from the command table.
            mWaitCounter = (mCp->arg2 << 8) | mCp->arg1;
        }
        mWaitCounter--;
        if(mWaitCounter == 0)
        {
            return 1;
        }
        return 0;
    }

    /*
     * Adds value to the current color. The color value is clamped to the valid
     * color range (0-255) and/or to the limit value.
     * @param[in] aValue A value to be added to this color component.
     * @param[in] aTargetValue A limiting value.
     * @return 1 if the target value is reached, else 0.
     */
    uint_8 addToColor(int_8 aValue, uint_8 aTargetValue)
    {
        uint_8 sign;
        sign = aValue & 0x80;

        mCurrentColor += aValue;
        // Set to 0 if negative.
        if(mCurrentColor & 0x8000)
        {
            mCurrentColor = 0;
        }
        // Clamp the value to 255 if it's too big.
        else if(mCurrentColor & 0x0100)
        {
            mCurrentColor = 255;
        }

        // Check if target color value is reached.
```

```
        if(sign) // Decreasing color value.
        {
            if(mCurrentColor <= aTargetValue)
            {
                mCurrentColor = aTargetValue;
                return 1;
            }
        }
        else // Increasing color value.
        {
            if(mCurrentColor >= aTargetValue)
            {
                mCurrentColor = aTargetValue;
                return 1;
            }
        }
        return 0;
    }
};

#endif /* COLORCOMPONENT_H_ */
```

```java
/**
 * File: BluetoothConnectionHandler.java
 */
package com.ile.moodlight;

import java.io.IOException;
import java.io.InputStream;
import java.io.InterruptedIOException;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.util.concurrent.LinkedBlockingQueue;

import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.os.Message;
import android.util.Log;

import com.ile.moodlight.activity.TopContainerActivity;

/**
 * This thread handles Bluetooth connection related tasks,
 * including establishing a connection, keeping the connection alive and
 * terminating the connection.
 *
 * This thread has two child threads: one for sending and one for receiving the
 * data. These child threads should be accessed only via this thread.
 *
 * There should be only one instance of this thread at a time.
 * @author Ilkka Saarelainen
 *
 */
public class BluetoothConnectionHandler extends Thread
{
    /**
     * Tag used in debugging.
     */
    private static final String LOG_NAME = "BluetoothConnectionHandler";
    /**
     * A message code used to notify the main thread about successfully created
     * connection to the target device. This is not used at the moment.
     */
    public static final int MSG_CONNECTED = 0x01;
    /**
     * A message code used to notify the main thread about disconnection.
     */
    public static final int MSG_DISCONNECTED = 0x02;
    /**
     * Depth of the input queue.
     */
    private static final int INPUT_QUEUE_SIZE = 128;
    /**
     * Depth of the output queue.
     */
    private static final int OUTPUT_QUEUE_SIZE = 50000;
    /**
     * Flag containing info about the connection state.
     */
    private static Boolean connected = false;
    /**
     *  The target device.
     */
```

```
private final BluetoothDevice  target;
/**
 *  Socket to get input and output streams.
 */
private final BluetoothSocket  socket;
/**
 *  The queue for the incoming data. The received data can be read directly from
 *  the queue from any thread.
 */
public final LinkedBlockingQueue<Integer> inputQueue;
/**
 *  The queue for the outgoing data. Output data must be written to the queue
 *  via sendData() -method.
 */
private final LinkedBlockingQueue<Integer> outputQueue;
/**
 * Thread for receiving the data.
 */
private InputHandler inputHandler;
/**
 * Thread for sending the data.
 */
private OutputHandler outputHandler;
/**
 * Flag to determine the running state of this thread.
 */
private boolean running = false;
/**
 * Flag to tell if the error message have been sent to the main thread.
 */
private boolean disconnectMessageSent = false;

/**
 * Constructor.
 * The constructor tries to open a bluetooth serial socket (RFCOMM) to the given
 * target device. Once the connection is established, start() -method can be
 * called. <br/>
 * If the socket cannot be opened, this thread is useless.<br/>
 * <br/>
 * @param aTarget A target device where the connection will be established.
 * @throws Exception if the socket can not be opened.
 */
public BluetoothConnectionHandler(BluetoothDevice aTarget) throws Exception
{
    connected = false;
    this.target = aTarget;

    BluetoothSocket temp = null;
    // Try to get a bluetooth serial socket for a target device.
    try
    {
        // Call a createRfcommSocket -method.
        Method method = this.target.getClass().
                getMethod("createRfcommSocket", new Class[] {int.class});
        temp = (BluetoothSocket) method.invoke(this.target, 1);
    }
    catch (Exception ex)
    {
        Log.e(LOG_NAME, "Unable to create socket for given target device", ex);
        throw new Exception("Unable to create socket for given target device.");
    }

    this.socket = temp;

    // Create queues.
    this.inputQueue = new LinkedBlockingQueue<Integer>(INPUT_QUEUE_SIZE);
```

```
            this.outputQueue =
                    new LinkedBlockingQueue<Integer>(OUTPUT_QUEUE_SIZE);
        }

        @Override
        public void run()
        {
            this.running = true;
            // Try to open the connection.
            try
            {
                this.openConnection();
                this.inputHandler = new InputHandler();
                this.outputHandler = new OutputHandler();

                this.inputHandler.start();
                this.outputHandler.start();

                // Set status to connected.
                connected = true;
            }
            catch(Exception ex)
            {
                // Stop the thread.
                Log.e(LOG_NAME, "Error while connecting:", ex);
                closeConnection();
                connected = false;
                running = false;
            }

            // Info the main thread about connection.
            Message msg = new Message();
            msg.what = MSG_CONNECTED;
            boolean test = TopContainerActivity.instance.connectionMessageHandler
                    .sendMessage(msg);

            try
            {
                while(this.running)
                {
                    sleep(10000);
                }
            }
            catch(InterruptedException iex)
            {
                connected = false;
                Log.i(LOG_NAME, "Thread interrupted.");
            }

            // Close the connection
            closeConnection();
            running = false;
        }

        /**
         * This method can be called from any sub thread to notify this thread about
         * unexpected errors, so this thread can notify the main thread about lost connec-
tion.
         */
        private synchronized void alertDisconnect()
        {
            // Only one message will be sent to the main thread, even if this
            // method is called  multiple times.
            if(!disconnectMessageSent)
            {
                Message msg = new Message();
```

```
            msg.what = MSG_DISCONNECTED;
            TopContainerActivity.instance.connectionMessageHandler.sendMessage(msg);
            this.disconnectMessageSent = true;
        }
    }


    /**
     * Messages can be sent to the target device via this method. If the connection
     * is not open, all data will be lost.
     * @param aByte - 8 bit value to send to the target device.
     */
    public synchronized void sendData(Integer aByte)
    {
        if(connected)
        {
            try
            {
                this.outputQueue.add(aByte);
            }
            catch(IllegalStateException ex)
            {
                Log.i(LOG_NAME, "Output queue full!");
            }

        }
        else
        {
            Log.w(LOG_NAME, "Trying to send data, but not connected! Data dropped.");
        }
    }

    /**
     * Get running state of this thread.
     * @return
     */
    public boolean isRunning()
    {
        return running;
    }


    /**
     * This method interrupts this thread and all sub threads.
     */
    public synchronized void stopThread()
    {
        try
        {
            this.inputHandler.interrupt();
        }
        catch(Exception ex)
        {
            Log.w(LOG_NAME, "Unable to interrupt input handler.", ex);
        }
        try
        {
            this.outputHandler.interrupt();
        }
        catch(Exception ex)
        {
            Log.w(LOG_NAME, "Unable to interrupt output handler.", ex);
        }
        try
        {
            this.interrupt();
```

```
    }
    catch(Exception ex)
    {
        Log.w(LOG_NAME, "Unable to interrupt "+LOG_NAME, ex);
    }
}

/**
 * Opens the connection.
 * <br />
 * @throws Exception if the connection can not be established.
 */
protected void openConnection() throws Exception
{
    try
    {
        this.socket.connect();
    }
    catch(IOException ex)
    {
        //Try to close the connection
        try
        {
            this.socket.close();
            Log.e(LOG_NAME, "Unable to open connection. " +
                    "Connection is now closed.", ex);
        }
        catch(IOException ex1)
        {
            Log.e(LOG_NAME, "Unable to open and close connection. " +
                    "Maybe already closed?", ex1);
        }
        throw new Exception("Unable to open the connection!");
    }
}

/**
 * Closes the Bluetooth connection.
 *
 * @throws Exception
 */
protected void closeConnection()
{
     try
     {
         this.socket.close();
     }
     catch(Exception ex)
     {
         Log.i(LOG_NAME, "Unable to close connection. Maybe already closed?");
     }

     try
     {
         alertDisconnect();
     }
     catch(Exception ex)
     {
         Log.e(LOG_NAME, "Critical error: ", ex);
     }

}

/**
 * This thread writes the outgoing data to the output stream.
 * @author Ilkka Saarelainen
```

```
 *
 */
private class OutputHandler extends Thread
{
    private final String LOG_NAME = "OutputHandler";
    private final OutputStream outputStream;
    private boolean running = false;

    /**
     * Constructor.<br/>
     * Obtains the output stream.
     *
     * @throws Exception if the output stream cannot be obtained.
     */
    public OutputHandler() throws Exception
    {
        OutputStream temp = null;
        // Try to obtain the input stream.
        try
        {
            temp = BluetoothConnectionHandler.this.socket.getOutputStream();
        }
        catch(Exception ex)
        {
            throw new Exception("Output stream cannot be obtained.");
        }

        this.outputStream = temp;
    }

    @Override
    public void run()
    {
        this.running = true;
        int byteOut;

        while(this.running)
        {
            try
            {
                byteOut = BluetoothConnectionHandler.this.outputQueue.take();
                this.outputStream.write(byteOut);
                this.outputStream.flush();
            }
            catch(InterruptedException iex)
            {
                // Exit the thread.
                Log.i(LOG_NAME, "Output Stream interrupted.");
                this.running = false;
            }
            catch(Exception ex)
            {
                // Exit the thread.
                Log.i(LOG_NAME, "Unknown error while sending data.");
                // Notify connection handler about the lost connection.
                alertDisconnect();
                this.running = false;
            }
        }
        this.running = false;
    }
}

/**
 * This thread reads the input stream and puts the data to the input queue.
 * @author Ilkka Saarelainen
```

```
 *
 */
private class InputHandler extends Thread
{
    private final String LOG_NAME = "InputHandler";
    private final InputStream inputStream;
    private boolean running = false;

    /**
     * Constructor.
     * Obtains the input stream.
     *
     * @throws Exception if the input stream cannot be obtained.
     */
    public InputHandler() throws Exception
    {
        InputStream temp = null;
        // Try to obtain the input stream.
        try
        {
            temp = BluetoothConnectionHandler.this.socket.getInputStream();
        }
        catch(Exception ex)
        {
            throw new Exception("Input stream cannot be obtained.");
        }

        this.inputStream = temp;
    }

    @Override
    public void run()
    {
        this.running = true;
        int byteIn;

        while(this.running)
        {
            try
            {
                // Wait for the next byte. Read received bytes.
                byteIn = this.inputStream.read();
                // Put byte to the input queue.
                BluetoothConnectionHandler.this.inputQueue.add(byteIn);
            }
            catch(InterruptedIOException iex)
            {
                // Exit thread.
                // See also the next catch block for another interrupt detection.
                Log.i(LOG_NAME, "Input Stream interrupted.");
                this.running = false;
            }
            catch(IOException ioex)
            {
                // When interrupted, the read() -method will cause IOException.
                // It will be detected here. When interrupted,
                // there's no need to call alertDisconnected() method.
                Log.i(LOG_NAME, "Input Stream interrupted1.");
                this.running = false;
                break;
            }
            catch(Exception ex)
            {
                // Exit thread.
                Log.i(LOG_NAME, "Unknown error while receiving data.", ex);
                // Notify connection handler about the lost connection.
```

```
                    alertDisconnect();
                    this.running = false;
                }
            }
        }
    }
}

/**
 * File: FlashUpdater.java
 */
package com.ile.moodlight;

import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

import android.os.AsyncTask;
import android.widget.TextView;

import com.ile.moodlight.activity.TopContainerActivity;

/**
 * This class handles the flash update process.
 *
 * @author Ilkka Saarelainen
 */
public class FlashUpdater extends AsyncTask<Void, String, Void>
{
    private int pagesize   = 64;     // Device specific page size.
    private int flashSize  = 32768;  // The size of the flash in bytes.
    private final String fileName;
    private String result;
    private final TextView statusView;

    private LinkedBlockingQueue<Integer> inputQueue;

    public FlashUpdater(String aFileName, TextView aStatusView)
    {
        this.inputQueue = TopContainerActivity.connectionHandler.inputQueue;
        this.fileName = aFileName;
        this.statusView = aStatusView;
    }

    @Override
    protected void onPreExecute()
    {
        super.onPreExecute();
        this.statusView.setText("Flash upload starting...");
    }

    @Override
    protected Void doInBackground(Void... params)
    {
        this.inputQueue.clear();
        try
        {
            this.startUpload();

            if(this.result == null)
            {
                this.result = "Upload OK! Please remember to switch " +
                        "the device back to the normal mode!";
            }
        }
        catch ( Exception e )
        {
```

```
            // Set the default error message.
            if(this.result == null)
            {
                this.result = "Unknown error while uploading the flash!";
            }

            e.printStackTrace();
        }

        return null;
    }

    @Override
    protected void onProgressUpdate(String... values)
    {
        this.statusView.setText(values[0]);
        super.onProgressUpdate(values);
    }

    /**
     * This method is executed automatically in the main thread after
     * the doInBackground method is finished.
     */
    @Override
    protected void onPostExecute(Void param)
    {
        this.statusView.setText(this.result);
    }

    private void startUpload() throws Exception
    {

        // Reboot the device
        this.sendByte(TopContainerActivity.Commands.REBOOT.getValue());
        // Wait the device to reboot.
        Thread.sleep(1000);

        // Get programmer id.
        String id = readProgrammerID();
        if(!id.contains("AVRBOOT"))
        {
            this.result = "Error: Set the device to the bootloader mode!";
            throw new Exception("Not in bootloader mode!");
        }

        this.publishProgress("Reading flash file...");
        HEXFile hex = new HEXFile(flashSize);

        this.publishProgress("Erasing flash memory...");
        hex.readFile(this.fileName);
        chipErase();

        this.publishProgress("Uploading the new flash...");
        this.writeFlash(hex);

        // Exit bootloader.
        this.exitBootloader();

    }

    /**
     * Sends a given byte to the target device. This is just a shortcut to the
     * sendData()-method of the connection handler.
     *
     * @param aByte The byte to be sent.
     * @throws InterruptedException
```

```
     */
    private void sendByte(int aByte)
    {
        TopContainerActivity.connectionHandler.sendData(aByte);
    }

    /**
     * This method is ported to Java from C-model of "Atmel AVR911:
     * AVR Open-source Programmer
     * for tinyAVR and megaAVR devices", from file "AVRBootloader.cpp", method
     * "writeFlash()".
     * <br /><br />
     * All functionality of the original file is not implemented.<br />
     * <br />
     * The original C-class can be downloaded from following URL, under
     * "Application Notes" section.
     * http://www.atmel.com/products/microcontrollers/avr/megaAVR.aspx?tab=documents
     *
     * @param aData - The HEX-file to upload.
     *
     * @author Ilkka Saarelainen
     */
    private void writeFlash(HEXFile aData)
    {
        // The target bootloader supports block mode, so only block mode is
        // implemented in this programmer.

        int start;
        int end;
        int address;
        int byteCount;
        int blockSize;

        start = aData.getRangeStart();
        end   = aData.getRangeEnd();

        // Get the block size
        this.sendByte(98); // char b
        try
        {
            int in = this.inputQueue.poll(10, TimeUnit.SECONDS);
            if(in != 89) // 89=Y
            {
                throw new Exception("Target doesn't support block mode!");
            }
            // Next 2 bytes should contain the block size.
            blockSize = this.inputQueue.poll(10, TimeUnit.SECONDS) << 8;
            blockSize |= this.inputQueue.poll(10, TimeUnit.SECONDS);

            address = start;

            // Need to write one odd byte first?
            if((address & 1) == 1)
            {
                setAddress(address >> 1); // Flash operations use word addresses.

                // Use only high byte.
                writeFlashLowByte(0xFF);
                writeFlashHighByte(aData.getData(address));

                address++;

                // Need to write page?
                if((address % pagesize) == 0 ||
                        address > end) // Just passed page limit or no more bytes to
write?
```

```
    {
        setAddress((address-2) >> 1); // Set to an address inside the page.
        writeFlashPage();
        setAddress(address >> 1);
    }
}

// Need to write from middle to end of block first?
if((address % blockSize) > 0)  // In the middle of a block?
{
    byteCount = blockSize - (address % blockSize); // Bytes left in block.

    if( (address+byteCount - 1) > end ) // Is that past the write range?
    {
        byteCount = end-address+1; // Bytes left in write range.
        byteCount &= ~0x01;        // Adjust to word count.
    }

    if(byteCount > 0)
    {
        setAddress(address >> 1); // Flash operations use word addresses.

        // Start flash block write
        this.sendByte(66); // 66=B
        this.sendByte((byteCount >> 8) & 0xFF); // Size, MSB first
        this.sendByte(byteCount & 0xFF); //LSB
        this.sendByte(70); // 70=F

        while(byteCount > 0)
        {
            this.sendByte(aData.getData(address));
            address++;
            byteCount--;
        }

        this.checkReturnCR("Write flash block failed. E=1.");
        System.out.print("X");
    }
}

//More complete blocks to write?
while((end - address + 1) >= blockSize)
{
    byteCount = blockSize;

    setAddress(address >> 1); // Flash operations use word addresses.

    // Start flash block write
    this.sendByte(66); // 66=B
    this.sendByte((byteCount >> 8) & 0xFF); // Size, MSB first
    this.sendByte(byteCount & 0xFF); //LSB
    this.sendByte(70); // 70=F

    while(byteCount > 0)
    {
        this.sendByte(aData.getData(address));
        address++;
        byteCount--;
    }
    this.checkReturnCR("Write flash block failed. E=2.");
    System.out.print("X");
}

//Any bytes left in last block?
if((end - address + 1) >= 1)
{
```

```
                byteCount = (end - address + 1); // Get bytes left to write.
                if((byteCount & 1 ) == 1)
                {
                    byteCount++;// Align to next word boundary.
                }

                setAddress(address >> 1); // Flash operations use word addresses.

                // Start flash block write
                this.sendByte(66); // 66=B
                this.sendByte((byteCount >> 8) & 0xFF); // Size, MSB first
                this.sendByte(byteCount & 0xFF); //LSB
                this.sendByte(70); // 70=F

                while(byteCount > 0)
                {
                    if(address > end)
                    {
                        this.sendByte(0xFF); // 66=B
                    }
                    else
                    {
                        this.sendByte(aData.getData(address));
                    }

                    address++;
                    byteCount--;
                }

                this.checkReturnCR("Write flash block failed. E=3.");
                System.out.print("X");
            }
            //Line feed.
            System.out.println("");
            System.out.println("Finished flashing!");

        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
    }

    /**
     * Leave the boot loader.
     * @throws Exception
     */
    private void exitBootloader() throws Exception
    {
        this.sendByte(69); // 69=E

        this.checkReturnCR("Exit bootloader failed.");
    }

    /**
     * Executes a flash page write command on the AVR.
     * @throws Exception if the program does not send "\r\n" response.
     */
    private void writeFlashPage() throws Exception
    {
        this.sendByte(109); // 109=m
        this.checkReturnCR("Write flash page failed.");
    }

    /**
     * Writes a lower byte part of the value.
```

```
     * @param value
     * @throws Exception if the program does not send "\r\n" response.
     */
    private void writeFlashLowByte(int value) throws Exception
    {
        this.sendByte(99); // 99=c
        this.sendByte(value);

        this.checkReturnCR("Write low byte failed.");
    }

    /**
     * Writes a higher byte part of the value.
     * @param value
     * @throws Exception if the program does not send "\r\n" response.
     */
    private void writeFlashHighByte(int value) throws Exception
    {
        this.sendByte(67); // 67=C
        this.sendByte(value);

        this.checkReturnCR("Write high byte failed.");
    }

    /**
     * Set the address pointer.
     * @param aAddress The address.
     * @throws Exception if the program does not send "\r\n" response.
     */
    private void setAddress(int aAddress) throws Exception
    {
        // All addresses are < 0x10000, so always use the 'A' command. 'H' is newer
used.
        this.sendByte(65); // 65=A
        this.sendByte((aAddress >> 8) & 0xFF); //high bits
        this.sendByte(aAddress & 0xFF);        //low bits

        this.checkReturnCR("Set address failed.");
    }

    /**
     * Private method to read linefeed response from the target device.
     * @param aErrorMsg - Error message for logging.
     * @throws Exception if the program does not send "\r\n" response.
     */
    private void checkReturnCR(String aErrorMsg) throws Exception
    {
        // Wait for \r
        Integer in = this.inputQueue.poll(10, TimeUnit.SECONDS);
        if(in == null)
        {
            throw new Exception(aErrorMsg + " Programmer didn't send \\r!");
        }
        if(in != 13)
        {
            throw new Exception(aErrorMsg + " Programmer didn't send \\r!");
        }
    }

    /**
     * Erases the device.
     */
    private void chipErase() throws Exception
    {
        // Clear input.
        this.inputQueue.clear();
```

```
        this.sendByte(101); //Send char e.
        checkReturnCR("Error while executing erase command:");
    }

    /**
     * Gets the ID of the bootloader.
     * @return - The bootloader ID.
     * @throws Exception if device does not respond or string length is invalid.
     */
    private String readProgrammerID()
    {
        String output = "";
        try
        {
            // Clear input.
            this.inputQueue.clear();

            this.sendByte(83); //Send char S.
            StringBuilder s = new StringBuilder();

            // Read 7 bytes from the receive buffer.
            for(int i = 0; i < 7; i ++)
            {
                Integer in = this.inputQueue.poll(5, TimeUnit.SECONDS);
                if(in == null)
                {
                    throw new Exception("Unsupported programmer.");
                }
                s.append(Character.toString((char)(int)in));
            }
            return s.toString();
        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
        return output;
    }
}
/**
 * File: HEXFile.java
 */
package com.ile.moodlight;

import java.io.BufferedReader;
import java.io.FileReader;

import android.util.Log;

/**
 * This class is ported to Java from C-model of "Atmel AVR911: AVR Open-source Program-
mer
 * for tinyAVR and megaAVR devices", from file "HEXParser.cpp".
 * All functionality of the original file is not implemented.
 *
 * Original C-class can be downloaded from following URL, under "Application Notes" sec-
tion.
 * http://www.atmel.com/products/microcontrollers/avr/megaAVR.aspx?tab=documents
 *
 * @author Ilkka Saarelainen
 */
public class HEXFile
{
    private static final String LOG_NAME = "HEXFile";
    private class HEXRecord
```

```
    {
        int length; // Length of the record in bytes.
        int offset;
        int type;
        int[] data;
    }

    private int[] data;
    private int   size;
    private int   start;
    private int   end;

    public HEXFile(int aBufferSize)
    {
        this.data = new int[aBufferSize];
        this.size = aBufferSize;
    }

    public void readFile(String aFileName) throws Exception
    {
        FileReader fr;
        fr = new FileReader(aFileName);
        BufferedReader br = new BufferedReader(fr);
        //Reader reader = new BufferedReader();

        String hexLine; // Contains one line at a time.
        HEXRecord rec;  // Temporary hex record.

        int baseAddress;
        int dataPos;

        baseAddress = 0;
        dataPos     = 0;
        this.start  = size;
        this.end    = 0;

        // Read all lines.
        while((hexLine = br.readLine()) != null)
        {
            rec = this.parseRecord(hexLine);

            switch(rec.type)
            {
                case 0x00: // Data record.
                    // Copy data.
                    if(baseAddress + rec.offset + rec.length > size)
                    {
                        throw new Exception("HEX file defines data outside "
                                + "buffer limits!");
                    }
                    for(dataPos = 0; dataPos < rec.length; dataPos++)
                    {
                        this.data[baseAddress + rec.offset + dataPos] =
rec.data[dataPos];
                    }

                    if(baseAddress + rec.offset < start)
                    {
                        this.start = baseAddress + rec.offset;
                    }
                    if(baseAddress + rec.offset + rec.length -1 > this.end)
                    {
                        this.end = baseAddress + rec.offset + rec.length -1;
                    }
                    break;
                case 0x01: // End of file.
```

```
                Log.i(LOG_NAME,"\nHEX file read OK!");
                fr.close();
                return;
            default:
                throw new Exception("Unknown HEX record type!");
        }
    }

    fr.close();
    // Throw exception if the file didn't end with the EOF-record.
    throw new Exception("Premature end of file encountered!");
}

public void setUsedRange(int aStart, int aEnd) throws Exception
{
    if(aStart < 0 || aEnd >= size || aStart > aEnd)
    {
        throw new Exception("Invalid range!");
    }
    this.start = aStart;
    this.end   = aEnd;
}

public int getRangeStart()
{
    return this.start;
}

public int getRangeEnd()
{
    return this.end;
}

public int getData(int aAddress) throws Exception
{
    if(aAddress < 0 || aAddress >= this.size)
    {
        throw new Exception("Address outside valid range!");
    }
    return this.data[aAddress];
}

private HEXRecord parseRecord(String aHEXLine) throws Exception
{
    HEXRecord result = new HEXRecord();
    // Check line size. Minimum lenght is 11 chars.
    if(aHEXLine.length() < 11)
    {
        throw new Exception("Invalid HEX file format.");
    }
    // Check the first char at the line.
    if(aHEXLine.charAt(0) != ':')
    {
        throw new Exception("Invalid HEX file format.");
    }

    // Parse lenght, offset and type
    result.length = Integer.parseInt(aHEXLine.substring(1,1+2), 16);
    result.offset = Integer.parseInt(aHEXLine.substring(3,3+4), 16);
    result.type   = Integer.parseInt(aHEXLine.substring(7,7+2), 16);

    // Check if the record is as long as it should be.
    if(aHEXLine.length() < 11 + result.length*2)
    {
        throw new Exception("Invalid HEX file format.");
    }
```

```
        // Calculate checksum
        int checksum;
        checksum = result.length;
        checksum = (checksum + (result.offset >> 8) & 0xFF) & 0xFF;
        checksum = (checksum + result.offset & 0xFF) & 0xFF;
        checksum = (checksum + result.type & 0xFF) & 0xFF;

        // Parse data fields
        if(result.length > 0)
        {

            result.data = new int[result.length];

            int recPos; // Current position inside this line.
            for(recPos = 0; recPos < result.length; recPos++)
            {
                int tmpPos = 9 + recPos * 2;
                result.data[recPos] = Integer.parseInt(
                        aHEXLine.substring(tmpPos, tmpPos+2),16);

                checksum = (checksum + result.data[recPos] & 0xFF) & 0xFF;
            }
        }

        // Verify the cheksum.
        checksum = (checksum + Integer.parseInt(
                aHEXLine.substring(9 + result.length*2,9 +
                        result.length*2+2),16) & 0xFF) & 0xFF;

        if(checksum != 0)
        {
            throw new Exception("Wrong checksum on line:"+aHEXLine);
        }
        return result;

    }
}

/**
 * File: LightProgramFragment.java
 */
package com.ile.moodlight;

import java.io.File;
import java.util.ArrayList;

import yuku.ambilwarna.R;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.ile.moodlight.activity.EditLightProgramActivity;
```

```
import com.ile.moodlight.activity.NewLightProgramDialogActivity;
import com.ile.moodlight.activity.TopContainerActivity;
import com.ile.moodlight.lightprogram.LightProgram;

/**
 * This class implements the functionality of the "Light Program" fragment (tab)
 * in the TopContainerActivity.
 *
 * @author Ilkka Saarelainen
 */
public class LightProgramFragment extends Fragment
{
    /**
     * Tag used in debugging.
     */
    private static final String LOG_NAME = "LightProgramFragment";
    /**
     * Intent extra name.
     */
    public static final String EXTRA_PROGRAM = "program";
    /**
     * Request code used to create a new program.<br />
     * All request codes in this class must start by 0x1*.
     */
    private static final int REQUEST_NEW_PROGRAM = 0x11;

    /**
     * The main View instance of this class.
     */
    private View thisView;

    /**
     * The array adapter used to list the light programs.
     */
    private ArrayAdapter<LightProgram> arrayAdapter;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState)

    {
        // Inflate the layout.
        this.thisView = inflater.inflate(R.layout.fragment_light_program, container,
false);

        // Set the onclick listener for the buttons.
        ButtonOnClickListener listener = new ButtonOnClickListener();
        this.thisView.findViewById(R.id.btnUseProgram).setOnClickListener(listener);
        this.thisView.findViewById(R.id.btnNewProgram).setOnClickListener(listener);
        this.thisView.findViewById(R.id.btnEditProgram).setOnClickListener(listener);

        return this.thisView;
    }

    @Override
    public void onResume()
    {
        // Load light commands
        ArrayList<LightProgram> lightPrograms = this.loadPrograms();

        // Create the array adapter.
        this.arrayAdapter = new ArrayAdapter<LightProgram>(getActivity(),
                android.R.layout.simple_list_item_activated_1, lightPrograms);

        ListView listView = (ListView)this.thisView.findViewById(android.R.id.list);
        listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
```

```
        listView.setAdapter(this.arrayAdapter);

        // Check (select) the first program.
        if (this.arrayAdapter.getCount() > 0)
        {
            listView.setItemChecked(0, true);
        }
        super.onResume();
    }

    /**
     * Loads the list of light programs available.
     * @return - A list of light programs available.
     */
    private ArrayList<LightProgram> loadPrograms()
    {
        String dirPath = LightProgram.LIGHT_PROGRAM_DIR;

        File dir = new File(dirPath);
        File[] files = dir.listFiles();
        ArrayList<LightProgram> lightPrograms = new ArrayList<LightProgram>();

        for(File f : files)
        {
            if(f.getName().endsWith(".lp"))
            {
                LightProgram lpf = new LightProgram(f.getAbsolutePath());
                lightPrograms.add(lpf);
            }
        }
        return lightPrograms;
    }

    public class ButtonOnClickListener implements OnClickListener
    {
        @Override
        public void onClick(View view)
        {
            Intent intent;
            Context context = LightProgramFragment.this.thisView.getContext();

            // Check if the new button is pressed.
            if(view.getId() == R.id.btnNewProgram)
            {
                intent = new Intent(context, NewLightProgramDialogActivity.class);
                startActivityForResult(intent, REQUEST_NEW_PROGRAM);
                return;
            }

            // Get the checked light program.
            ListView lv = (ListView)LightProgramFragment.
                    this.thisView.findViewById(android.R.id.list);

            int selectedId = lv.getCheckedItemPosition();

            // Do nothing if nothing is selected.
            if(selectedId == AdapterView.INVALID_POSITION)
            {
                return;
            }

            LightProgram lp = (LightProgram)lv.getItemAtPosition(selectedId);

            switch(view.getId())
            {
            case R.id.btnEditProgram:
```

```
                intent = new Intent(context, EditLightProgramActivity.class);
                intent.putExtra(EXTRA_PROGRAM, lp);
                context.startActivity(intent);
                break;
            case R.id.btnUseProgram:
                // Create new handler to handle the results of program load.
                LightProgramLoadHandler handler = new LightProgramLoadHandler(lp);
                lp.load(handler);
                break;
            default:
                break;
            }
        }
    }

    /**
     * This method will be executed automatically after the "new program" activity is
closed.
     */
    @Override
    public void onActivityResult(int aRequestCode, int aResultCode, Intent aData)
    {
        if(aRequestCode == REQUEST_NEW_PROGRAM)
        {
            if(aResultCode == Activity.RESULT_OK)
            {
                // Get the command from the intent.
                LightProgram lp = (LightProgram)
                        aData.getSerializableExtra(
                                NewLightProgramDialogActivity.EXTRA_PROGRAM);

                Intent intent;
                intent = new Intent(this.thisView.getContext(),
                        EditLightProgramActivity.class);
                intent.putExtra(EXTRA_PROGRAM, lp);
                this.thisView.getContext().startActivity(intent);
            }
            else
            {
                Log.e(LOG_NAME, "New program add cancelled.");
            }
        }
        else
        {
            super.onActivityResult(aRequestCode, aResultCode, aData);
        }
    }

    /**
     * This class is used handle the result of light program load -operation.
     * If the program is loaded successfully, the program will be sent to the device.
     *
     * @author Ilkka Saarelainen
     */
    private static class LightProgramLoadHandler extends Handler
    {

        LightProgram lightProgram;
        /**
         * Constructor.
         * @param aLightProgram - Program to upload when load is finished.
         */
        public LightProgramLoadHandler(LightProgram aLightProgram)
        {
            this.lightProgram = aLightProgram;
        }
```

```
        @Override
        public void handleMessage(Message msg)
        {
            Toast t;
            switch (msg.what)
            {
            case LightProgram.MSG_LOAD_OK:
                ArrayList<Integer> bytes = this.lightProgram.getProgramAsByteArray();
                for(int i = 0; i < bytes.size(); i ++)
                {
                    TopContainerActivity.connectionHandler.sendData(bytes.get(i));
                }

                t = Toast.makeText(TopContainerActivity.instance,
                        R.string.notify_lp_sent_successfully, Toast.LENGTH_SHORT);
                t.show();

                break;
            case LightProgram.MSG_LOAD_FAIL:
                t = Toast.makeText(TopContainerActivity.instance,
                        R.string.error_while_uploading_lp, Toast.LENGTH_SHORT);
                t.show();
                break;
            default:
                break;
            }
            super.handleMessage(msg);
        }
    }
}

/**
 * File: LightProgramFragment.java
 */
package com.ile.moodlight;

import java.io.File;
import java.io.UnsupportedEncodingException;
import java.text.Normalizer;
import java.util.regex.Pattern;

import yuku.ambilwarna.R;
import android.os.Bundle;
import android.os.Environment;
import android.support.v4.app.Fragment;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.ile.moodlight.activity.TopContainerActivity;

/**
 * This class implements the functionality of the Maintenance fragment (tab)
 * in the TopContainerActivity.
 *
 * @author Ilkka Saarelainen
 */
public class MaintenanceFragment extends Fragment
{
    private View thisView;
```

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState)

    {
        this.thisView = inflater.inflate(R.layout.fragment_maintenance, container,
false);

        UploadFlashClickListener listener = new UploadFlashClickListener();
        // Set the onclick listener to the flash upload -button.
        this.thisView.findViewById(R.id.btnFlashUpload)
            .setOnClickListener(listener);
        this.thisView.findViewById(R.id.btnFlashLoad)
            .setOnClickListener(listener);

        View nameChangeButton = this.thisView.findViewById(R.id.btnChangeName);
        View pinChangeButton = this.thisView.findViewById(R.id.btnChangePin);

        // Set listeners to the pin and name input fields.
        EditText et = (EditText)this.thisView.findViewById(R.id.editTextBluetoothName);
        et.addTextChangedListener(new NameChangedListener(et, nameChangeButton));

        EditText etp = (EditText)this.thisView.findViewById(R.id.editTextBluetoothPin);
        etp.addTextChangedListener(new PinChangedListener(pinChangeButton));

        // Set listeners to the pin and name change buttons.
        nameChangeButton.setOnClickListener(new OnNameChangeButtonClickListener(et));
        pinChangeButton.setOnClickListener(new OnPinChangeButtonClickListener(etp));
        return this.thisView;
    }

    /**
     * This class sends the light program to the light device via a
     * clicked button using this listener.
     */
    public class UploadFlashClickListener implements OnClickListener
    {
        @Override
        public void onClick(View v)
        {
            String file = Environment.getExternalStorageDirectory().getPath()
                    +"/MoodLight/RGBLight.hex";

            switch(v.getId())
            {
            case R.id.btnFlashUpload:

                TextView statusText = (TextView)MaintenanceFragment.this.thisView
                        .findViewById(R.id.textFlashUploadStatus);
                FlashUpdater updater = new FlashUpdater(file, statusText);
                updater.execute();
                break;
            case R.id.btnFlashLoad:
                // Check if the flash file exists.
                File f = new File(file);
                Toast t;
                if(f.exists())
                {
                    // Enable the upload button.
                    MaintenanceFragment.this.thisView
                        .findViewById(R.id.btnFlashUpload).setEnabled(true);
                    t = Toast.makeText(MaintenanceFragment.this.thisView.getContext(),
                            "Flash file loaded successfully!", Toast.LENGTH_LONG);
                    t.show();
                }
```

```
            else
            {
                // Show error message and disable the upload button.
                MaintenanceFragment.this.thisView
                .findViewById(R.id.btnFlashUpload).setEnabled(false);
                t = Toast.makeText(MaintenanceFragment.this.thisView.getContext(),
                        "Unable to load the flash file!", Toast.LENGTH_LONG);
                t.show();
            }
            break;
        }
    }
}

/**
 * This class is used as onClickListener on bluetooth name change button.
 *
 */
private class OnNameChangeButtonClickListener implements OnClickListener
{
    private EditText inputField;
    /**
     * @param aTextEdit - The pin input Edit Text view.
     */
    public OnNameChangeButtonClickListener(View aEditText)
    {
        this.inputField = (EditText) aEditText;

    }
    @Override
    public void onClick(View aView)
    {
        // Get the name
        String name = this.inputField.getText().toString();

        // Get the connection handler for faster access
        BluetoothConnectionHandler ch = TopContainerActivity.connectionHandler;

        byte[] chars;
        try
        {
            chars = name.getBytes("ascii");
        }
        catch (UnsupportedEncodingException e)
        {
            e.printStackTrace();
            return ;
        }

        // Write the command identifier.
        ch.sendData(TopContainerActivity.Commands.BLUETOOTH_CONTROL.getValue());
        ch.sendData(0x02);  // name change command
        // Write the amount of arguments, ie. lenght of the pin.
        ch.sendData(chars.length);

        for(byte b : chars)
        {
            Integer value = (int)b;
            ch.sendData(value);
        }

        // Write escape characters (\r\n = 0x0D 0x0A)
        ch.sendData(0x0D);
        ch.sendData(0x0A);
    }
}
```

```
    /**
     * This class is used as onClickListener on bluetooth pin change button.
     *
     */
    private class OnPinChangeButtonClickListener implements OnClickListener
    {
        private EditText inputField;
        /**
         * @param aTextEdit - The pin input Edit Text view.
         */
        public OnPinChangeButtonClickListener(View aEditText)
        {
            this.inputField = (EditText) aEditText;

        }
        @Override
        public void onClick(View aView)
        {
            // Get the pin
            String pin = this.inputField.getText().toString();

            // Get the connection handler for faster access
                        BluetoothConnectionHandler ch = TopContainerActivi-
ty.connectionHandler;

            byte[] chars;
            try
            {
                chars = pin.getBytes("ascii");
            }
            catch (UnsupportedEncodingException e)
            {
                e.printStackTrace();
                return ;
            }

            // Write the command identifier.
            ch.sendData(TopContainerActivity.Commands.BLUETOOTH_CONTROL.getValue());
            ch.sendData(0x01);  // pin change command
            // Write the amount of arguments, ie. lenght of the pin.
            ch.sendData(chars.length);

            for(byte b : chars)
            {
                Integer value = (int)b;
                ch.sendData(value);
            }

            // Write escape characters (\r\n = 0x0D 0x0A)
            ch.sendData(0x0D);
            ch.sendData(0x0A);
        }
    }

    /**
     * This class is used to detect text changes on the bluetooth pin input field.
     */
    private class PinChangedListener implements TextWatcher
    {
        private View     button;
        /**
         *
         * @param aChangeButton - Button to enable/disable
         */
        public PinChangedListener(View aChangeButton)
```

```
    {
        this.button = aChangeButton;
    }

    @Override
    public void afterTextChanged(Editable es)
    {
        String pin = es.toString();

        //If size is 0, disable the button
        if(pin.length() == 0)
        {
            this.button.setEnabled(false);
        }
        else
        {
            this.button.setEnabled(true);
        }
    }

    @Override
    public void beforeTextChanged(CharSequence s, int start, int count,
            int after)
    {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before,
            int count)
    {
    }
}

/**
 * This class is used to detect text changes on the bluetooth name input field.
 */
private class NameChangedListener implements TextWatcher
{
    private View     button;
    private EditText editText;
    /**
     *
     * @param aChangeButton - Button to enable/disable
     */
    public NameChangedListener(View aEditText, View aChangeButton)
    {
        this.button = aChangeButton;
        this.editText = (EditText)aEditText;
    }

    @Override
    public void afterTextChanged(Editable es)
    {
        String name = es.toString();

        //If size is 0, disable the button
        if(name.length() == 0)
        {
            this.button.setEnabled(false);
        }
        else
        {
            this.button.setEnabled(true);
        }

        String output = "";
```

```
            String normalized = Normalizer.normalize(name, Normalizer.Form.NFKD);
            // This regex is from http://stackoverflow.com/questions/15356716/how-can-i-convert-unicode-string-to-ascii-in-java
            String regex = Pattern.quote("[\\p{InCombiningDiacriticalMarks}\\p{IsLm}\\p{IsSk}]+");
            try
            {
                output = new String(normalized.replaceAll(regex, "").getBytes("ascii"), "ascii");
            }
            catch (UnsupportedEncodingException e)
            {

                e.printStackTrace();
                return;
            }
            // Update text if input and output is not equal
            if(!name.equals(output))
            {
                this.editText.setText(output);
                return;
            }
        }

        @Override
        public void beforeTextChanged(CharSequence s, int start, int count,
                int after)
        {
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before,
                int count)
        {
        }
    }
}

/**
 * File: SequenceAdapter.java
 */
package com.ile.moodlight;

import java.util.List;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import com.ile.moodlight.lightprogram.CommandFade;
import com.ile.moodlight.lightprogram.CommandNotifyColor;
import com.ile.moodlight.lightprogram.CommandSet;
import com.ile.moodlight.lightprogram.CommandWait;
import com.ile.moodlight.lightprogram.CommandWaitColor;
import com.ile.moodlight.lightprogram.LightCommand;

/**
 * An instance of this class is used as an adapter between the array of light
 * program commands of a color sequence, and list view.
 * @author Ilkka Saarelainen
 *
 */
```

```
public class SequenceAdapter extends ArrayAdapter<LightCommand> {

    private int color;
    private List<LightCommand> data;

    public SequenceAdapter(Context aContext, int aResource,
            List<LightCommand> aData, int aColor)
    {
        super(aContext, aResource, aData);

        this.color = aColor;
        this.data = aData;
    }

    @Override
    public View getView(int aPosition, View aConvertView, ViewGroup aViewGroup)
    {
        View view = aConvertView;

        if(view == null)
        {
            LayoutInflater vi;
            vi = LayoutInflater.from(getContext());
            view = vi.inflate(R.layout.list_item_click_remove, null);
        }

        // Set correct color of the drag handle
        ImageView dragHandle = (ImageView) view.findViewById(R.id.drag_handle);
        dragHandle.setBackgroundColor(this.color);

        LightCommand command = this.data.get(aPosition);

        // Set the text.
        if(command != null)
        {
            if(command instanceof CommandFade)
            {
                TextView textView = (TextView) view.findViewById(R.id.textCommand);
                textView.setText(command.getDisplayName());
            }
            else if(command instanceof CommandSet)
            {
                TextView textView = (TextView) view.findViewById(R.id.textCommand);
                textView.setText(command.getDisplayName());
            }
            else if(command instanceof CommandWait)
            {
                TextView textView = (TextView) view.findViewById(R.id.textCommand);
                textView.setText(command.getDisplayName());
            }
            else if(command instanceof CommandWaitColor)
            {
                TextView textView = (TextView) view.findViewById(R.id.textCommand);
                textView.setText(command.getDisplayName());
            }
            else if(command instanceof CommandNotifyColor)
            {
                TextView textView = (TextView) view.findViewById(R.id.textCommand);
                textView.setText(command.getDisplayName());
            }
        }
        return view;
    }
    @Override
    public LightCommand getItem(int index)
    {
```

```
        return this.data.get(index);
    }
    @Override
    public void remove(LightCommand object)
    {
        //super.remove(object);
        this.data.remove(object);
    }

    @Override
    public void insert(LightCommand object, int index)
    {
        //super.insert(object, index);
        this.data.add(index, object);
    }

    @Override
    public int getCount()
    {
        return this.data.size();
    }
}

/**
 * File: StaticColorFragment.java
 */
package com.ile.moodlight;

import java.util.concurrent.TimeUnit;

import yuku.ambilwarna.AmbilWarnaKotak;
import yuku.ambilwarna.R;
import android.graphics.Color;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.ile.moodlight.activity.TopContainerActivity;

/**
 * This class implements the functionality of the static color fragment (tab)
 * in the TopContainerActivity.
 *
 * @author Ilkka Saarelainen
 */
public class StaticColorFragment extends Fragment
{
    View viewHue;
    AmbilWarnaKotak viewSatVal;
    ImageView viewCursor;
    View viewOldColor;
    View viewNewColor;
    ImageView viewTarget;
    ViewGroup viewContainer;
```

```
float[] currentColorHsv = new float[3];
public int currentSeekRed = 0;
public int currentSeekGreen = 0;
public int currentSeekBlue = 0;

private View thisView;

@Override
public View onCreateView(LayoutInflater aInflater, ViewGroup aContainer,
        Bundle savedInstanceState)
{
    this.thisView = aInflater.inflate(R.layout.fragment_static_color,
            aContainer, false);

    int defaultColor = 0xFF000000;

    initColorPickerView(this.thisView, defaultColor);

    //Set seekbar listeners
    SeekBar sb = (SeekBar)this.thisView.findViewById(R.id.seekBarRed);
    sb.setOnSeekBarChangeListener(new SeekBarChangeListener());

    sb = (SeekBar)this.thisView.findViewById(R.id.seekBarGreen);
    sb.setOnSeekBarChangeListener(new SeekBarChangeListener());

    sb = (SeekBar)this.thisView.findViewById(R.id.seekBarBlue);
    sb.setOnSeekBarChangeListener(new SeekBarChangeListener());

    // Set the power button listener
    this.thisView.findViewById(R.id.btnTogglePower).setOnClickListener(
            new OnPowerButtonClickListener());

    // Set the listener for the color correction button
    this.thisView.findViewById(R.id.toggleColorCorrection).setOnClickListener(
            new OnToggleColorCorrectionClickListener());

    return this.thisView;
}

/**
 * Initializes the color picker.
 * This method contains code copied from the Color Picker examples.
 */
private void initColorPickerView(final View view, int aInitialColor)
{
    viewHue = view.findViewById(R.id.ambilwarna_viewHue);
    viewSatVal = (AmbilWarnaKotak) view.findViewById(R.id.ambilwarna_viewSatBri);
    viewCursor = (ImageView) view.findViewById(R.id.ambilwarna_cursor);
    viewTarget = (ImageView) view.findViewById(R.id.ambilwarna_target);
    viewContainer = (ViewGroup) view.findViewById(R.id.ambilwarna_viewContainer);

    viewSatVal.setHue(getHue());

    viewHue.setOnTouchListener(new View.OnTouchListener()
    {
        @Override public boolean onTouch(View v, MotionEvent event)
        {
            if (event.getAction() == MotionEvent.ACTION_MOVE
                    || event.getAction() == MotionEvent.ACTION_DOWN
                    || event.getAction() == MotionEvent.ACTION_UP) {

                float y = event.getY();
                if (y < 0.f) y = 0.f;
                if (y > viewHue.getMeasuredHeight())
                {
                    // to avoid looping from end to start.
```

```
                    y = viewHue.getMeasuredHeight() - 0.001f;
                }
                float hue = 360.f - 360.f / viewHue.getMeasuredHeight() * y;
                if (hue == 360.f) hue = 0.f;
                setHue(hue);

                // update view
                viewSatVal.setHue(getHue());
                moveCursor();

                StaticColorFragment.this.sendColor();

                return true;
            }
            return false;
        }
    });
    viewSatVal.setOnTouchListener(new View.OnTouchListener()
    {
        @Override public boolean onTouch(View v, MotionEvent event)
        {
            if (event.getAction() == MotionEvent.ACTION_MOVE
                    || event.getAction() == MotionEvent.ACTION_DOWN
                    || event.getAction() == MotionEvent.ACTION_UP)
            {

                float x = event.getX(); // touch event are in dp units.
                float y = event.getY();

                if (x < 0.f) x = 0.f;
                if (x > viewSatVal.getMeasuredWidth())
                {
                    x = viewSatVal.getMeasuredWidth();
                }
                if (y < 0.f) y = 0.f;
                if (y > viewSatVal.getMeasuredHeight())
                {
                    y = viewSatVal.getMeasuredHeight();
                }
                setSat(1.f / viewSatVal.getMeasuredWidth() * x);
                setVal(1.f - (1.f / viewSatVal.getMeasuredHeight() * y));

                // update view
                moveTarget();

                StaticColorFragment.this.sendColor();

                return true;
            }
            return false;
        }
    });

    // move cursor & target on first draw
    ViewTreeObserver vto = view.getViewTreeObserver();
    vto.addOnGlobalLayoutListener(new ViewTreeObserver.OnGlobalLayoutListener()
    {
        @Override public void onGlobalLayout()
        {
            moveCursor();
            moveTarget();
            view.getViewTreeObserver().removeGlobalOnLayoutListener(this);
        }
    });
}
```

```
    public void setToggleColorCorrection(boolean aState)
    {
        ToggleButton btn = (ToggleButton) this.thisView.
                findViewById(R.id.toggleColorCorrection);
        btn.setChecked(aState);
    }

    /**
     * Sends all the color components to the light device.
     */
    private void sendColor()
    {
        TopContainerActivity.connectionHandler.
            sendData(TopContainerActivity.Commands.SET_RGB.getValue());
        TopContainerActivity.connectionHandler.sendData(getRedComponent());
        TopContainerActivity.connectionHandler.sendData(getGreenComponent());
        TopContainerActivity.connectionHandler.sendData(getBlueComponent());

        // Update seek bars
        SeekBar redBar = (SeekBar)this.thisView.findViewById(R.id.seekBarRed);
        SeekBar greenBar = (SeekBar)this.thisView.findViewById(R.id.seekBarGreen);
        SeekBar blueBar = (SeekBar)this.thisView.findViewById(R.id.seekBarBlue);

        redBar.setProgress(getRedComponent());
        greenBar.setProgress(getGreenComponent());
        blueBar.setProgress(getBlueComponent());
    }

    /**
     * Get the current value of the red color component.
     * @return
     */
    private int getRedComponent()
    {
        int fullColor = Color.HSVToColor(currentColorHsv);
        return ((fullColor >> 16) & 0xFF);
    }

    /**
     * Get the current value of the red color component.
     * @return
     */
    private int getGreenComponent()
    {
        int fullColor = Color.HSVToColor(currentColorHsv);
        return ((fullColor >> 8) & 0xFF);
    }

    /**
     * Get the current value of the red color component.
     * @return
     */
    private int getBlueComponent()
    {
        int fullColor = Color.HSVToColor(currentColorHsv);
        return (fullColor & 0xFF);
    }

    protected void moveCursor()
    {
        float y = viewHue.getMeasuredHeight() - (getHue() * viewHue.getMeasuredHeight()
/ 360.f);
        if (y == viewHue.getMeasuredHeight()) y = 0.f;
        RelativeLayout.LayoutParams layoutParams = (RelativeLayout.LayoutParams) view-
Cursor.getLayoutParams();
```

```
        layoutParams.leftMargin = (int) (viewHue.getLeft() -
Math.floor(viewCursor.getMeasuredWidth() / 2) - viewContainer.getPaddingLeft());
        layoutParams.topMargin = (int) (viewHue.getTop() + y -
Math.floor(viewCursor.getMeasuredHeight() / 2) - viewContainer.getPaddingTop());
        viewCursor.setLayoutParams(layoutParams);
    }

    protected void moveTarget()
    {
        float x = getSat() * viewSatVal.getMeasuredWidth();
        float y = (1.f - getVal()) * viewSatVal.getMeasuredHeight();
        RelativeLayout.LayoutParams layoutParams = (RelativeLayout.LayoutParams) view-
Target.getLayoutParams();
        layoutParams.leftMargin = (int) (viewSatVal.getLeft() + x -
Math.floor(viewTarget.getMeasuredWidth() / 2) - viewContainer.getPaddingLeft());
        layoutParams.topMargin = (int) (viewSatVal.getTop() + y -
Math.floor(viewTarget.getMeasuredHeight() / 2) - viewContainer.getPaddingTop());
        viewTarget.setLayoutParams(layoutParams);
    }

    private float getHue()
    {
        return currentColorHsv[0];
    }

    private float getSat()
    {
        return currentColorHsv[1];
    }

    private float getVal()
    {
        return currentColorHsv[2];
    }

    private void setHue(float hue)
    {
        currentColorHsv[0] = hue;
    }

    private void setSat(float sat)
    {
        currentColorHsv[1] = sat;
    }

    private void setVal(float val)
    {
        currentColorHsv[2] = val;
    }

    private class OnPowerButtonClickListener implements OnClickListener
    {

        @Override
        public void onClick(View aView)
        {
            TopContainerActivity.connectionHandler.
                sendData(TopContainerActivity.Commands.TOGGLE_POWER.getValue());
        }
    }

    /**
     * Class used to handle the clicking of the color correction button.
     */
    private class OnToggleColorCorrectionClickListener implements OnClickListener
    {
```

```
        @Override
        public void onClick(View aView)
        {
            TopContainerActivity.connectionHandler.
                sendData(TopContainerActivity.Commands.TOGGLE_COLOR_CORR.getValue());
            try
            {
                int state = TopContainerActivity.connectionHandler.inputQueue.poll(1,
TimeUnit.SECONDS);
                String onOff;
                if(state == 0)
                {
                    onOff = "Color correction is now off.";
                }
                else
                {
                    onOff = "Color correction is now on.";
                }
                Toast t = Toast.makeText(StaticColorFragment.this.thisView.getContext(),
                        onOff, Toast.LENGTH_LONG);
                t.show();
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
    }


    private class SeekBarChangeListener implements OnSeekBarChangeListener
    {
        public void onProgressChanged(SeekBar seekBar, int progress,boolean fromUser)
        {
            // Always update current colors
            switch(seekBar.getId())
            {
            case R.id.seekBarRed:
                currentSeekRed = progress;
                break;
            case R.id.seekBarGreen:
                currentSeekGreen = progress;
                break;
            case R.id.seekBarBlue:
                currentSeekBlue = progress;
                break;
            }

            // Do nothing if progress is not updated by user.
            if(!fromUser)
            {
                return;
            }
            try
            {
                boolean fail = false;
                switch(seekBar.getId())
                {
                case R.id.seekBarRed:
                    TopContainerActivity.connectionHandler.
                        sendData(TopContainerActivity.Commands.SET_R.getValue());
                    break;
                case R.id.seekBarGreen:
                    TopContainerActivity.connectionHandler.
                        sendData(TopContainerActivity.Commands.SET_G.getValue());
```

```
                    break;
                case R.id.seekBarBlue:
                    TopContainerActivity.connectionHandler.
                        sendData(TopContainerActivity.Commands.SET_B.getValue());
                    break;
                default:
                    fail = true;
                }
                //If seek bar is valid, send the progress. (should always happen)
                if(!fail)
                {
                    StaticColorFragment.this.setColorPicker(
                                                        currentSeekRed,
                                                        currentSeekGreen,
                                                        currentSeekBlue);
                    TopContainerActivity.connectionHandler.sendData(progress);
                }
            }
            catch(Exception ex)
            {
                Log.e("fistSend","Error on sending data", ex);
            }
        }

        public void onStartTrackingTouch(SeekBar seekBar)
        {
        }

        public void onStopTrackingTouch(SeekBar seekBar)
        {
        }
    }

    public void setColorPicker(int aRed, int aGreen, int aBlue)
    {
        // Update the color picker.
        float[] hsv = new float[3];

        Color.RGBToHSV(aRed, aGreen, aBlue, hsv);
        currentColorHsv[0] = hsv[0];
        currentColorHsv[1] = hsv[1];
        currentColorHsv[2] = hsv[2];
        moveTarget();
        moveCursor();
        viewSatVal.setHue(getHue());
    }

    public void setSeekbars(int aRed, int aGreen, int aBlue)
    {
        // Update seek bars
        SeekBar redBar = (SeekBar)this.thisView.findViewById(R.id.seekBarRed);
        SeekBar greenBar = (SeekBar)this.thisView.findViewById(R.id.seekBarGreen);
        SeekBar blueBar = (SeekBar)this.thisView.findViewById(R.id.seekBarBlue);

        redBar.setProgress(aRed);
        greenBar.setProgress(aGreen);
        blueBar.setProgress(aBlue);

        this.currentSeekRed = aRed;
        this.currentSeekGreen = aGreen;
        this.currentSeekBlue = aBlue;
    }
}
```

```
/**
 * File: EditLightProgramActivity.java
 */
package com.ile.moodlight.activity;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Toast;

import com.ile.moodlight.LightProgramFragment;
import com.ile.moodlight.R;
import com.ile.moodlight.lightprogram.Color;
import com.ile.moodlight.lightprogram.LightProgram;

/**
 * This activity is the top level activity of editing a light program.
 * Activity contains for example buttons to edit light sequences and to
 * delete the light program. Light program can be also renamed via this
 * activity.
 *
 * @author Ilkka Saarelainen
 *
 */
public class EditLightProgramActivity extends Activity
{
    public static LightProgram currentLightProgram;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_light_program);
        getActionBar().setDisplayHomeAsUpEnabled(true);

        // Set listener to all the buttons.
        ButtonListener listener = new ButtonListener();
        findViewById(R.id.btnRedSeq).setOnClickListener(listener);
        findViewById(R.id.btnGreenSeq).setOnClickListener(listener);
        findViewById(R.id.btnBlueSeq).setOnClickListener(listener);
        findViewById(R.id.btnSaveLightProg).setOnClickListener(listener);
        findViewById(R.id.btnDeleteLightProg).setOnClickListener(listener);

        // Get the light program from the intent;
        LightProgram lp = (LightProgram)
                getIntent().getSerializableExtra(LightProgramFragment.EXTRA_PROGRAM);

        // Load the program.
        if(!lp.load(null))
        {
            Toast t = Toast.makeText(this, "Unable to load light program!",
Toast.LENGTH_LONG);
            t.show();
            this.finish();
        }
        EditLightProgramActivity.currentLightProgram = lp;

        // Set the program name
        EditText et = (EditText) findViewById(R.id.editTextLightProgramName);
        et.setText(lp.toString());
```

```
}

private class ButtonListener implements OnClickListener
{
    @Override
    public void onClick(View view)
    {
        Intent intent;
        Context context = EditLightProgramActivity.this;
        intent = new Intent(context, EditLightSequenceActivity.class);
        switch(view.getId())
        {
        case R.id.btnRedSeq:
            // Set the sequence color.
            intent.putExtra(EditLightSequenceActivity.EXTRA_SEQUENCE_NAME,
                    Color.ColorEnum.RED);
            // Start activity.
            context.startActivity(intent);
            break;
        case R.id.btnGreenSeq:
            // Set the sequence color
            intent.putExtra(EditLightSequenceActivity.EXTRA_SEQUENCE_NAME,
                    Color.ColorEnum.GREEN);
            // Start activity.
            context.startActivity(intent);
            break;
        case R.id.btnBlueSeq:
            // Set the sequence color
            intent.putExtra(EditLightSequenceActivity.EXTRA_SEQUENCE_NAME,
                    Color.ColorEnum.BLUE);
            // Start activity.
            context.startActivity(intent);
            break;
        case R.id.btnSaveLightProg:
            EditText et = (EditText)findViewById(R.id.editTextLightProgramName);
            String name = et.getText().toString();
            try
            {
                // Update the light program.
                EditLightProgramActivity.currentLightProgram =
                EditLightProgramActivity.currentLightProgram
                        .saveAs(name, true);

                String msg = getResources().getString(
                        R.string.toast_notice_program_save_ok);

                Toast t = Toast.makeText(EditLightProgramActivity.this,
                        msg, Toast.LENGTH_LONG);
                t.show();
            }
            catch (LightProgram.FileAlreadyFoundException ex)
            {
                String errorMsg = getResources().getString(
                        R.string.toast_error_program_name_already_in_use);
                Toast t = Toast.makeText(EditLightProgramActivity.this,
                        errorMsg, Toast.LENGTH_LONG);
                t.show();
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
            break;
        case R.id.btnDeleteLightProg:
            EditLightProgramActivity.currentLightProgram.delete();
            finish();
```

```
                break;
            default:
                break;
            }
        }
    }
}

/**
 * File: EditLightSequence.java
 */
package com.ile.moodlight.activity;

import java.util.ArrayList;

import android.app.Activity;
import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;

import com.ile.moodlight.R;
import com.ile.moodlight.SequenceAdapter;
import com.ile.moodlight.lightprogram.Color;
import com.ile.moodlight.lightprogram.LightCommand;
import com.ile.moodlight.lightprogram.LightProgram;
import com.mobeta.android.dslv.DragSortListView;

/**
 * This activity contains a list of light program commands for selected
 * color component sequence. Commands can be added, deleted or modified via
 * this activity.
 *
 * @author Ilkka Saarelainen
 *
 */
public class EditLightSequenceActivity extends ListActivity
{

    private static String LOG_NAME = "EditLightSequence";
    private static final int REQUEST_NEW_COMMAND = 1;
    private static final int REQUEST_EDIT_COMMAND = 2;
    public static final String EXTRA_SEQUENCE_NAME = "seq_name";
    public static final String EXTRA_EDIT_COMMAND = "edit_command";
    public static final String EXTRA_POSITION = "pos";
    private Color.ColorEnum colorSeqName;

    private SequenceAdapter adapter;

    private ArrayList<LightCommand> lightSequence;

    private DragSortListView.DropListener onDrop =
        new DragSortListView.DropListener()
        {
            @Override
            public void drop(int from, int to)
            {
                Log.i("drop", "from " + from + " to "+to);
                LightCommand item = adapter.getItem(from);
```

```
                adapter.remove(item);
                adapter.insert(item, to);
                adapter.notifyDataSetChanged();
            }
        };

    private DragSortListView.RemoveListener onRemove =
        new DragSortListView.RemoveListener()
        {
            @Override
            public void remove(int which)
            {
                Log.i("remove", "id " + which);
                adapter.remove(adapter.getItem(which));
                adapter.notifyDataSetChanged();
            }
        };

    private DragSortListView.DragScrollProfile ssProfile =
        new DragSortListView.DragScrollProfile()
    {
            @Override
            public float getSpeed(float w, long t)
            {
                if(w > 0.8f)
                {
                    return ((float) adapter.getCount()) / 0.001f;
                }
                else
                {
                    return 10.0f * w;
                }
            }
        };

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_edit_light_sequence);
        getActionBar().setDisplayHomeAsUpEnabled(true);

        DragSortListView lv = (DragSortListView) getListView();

        lv.setDropListener(onDrop);
        lv.setRemoveListener(onRemove);
        lv.setDragScrollProfile(ssProfile);

        OnLongClickListener longClicklistener = new OnLongClickListener();
        lv.setLongClickable(true);
        lv.setOnItemLongClickListener(longClicklistener);

        Intent startIntent = getIntent();
        this.colorSeqName = (Col-
or.ColorEnum)startIntent.getSerializableExtra(EXTRA_SEQUENCE_NAME);
        int dragHandleColor = 0;

        // Get copy of the current light sequence.
        switch(this.colorSeqName)
        {
        case RED:
            this.lightSequence =
                new ArrayList<LightCommand>(
                        EditLightProgramActivity.currentLightProgram.redSequence);
            dragHandleColor = 0xFFFF0000;
            break;
```

```
        case GREEN:
            this.lightSequence = new ArrayList<LightCommand>(
                    EditLightProgramActivity.currentLightProgram.greenSequence);
            dragHandleColor = 0xFF00FF00;
            break;
        case BLUE:
            this.lightSequence = new ArrayList<LightCommand>(
                    EditLightProgramActivity.currentLightProgram.blueSequence);
            dragHandleColor = 0xFF0000FF;
            break;
        }

        this.adapter = new SequenceAdapter(this, R.layout.list_item_click_remove,
                this.lightSequence, dragHandleColor);
        setListAdapter(this.adapter);

        // Set click listener to the buttons.
        ClickListener listener = new ClickListener();
        findViewById(R.id.btnAddNewCommand).setOnClickListener(listener);
        findViewById(R.id.btnSaveSequence).setOnClickListener(listener);
    }

    /**
     * This will be executed automatically after the "new command" activity is closed.
     */
    @Override
    protected void onActivityResult(int aRequestCode, int aResultCode, Intent aData)
    {
        switch(aRequestCode)
        {
        case REQUEST_NEW_COMMAND:
            if(aResultCode == Activity.RESULT_OK)
            {
                // Get the command from the intent.
                LightCommand command = (LightCommand)
                        aData.getSerializableExtra(NewCommandActivity.EXTRA_COMMAND);

                this.adapter.add(command);
                this.adapter.notifyDataSetChanged();
            }
            else
            {
                Log.e(LOG_NAME, "New command add cancelled.");
            }
            break;
        case REQUEST_EDIT_COMMAND:
            if(aResultCode == Activity.RESULT_OK)
            {
                LightCommand command = (LightCommand)
                        aData.getSerializableExtra(NewCommandActivity.EXTRA_COMMAND);
                int pos = aData.getIntExtra(EXTRA_POSITION, 0);

                LightCommand commandToRemove = this.adapter.getItem(pos);

                this.adapter.remove(commandToRemove);
                this.adapter.insert(command, pos);
                this.adapter.notifyDataSetChanged();
            }
            else
            {
                Log.e(LOG_NAME, "Edit command cancelled.");
            }
            break;
        default:
            super.onActivityResult(aRequestCode, aResultCode, aData);
        }
```

```
}

private void newCommand()
{
    Intent intent = new Intent(this, NewCommandActivity.class);
    intent.putExtra(EXTRA_SEQUENCE_NAME, this.colorSeqName);
    startActivityForResult(intent, REQUEST_NEW_COMMAND);
}

// Click listeners for save and add new command -buttons.
private class ClickListener implements OnClickListener
{
    @Override
    public void onClick(View aView)
    {
        switch(aView.getId())
        {
        case R.id.btnSaveSequence:

            // The following is just to keep the code clean
            LightProgram clp= EditLightProgramActivity.currentLightProgram;
            EditLightSequenceActivity inst = EditLightSequenceActivity.this;

            // Update the current sequence.
            switch(inst.colorSeqName)
            {
            case RED:
                clp.redSequence = inst.lightSequence;
                break;
            case GREEN:
                clp.greenSequence = inst.lightSequence;
                break;
            case BLUE:
                clp.blueSequence = inst.lightSequence;
                break;
            }

            finish();
            break;
        case R.id.btnAddNewCommand:
            EditLightSequenceActivity.this.newCommand();
            break;
        default:
            Log.e(LOG_NAME,"Unsupported button on listener.");

        }
    }
}

/**
 * This listener is used to open the edit command window on long click.
 * @author Ilkka Saarelainen
 */
private class OnLongClickListener implements OnItemLongClickListener
{

    @Override
    public boolean onItemLongClick(AdapterView<?> arg0, View arg1,
            int pos, long id)
    {
        // Get the selected command.
        LightCommand command = (LightCommand) arg0.getItemAtPosition(pos);

        Intent intent = new Intent(EditLightSequenceActivity.this,
                NewCommandActivity.class);
```

```
            intent.putExtra(EXTRA_SEQUENCE_NAME, EditLightSequenceActivi-
ty.this.colorSeqName);
            intent.putExtra(EXTRA_EDIT_COMMAND, command);
            intent.putExtra(EXTRA_POSITION, pos);

            startActivityForResult(intent, REQUEST_EDIT_COMMAND);

            return false;
        }

    }
}

/**
 * File: NewCommandActivity.java
 */

package com.ile.moodlight.activity;

import java.util.AbstractMap.SimpleEntry;
import java.util.ArrayList;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Spinner;

import com.ile.moodlight.R;
import com.ile.moodlight.lightprogram.Color;
import com.ile.moodlight.lightprogram.CommandFade;
import com.ile.moodlight.lightprogram.CommandNotifyColor;
import com.ile.moodlight.lightprogram.CommandSet;
import com.ile.moodlight.lightprogram.CommandWait;
import com.ile.moodlight.lightprogram.CommandWaitColor;
import com.ile.moodlight.lightprogram.LightCommand;

/**
 * This activity appears as a dialog over the EditLightSequence activity.
 * This activity is used to add new commands to a sequence and to edit existing
 * commands.
 *
 * @author Ilkka Saarelainen
 *
 */
public class NewCommandActivity extends Activity
{
    private static final String LOG_NAME = "NewCommandActivity";

    // SelectColor1 and 2 are the color names and values used in
    // "notify" and "wait color" -commands.
    private SimpleEntry<Color.ColorBytesEnum, String> selectColor1;
    private SimpleEntry<Color.ColorBytesEnum, String> selectColor2;
    private CheckBox c1;
    private CheckBox c2;

    private boolean isNewCommand;
    private int position;
```

```
    ArrayAdapter<LightCommand> adapter;
    LightCommand selectedCommand;

    public static String EXTRA_COMMAND = "command";
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_command);

        // Set click listener to the OK and calcel -buttons.
        ClickListener listener = new ClickListener();
        findViewById(R.id.btnNewCommandOK).setOnClickListener(listener);
        findViewById(R.id.btnNewCommandCancel).setOnClickListener(listener);

        initializeSpinner();

        // Get the current color.
        Intent startIntent = getIntent();
        Color.ColorEnum col = (Color.ColorEnum)startIntent.
                getSerializableExtra(EditLightSequenceActivity.EXTRA_SEQUENCE_NAME);

        // Select the correct color names and values to the check box.
        switch(col)
        {
        case RED:
            this.selectColor1 = newColorNameEntry(Color.ColorBytesEnum.GREEN);
            this.selectColor2 = newColorNameEntry(Color.ColorBytesEnum.BLUE);
            break;
        case GREEN:
            this.selectColor1 = newColorNameEntry(Color.ColorBytesEnum.RED);
            this.selectColor2 = newColorNameEntry(Color.ColorBytesEnum.BLUE);
            break;
        case BLUE:
            this.selectColor1 = newColorNameEntry(Color.ColorBytesEnum.RED);
            this.selectColor2 = newColorNameEntry(Color.ColorBytesEnum.GREEN);
            break;
        }

        // Set correct names for the check boxes.
        this.c1 = (CheckBox) findViewById(R.id.checkBoxWaitColor1);
        this.c2 = (CheckBox) findViewById(R.id.checkBoxWaitColor2);

        this.c1.setText(this.selectColor1.getValue());
        this.c2.setText(this.selectColor2.getValue());

        // Check if there should be added a new command or edited an old one.
        LightCommand lc = (LightCommand) getIntent().getSerializableExtra(
                EditLightSequenceActivity.EXTRA_EDIT_COMMAND);
        if(lc != null)
        {
            // Save the position in the list view of the parent activity.
            this.position = getIntent().getIntExtra(
                    EditLightSequenceActivity.EXTRA_POSITION, 0);
            this.isNewCommand = false;

            Spinner spinner = (Spinner)findViewById(R.id.spinner_command_types);

            spinner.setSelection(this.matchCommandType(lc));
            // Set form values.
            if(lc.getClass().equals(CommandFade.class))
            {
                CommandFade tmp = (CommandFade)lc;
                ((Edit-
Text)findViewById(R.id.editTextTargetValue)).setText(""+tmp.getTargetValue());
```

```
                ((Edit-
Text)findViewById(R.id.editTextStepSize)).setText(""+tmp.getStepSize());
            }
            else if(lc.getClass().equals(CommandSet.class))
            {
                CommandSet tmp = (CommandSet)lc;
                ((Edit-
Text)findViewById(R.id.editTextTargetValue)).setText(""+tmp.getTargetValue());
            }
            else if(lc.getClass().equals(CommandWait.class))
            {
                CommandWait tmp = (CommandWait)lc;
                ((Edit-
Text)findViewById(R.id.editTextTargetValue)).setText(""+tmp.getTicks());
            }
            else if(lc.getClass().equals(CommandNotifyColor.class))
            {
                CommandNotifyColor tmp = (CommandNotifyColor)lc;

                // Select correct check boxes.
                if((this.selectColor1.getKey().getValue() & tmp.getIntFromBooleans()) !=
0)
                {
                    this.c1.setChecked(true);
                }
                if((this.selectColor2.getKey().getValue() & tmp.getIntFromBooleans()) !=
0)
                {
                    this.c2.setChecked(true);
                }
            }
            else if(lc.getClass().equals(CommandWaitColor.class))
            {
                CommandWaitColor tmp = (CommandWaitColor)lc;

                // Select correct check boxes.
                if((this.selectColor1.getKey().getValue() & tmp.getIntFromBooleans()) !=
0)
                {
                    this.c1.setChecked(true);
                }
                if((this.selectColor2.getKey().getValue() & tmp.getIntFromBooleans()) !=
0)
                {
                    this.c2.setChecked(true);
                }
            }

            // Change the title of the dialog.
            this.setTitle(R.string.title_activity_edit_command);
        }
        else
        {
            this.isNewCommand = true;
        }
    }

    /**
     * This method returns the id of the command type in the command type selection
list.
     * @param aCommand
     * @return
     */
    private int matchCommandType(LightCommand aCommand)
    {
        Spinner spinner = (Spinner)findViewById(R.id.spinner_command_types);
```

```
            for(int i = 0; i < spinner.getCount(); i++)
            {
                LightCommand listCommand = (LightCommand) spinner.getItemAtPosition(i);
                if(listCommand.getClass().equals(aCommand.getClass()))
                {
                    return i;
                }
            }
            return -1;

        }

        /**
         * This method is used to create a new key-value pair containing a byte value
         * of a color component as a key, and the corresponding human readable name of the
         * color component (defined in the strings.xml).<br />
         * <br />
         * This method is just to make the "onCreate" method simpler.<br />
         * <br />
         * @param aColorByteValue - a byte value of the color.
         * @return
         */
        private SimpleEntry<Color.ColorBytesEnum, String> newColorNameEntry(
                Color.ColorBytesEnum aColorByteValue)
        {
            String name = "";
            switch(aColorByteValue)
            {
            case BLUE:
                name = this.getResources().getString(R.string.blue_name);
                break;
            case GREEN:
                name = this.getResources().getString(R.string.green_name);
                break;
            case RED:
                name = this.getResources().getString(R.string.red_name);
                break;
            default:
                break;
            }
            return new SimpleEntry<Color.ColorBytesEnum, String>(aColorByteValue, name);
        }

        /**
         * Initializes the spinner (drop down menu) containing the command types.
         */
        private void initializeSpinner()
        {
            // List of one of each command type.
            ArrayList<LightCommand> commandList = new ArrayList<LightCommand>();
            commandList.add(new CommandFade());
            commandList.add(new CommandSet());
            commandList.add(new CommandWait());
            commandList.add(new CommandWaitColor());
            commandList.add(new CommandNotifyColor());

            this.adapter = new ArrayAdapter<LightCommand>(this,
                    android.R.layout.simple_spinner_item, commandList);

    this.adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

            Spinner spinner = (Spinner)findViewById(R.id.spinner_command_types);
            spinner.setAdapter(this.adapter);

            // Set listener.
```

```
    ItemSelectedListener listener = new ItemSelectedListener();
    spinner.setOnItemSelectedListener(listener);
}

/**
 * Listener class for the command spinner.
 */
private class ItemSelectedListener implements OnItemSelectedListener
{
    /**
     * This method is triggered when a command is selected in the spinner.
     */
    @Override
    public void onItemSelected(AdapterView<?> aParent, View view, int aPosition,
            long aID)
    {
        LightCommand selected = (LightCommand)aParent.getItemAtPosition(aPosition);

        // Set the currently selected command.
        NewCommandActivity.this.selectedCommand = selected;

        switch(selected.getCommandID())
        {
        case CommandFade.COMMAND_ID:
            setVisibleTargetValueField(true);
            setVisibleStepSize(true);
            setVisibleWaitColorCheckBoxes(false);
            break;

        case CommandSet.COMMAND_ID:
            setVisibleTargetValueField(true);
            setVisibleStepSize(false);
            setVisibleWaitColorCheckBoxes(false);
            break;

        case CommandWait.COMMAND_ID:
            setVisibleTargetValueField(true);
            setVisibleStepSize(false);
            setVisibleWaitColorCheckBoxes(false);
            break;

        case CommandWaitColor.COMMAND_ID:
            setVisibleTargetValueField(false);
            setVisibleStepSize(false);
            setVisibleWaitColorCheckBoxes(true);
            break;

        case CommandNotifyColor.COMMAND_ID:
            setVisibleTargetValueField(false);
            setVisibleStepSize(false);
            setVisibleWaitColorCheckBoxes(true);
            break;

        default:
            Log.e(LOG_NAME, "Unsupported command!");
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0)
    {

    }
    private void setVisibleTargetValueField(boolean aVisible)
    {
        if (aVisible)
```

```
        {
            findViewById(R.id.textTargetValue).setVisibility(View.VISIBLE);
            findViewById(R.id.editTextTargetValue).setVisibility(View.VISIBLE);
        }
        else
        {
            findViewById(R.id.textTargetValue).setVisibility(View.GONE);
            findViewById(R.id.editTextTargetValue).setVisibility(View.GONE);
        }
    }

    private void setVisibleStepSize(boolean aVisible)
    {
        if (aVisible)
        {
            findViewById(R.id.textStepSize).setVisibility(View.VISIBLE);
            findViewById(R.id.editTextStepSize).setVisibility(View.VISIBLE);
        }
        else
        {
            findViewById(R.id.textStepSize).setVisibility(View.GONE);
            findViewById(R.id.editTextStepSize).setVisibility(View.GONE);
        }
    }

    private void setVisibleWaitColorCheckBoxes(boolean aVisible)
    {
        if (aVisible)
        {
            findViewById(R.id.checkBoxWaitColor1).setVisibility(View.VISIBLE);
            findViewById(R.id.checkBoxWaitColor2).setVisibility(View.VISIBLE);
        }
        else
        {
            findViewById(R.id.checkBoxWaitColor1).setVisibility(View.GONE);
            findViewById(R.id.checkBoxWaitColor2).setVisibility(View.GONE);
        }
    }

}



/**
 * OnClick listener class for the OK-button.
 * @author Ilkka Saarelainen
 */
private class ClickListener implements OnClickListener
{

    @Override
    public void onClick(View aView)
    {
        int target, step;

        switch(aView.getId())
        {
        case R.id.btnNewCommandOK:
            switch(NewCommandActivity.this.selectedCommand.getCommandID())
            {
            case CommandFade.COMMAND_ID:
                // Get values.
                target = getValueFromEditText(R.id.editTextTargetValue, 0, 255);
                step = getValueFromEditText(R.id.editTextStepSize, -127, 127);
                NewCommandActivity.this.selectedCommand =
                   new CommandFade(target, step);
```

```
                break;
            case CommandSet.COMMAND_ID:
                target = getValueFromEditText(R.id.editTextTargetValue, 0, 255);
                NewCommandActivity.this.selectedCommand = new CommandSet(target);
                break;
            case CommandWait.COMMAND_ID:
                int time = getValueFromEditText(R.id.editTextTargetValue,
                             0, 0xFFFF);
                NewCommandActivity.this.selectedCommand = new CommandWait(time);
                break;
            case CommandWaitColor.COMMAND_ID:
                // Get the color byte values from check boxes.
                int colorsToWait = 0;
                colorsToWait |= (c1.isChecked() ? selectColor1
                                  .getKey().getValue() : 0);
                colorsToWait |= (c2.isChecked() ? selectColor2
                                  .getKey().getValue() : 0);
                NewCommandActivity.this.selectedCommand =
                          new CommandWaitColor(colorsToWait);
                break;
            case CommandNotifyColor.COMMAND_ID:
                // Get the color byte values from check boxes.
                int colorsToNotify = 0;
                colorsToNotify |= (c1.isChecked() ? selectColor1
                                    .getKey().getValue() : 0);
                colorsToNotify |= (c2.isChecked() ? selectColor2
                                    .getKey().getValue() : 0);
                NewCommandActivity.this.selectedCommand =
                       new CommandNotifyColor(colorsToNotify);
                break;

            default:
                Log.e(LOG_NAME, "Unsupported command!");
            }

            Intent intent = new Intent();
            intent.putExtra(NewCommandActivity.EXTRA_COMMAND,
                    NewCommandActivity.this.selectedCommand);

            /* If this activity edited an existing command, add the position
             * of the command in the list to the intent.*/
            if(!NewCommandActivity.this.isNewCommand)
            {
                intent.putExtra(EditLightSequenceActivity.EXTRA_POSITION,
                        NewCommandActivity.this.position);
            }

            setResult(RESULT_OK, intent);
            finish();
            break;
        case R.id.btnNewCommandCancel:
            setResult(RESULT_CANCELED);
            finish();
        }
    }

    private int getValueFromEditText(int aResourceID, int aClampMin, int aClampMax)
    {
        String t;
        int res = 0;
        t = ((EditText)findViewById(aResourceID)).getText().toString();
        if(t.isEmpty())
        {

            return aClampMin;
        }
```

```
            try
            {
                res = Integer.parseInt(t);
                return clamp(res, aClampMin, aClampMax);
            }
            catch(Exception ex)
            {
                Log.e(LOG_NAME, "Invalid number:"+t);
            }
            return aClampMin;
        }
        /**
         * Clamp value to given range.
         * @param aValue
         * @param aMin
         * @param aMax
         * @return
         */
        private int clamp(int aValue, int aMin, int aMax)
        {
            if(aValue <= aMin)
            {
                return aMin;
            }
            if(aValue >= aMax)
            {
                return aMax;
            }
            return aValue;
        }
    }
}

/**
 * File: NewLightProgramDialogActivity.java
 */

package com.ile.moodlight.activity;

import java.io.IOException;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.Toast;

import com.ile.moodlight.R;
import com.ile.moodlight.lightprogram.LightProgram;

/**
 * This activity appears as a dialog over the LightProgramFragment. This is used
 * to query a name for new light program.
 *
 * @author Ilkka Saarelainen
 *
 */
public class NewLightProgramDialogActivity extends Activity
{
    private static String LOG_NAME = "NewLightProgramDialogActivity";
    public static final String EXTRA_PROGRAM = "program";
```

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_new_light_program_dialog);

    // Set the onclick listener
    findViewById(R.id.btnNewProgramOK).setOnClickListener(
            new ClickListener());

}

/**
 * OnClick listener class for the OK-button.
 * @author Ilkka Saarelainen
 */
private class ClickListener implements OnClickListener
{

    @Override
    public void onClick(View v)
    {
        Toast t;

        // Get the text from the edit text box.
        EditText et = (EditText) findViewById(R.id.editTextNewProgramName);
        String name = et.getText().toString();

        // Check if given name is already found.
        LightProgram lp = new LightProgram(LightProgram.LIGHT_PROGRAM_DIR +
                "/" + name + LightProgram.FILE_EXTENSION);
        if(lp.exists())
        {
            t = Toast.makeText(NewLightProgramDialogActivity.this,
                    getResources().getString(
                        R.string.toast_error_program_name_already_in_use),
                    Toast.LENGTH_LONG);
            t.show();
            return;
        }

        // Try to create the new file.
        try
        {
            lp.initNewProgram();
        }
        catch (IOException e)
        {
            t = Toast.makeText(NewLightProgramDialogActivity.this,
                    getResources().getString(R.string.error_while_creating_new_lp),
                    Toast.LENGTH_LONG);
            t.show();
            Log.e(LOG_NAME, "Error while creating the new light program", e);

            setResult(RESULT_CANCELED);
            finish();
        }

        Intent intent = new Intent();
        intent.putExtra(NewLightProgramDialogActivity.EXTRA_PROGRAM, lp);
        setResult(RESULT_OK, intent);
        finish();
    }

}
```

```
}

/**
 * File: SelectDeviceActivity.java
 */

/*
 * Copyright (C) 2009 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * ---------------------------------------------------------------------------
 *
 * This file is copied from Android API demo "Bluetooth chat". A few
 * modifications have been made to fit the needs of this program.
 * The original file name is "DeviceListActivity.java".
 * Modified by Ilkka Saarelainen on 5.4.2013.
 *
 * Original file can be downloaded from following URL:
 * http://developer.android.com/tools/samples/index.html
 */

package com.ile.moodlight.activity;

import java.util.Set;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

import com.ile.moodlight.R;

/**
 * This Activity appears as a dialog. It lists any paired devices and
 * devices detected in the area after discovery. When a device is chosen
 * by the user, the MAC address of the device is sent back to the parent
 * Activity in the result Intent.
 *
 * @author Ilkka Saarelainen
 */
```

```
public class SelectDeviceActivity extends Activity
{
    // Debugging
    private static final String TAG = "DeviceListActivity";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Setup the window
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.activity_select_device);

        // Set result CANCELED in case the user backs out
        setResult(Activity.RESULT_CANCELED);

        // Initialize the button to perform device discovery
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                doDiscovery();
                v.setVisibility(View.GONE);
            }
        });

        // Initialize array adapters. One for already paired devices and
        // one for newly discovered devices
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);

        // Find and set up the ListView for paired devices
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);

        // Find and set up the ListView for newly discovered devices
        ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);

        // Register for broadcasts when a device is discovered
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);

        // Register for broadcasts when discovery has finished
        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(mReceiver, filter);

        // Get the local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();

        // Get a set of currently paired devices
        Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

        // If there are paired devices, add each one to the ArrayAdapter
```

```
        if (pairedDevices.size() > 0) {
            findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
                device.getAddress());
            }
        } else {
            String noDevices = getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        // Make sure we're not doing discovery anymore
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
        }

        // Unregister broadcast listeners
        this.unregisterReceiver(mReceiver);
    }

    /**
     * Start device discover with the BluetoothAdapter
     */
    private void doDiscovery() {
        if (D) Log.d(TAG, "doDiscovery()");

        // Indicate scanning in the title
        setProgressBarIndeterminateVisibility(true);
        setTitle(R.string.scanning);

        // Turn on sub-title for new devices
        findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

        // If we're already discovering, stop it
        if (mBtAdapter.isDiscovering()) {
            mBtAdapter.cancelDiscovery();
        }

        // Request discover from BluetoothAdapter
        mBtAdapter.startDiscovery();
    }

    // The on-click listener for all devices in the ListViews
    private OnItemClickListener mDeviceClickListener = new OnItemClickListener() {
        public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {
            // Cancel discovery because it's costly and we're about to connect
            mBtAdapter.cancelDiscovery();

            // Get the device MAC address, which is the last 17 chars in the View
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Create the result Intent and include the MAC address
            Intent intent = new Intent();
            intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

            // Set result and finish this Activity
            setResult(Activity.RESULT_OK, intent);
            finish();
        }
    };
```

```
    // The BroadcastReceiver that listens for discovered devices and
    // changes the title when discovery is finished
    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();

            // When discovery finds a device
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Get the BluetoothDevice object from the Intent
                BluetoothDevice device = in-
tent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
                // If it's already paired, skip it, because it's been listed already
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    mNewDevicesArrayAdapter.add(device.getName() + "\n" + de-
vice.getAddress());
                }
            // When discovery is finished, change the Activity title
            } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
                setProgressBarIndeterminateVisibility(false);
                setTitle(R.string.select_device);
                if (mNewDevicesArrayAdapter.getCount() == 0) {
                    String noDevices =
getResources().getText(R.string.none_found).toString();
                    mNewDevicesArrayAdapter.add(noDevices);
                }
            }
        }
    };

}

/**
 * File: TopContainerActivity.java
 */
package com.ile.moodlight.activity;

import java.io.File;
import java.util.HashMap;
import java.util.concurrent.TimeUnit;

import android.app.ActionBar;
import android.app.FragmentTransaction;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.ile.moodlight.BluetoothConnectionHandler;
import com.ile.moodlight.LightProgramFragment;
import com.ile.moodlight.MaintenanceFragment;
import com.ile.moodlight.R;
import com.ile.moodlight.StaticColorFragment;

/**
 * This is the top level of activities in this program. This activity contains
 * the Bluetooth connection handler as a static member. BTConnection handler
```

```
 * is closed when this activity is closing.
 *
 * This does not contain any directly visible elements. There are only the Fragment-
 * Layout having 3 fragments (= tabs). FragmentLayout contains all the visible
 * part of this activity.
 *
 * @author Ilkka Saarelainen
 *
 */
public class TopContainerActivity extends FragmentActivity
                                   implements ActionBar.TabListener {

    private static final String STATE_SELECTED_NAVIGATION_ITEM =
            "selected_navigation_item";

    private HashMap<Integer, Fragment> fragments;

    public static BluetoothConnectionHandler connectionHandler;
    private final String LOG_NAME = "TopContainerActivity";
    //public static LightProgram currentLightProgram;
    public static TopContainerActivity instance;
    public Handler connectionMessageHandler;

    private boolean isClosing = false;

    // Request codes
    private final int REQUEST_TARGET_DEVICE = 0x01;

    //Commands
    public enum Commands
    {
        SET_R               (0xC1),
        SET_G               (0xC2),
        SET_B               (0xC3),
        SET_RGB             (0xC4),
        UPLOAD_PROG         (0xC6),
        TOGGLE_POWER        (0xCF),
        BLUETOOTH_CONTROL   (0xCE),
        REBOOT              (0xC7),
        GET_RGB             (0xC9),
        GET_COLOR_CORR      (0xCC),
        TOGGLE_COLOR_CORR   (0xCD);

        private int value;
        Commands(int aValue)
        {
            this.value = aValue;
        }
        public int getValue()
        {
            return this.value;
        }
    }

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Check if the required folders are created.
        File file = new File(Environment.getExternalStorageDirectory().getPath()
                +"/MoodLight/LightPrograms");
        if(!file.exists())
        {
            try
            {
```

```
                file.mkdirs();
            }
            catch(Exception ex)
            {
                Log.e(LOG_NAME, "Error while creating directories:",ex);
            }
        }

        setContentView(R.layout.activity_top_container);

        this.openSelectDeviceActivity();

        TopContainerActivity.instance = this;

        // Set up the action bar.
        final ActionBar actionBar = getActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

        // Create map for tab fragments.
        this.fragments = new HashMap<Integer, Fragment>();
        // Create instantiate all fragments.
        Fragment staticColorFragment = new StaticColorFragment();
        Fragment lightProgramFragment = new LightProgramFragment();
        Fragment maintenanceFragment = new MaintenanceFragment();
        this.fragments.put(0, staticColorFragment);
        this.fragments.put(1, lightProgramFragment);
        this.fragments.put(2, maintenanceFragment);

        // Add all fragments to the action bar.
        actionBar.addTab(actionBar.newTab().setText(
                R.string.title_fragment_static_color).setTabListener(this));
        actionBar.addTab(actionBar.newTab().setText(
                R.string.title_fragment_light_program).setTabListener(this));
        actionBar.addTab(actionBar.newTab().setText(
                R.string.title_fragment_maintenance).setTabListener(this));

        // Set the message handler to get messages from another threads.
        this.connectionMessageHandler = new ConnectionMessageHandler();

        this.isClosing = false;
    }


    /**
     * This will be executed automatically after the target device activity is closed.
     */
    @Override
    protected void onActivityResult(int aRequestCode, int aResultCode, Intent aData)
    {
        if(aRequestCode == REQUEST_TARGET_DEVICE)
        {
            if(aResultCode == SelectDeviceActivity.RESULT_OK)
            {
                String targetAddress = aData.
                        getStringExtra(SelectDeviceActivity.EXTRA_DEVICE_ADDRESS);
                TopContainerActivity.instance.connectToDevice(targetAddress);
            }
            else
            {
                Toast t = Toast.makeText(this,
                        "This program can not be used without selecting a target de-
vice!",
                        Toast.LENGTH_LONG);
                t.show();
                Log.e(LOG_NAME, "User didn't select any device");
                TopContainerActivity.instance.finish();
```

```
        }
    }
    else
    {
        super.onActivityResult(aRequestCode, aResultCode, aData);
    }
}

/**
 * Starts the activity to select target device.
 */
private void openSelectDeviceActivity()
{
    Intent intent = new Intent(this, SelectDeviceActivity.class);
    startActivityForResult(intent, REQUEST_TARGET_DEVICE);
}

private void connectToDevice(String aAddress)
{
    // Start connection thread if it's not running.
    boolean start = true;
    if(connectionHandler != null)
    {
        if(connectionHandler.isRunning())
        {
            start = false;
        }
    }
    if(start)
    {
        // Get the adapter.
        BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
        BluetoothDevice device = adapter.getRemoteDevice(aAddress);

        try
        {
            connectionHandler = new BluetoothConnectionHandler(device);
            connectionHandler.start();
        }
        catch(Exception ex)
        {
            Log.e(LOG_NAME, ex.getMessage(), ex);
        }
    }
}

@Override
public void onRestoreInstanceState(Bundle savedInstanceState)
{
    if (savedInstanceState.containsKey(STATE_SELECTED_NAVIGATION_ITEM))
    {
        getActionBar().setSelectedNavigationItem(
                savedInstanceState.getInt(STATE_SELECTED_NAVIGATION_ITEM));
    }
}

@Override
public void onSaveInstanceState(Bundle outState)
{
    outState.putInt(STATE_SELECTED_NAVIGATION_ITEM,
            getActionBar().getSelectedNavigationIndex());
}


@Override
public void onDestroy()
```

```
    {
        super.onDestroy();
        if(TopContainerActivity.connectionHandler != null)
        {
            this.isClosing = true;
            TopContainerActivity.connectionHandler.stopThread();
        }
    }

    @Override
    public void onTabUnselected(ActionBar.Tab tab, FragmentTransaction fragmentTransac-
tion) {
    }

    @Override
    public void onTabSelected(ActionBar.Tab aTab, FragmentTransaction aFragmentTransac-
tion)
    {
        int tabId = aTab.getPosition();
        getSupportFragmentManager().beginTransaction()
                .replace(R.id.container, this.fragments.get(tabId))
                .commit();
    }

    @Override
    public void onTabReselected(ActionBar.Tab tab, FragmentTransaction fragmentTransac-
tion)
    {
    }

    public class ConnectionMessageHandler extends Handler
    {
        @Override
        public void handleMessage(Message msg)
        {
            // If the program is being closed, don't ask to reconnect.
            if(isClosing)
            {
                return;
            }

            switch (msg.what)
            {
            case BluetoothConnectionHandler.MSG_DISCONNECTED:
                Toast t = Toast.makeText(TopContainerActivity.instance,
                        "Unable to connect the device!", Toast.LENGTH_LONG);
                t.show();

                // Close and kill previous bluetooth connection
                TopContainerActivity.connectionHandler.stopThread();
                TopContainerActivity.instance.openSelectDeviceActivity();

                break;
            case BluetoothConnectionHandler.MSG_CONNECTED:
                Log.i("color correction:","get state");
                // Get the color correction status and current color values from the de-
vice.
                TopContainerActivity.connectionHandler.
                    sendData(TopContainerActivity.Commands.GET_COLOR_CORR.getValue());
                try
                {
                    int state = TopContainerActivity.
                            connectionHandler.inputQueue.poll(3, TimeUnit.SECONDS);
                    boolean checked = false;
                    Log.i("color correction:",""+state);
                    if(state == 1)
```

```
                    {
                        checked = true;
                    }
                    Fragment tab = TopContainerActivity.instance.fragments.get(0);
                    ToggleButton btn = (ToggleButton)
tab.getView().findViewById(R.id.toggleColorCorrection);
                    btn.setChecked(checked);

                    // Get RGB values
                    TopContainerActivity.connectionHandler.
                        sendData(TopContainerActivity.Commands.GET_RGB.getValue());
                    int red = TopContainerActivity.
                            connectionHandler.inputQueue.poll(3, TimeUnit.SECONDS);
                    int green = TopContainerActivity.
                            connectionHandler.inputQueue.poll(3, TimeUnit.SECONDS);
                    int blue = TopContainerActivity.
                            connectionHandler.inputQueue.poll(3, TimeUnit.SECONDS);

                    Log.i("Current RGB: ", red+" "+green+" "+blue);

                    StaticColorFragment scf = (StaticColorFragment) tab;
                    scf.setColorPicker(red, green, blue);
                    scf.setSeekbars(red, green, blue);
                }
                catch (Exception e)
                {
                    e.printStackTrace();
                }

            default:
                break;
            }
            super.handleMessage(msg);
        }
    }
}
```

```java
/**
 * File: Color.java
 */
package com.ile.moodlight.lightprogram;

import java.io.Serializable;

/**
 * This class contains common color related values.
 * @author Ilkka Saarelainen
 *
 */
public class Color implements Serializable
{
    // Color identifier values. DO NOT EDIT THESE VALUES!
    public enum ColorEnum
    {
        RED (1),
        GREEN (2),
        BLUE (3);

        private int value;
        ColorEnum(int aValue)
        {
            this.value = aValue;
        }
        public int getValue()
        {
            return this.value;
        }
        public void setValue(int aValue)
        {
            this.value = aValue;
        }
    }

    // Color byte masks. DO NOT EDIT THESE VALUES!
    public enum ColorBytesEnum
    {
        RED (1),
        GREEN (2),
        BLUE (4);

        private int value;
        ColorBytesEnum(int aValue)
        {
            this.value = aValue;
        }
        public int getValue()
        {
            return this.value;
        }
        public void setValue(int aValue)
        {
            this.value = aValue;
        }
    }
}

/**
 * File: CommandFade.java
 */
```

```java
package com.ile.moodlight.lightprogram;

/**
 *
 * This class represents the Fade light program command.
 * @author Ilkka Saarelainen
 *
 */
public class CommandFade extends LightCommand
{
    public static final int COMMAND_ID = 0x04;
    private static final String DISPLAY_NAME = "Fade";
    private int targetValue;
    private int stepSize;

    /**
     * Constructor to create empty command.
     */
    public CommandFade()
    {

    }
    /**
     *
     * @param aTargetColor
     * @param aStepSize
     */
    public CommandFade(int aTargetValue, int aStepSize)
    {
        this.stepSize = aStepSize;
        this.targetValue = aTargetValue;
    }

    /**
     * Returns the string representation of this command
     */
    @Override
    public String getDisplayName()
    {
        return "Fade to " + this.targetValue + " by " + this.stepSize +  "steps.";
    }

    @Override
    public int getCommandID()
    {
        return COMMAND_ID;
    }

    public int getTargetValue()
    {
        return targetValue;
    }

    public void setTargetValue(int targetValue)
    {
        this.targetValue = targetValue;
    }

    public int getStepSize()
    {
        return stepSize;
    }

    public void setStepSize(int stepSize)
    {
        this.stepSize = stepSize;
```

```
    }

    public String toString()
    {
        return DISPLAY_NAME;
    }

    @Override
    public Integer[] getBytes()
    {
        Integer[] bytes = new Integer[3];
        bytes[0] = COMMAND_ID;
        bytes[1] = this.stepSize;
        bytes[2] = this.targetValue;
        return bytes;
    }

    @Override
    public String getBytesString()
    {
        Integer[] bytes = this.getBytes();
        String result = LightCommand.formaByteToHEX(bytes[0]);
        result +=       LightCommand.formaByteToHEX(bytes[1]);
        result +=       LightCommand.formaByteToHEX(bytes[2]);
        return result;
    }

}

/**
 * File: CommandNotifyColor.java
 */

package com.ile.moodlight.lightprogram;

/**
 *
 * This class represents the Notify Color light program command.
 * @author Ilkka Saarelainen
 *
 */
public class CommandNotifyColor extends LightCommand
{
    public static final int COMMAND_ID = 0x07;
    private static final String DISPLAY_NAME = "Notify color";
    private boolean notifyRed;
    private boolean notifyGreen;
    private boolean notifyBlue;

    /**
     * Constructor to create empty command.
     */
    public CommandNotifyColor()
    {
        this.notifyRed = false;
        this.notifyGreen = false;
        this.notifyBlue = false;
    }
    /**
     * @param aColor Color to
     * @param aTargetColor
     */
    public CommandNotifyColor(boolean aNotifyRed, boolean aNotifyGreen,
            boolean aNotifyBlue)
    {
```

```
        this.notifyRed = aNotifyRed;
        this.notifyGreen = aNotifyGreen;
        this.notifyBlue = aNotifyBlue;
    }

    public CommandNotifyColor(int aColorsToWait)
    {
        this.setBooleansFromByte(aColorsToWait);
    }

    @Override
    public String getDisplayName()
    {
        //int nOfColorsToWait = this.getNumberOfColorsToWait();
        int nOfColorNamesApplied = 0;
        StringBuilder sb = new StringBuilder();

        if(this.notifyRed)
        {
            sb.append("Red");
            nOfColorNamesApplied++;
        }

        if(this.notifyGreen)
        {
            if(nOfColorNamesApplied == 1)
                sb.append(", ");

            sb.append("Green");
            nOfColorNamesApplied++;
        }

        if(this.notifyBlue)
        {
            if(nOfColorNamesApplied > 0)
                sb.append(", ");

            sb.append("Blue");
            nOfColorNamesApplied++;
        }

        return "Notify color(s): " + sb.toString();
    }

    @Override
    public int getCommandID()
    {
        return COMMAND_ID;
    }

    public boolean getnotifyRed()
    {
        return this.notifyRed;
    }

    public boolean getnotifyGreen()
    {
        return this.notifyGreen;
    }

    public boolean getnotifyBlue()
    {
        return this.notifyBlue;
    }

    public void setnotifyRed()
```

```
{
    this.notifyRed = true;
}

public void setnotifyGreen()
{
    this.notifyGreen = true;
}

public void setnotifyBlue()
{
    this.notifyBlue= true;
}

/**
 * Returns the string representation of this command
 */
public String toString()
{
    return DISPLAY_NAME;
}

@Override
public Integer[] getBytes()
{
    Integer[] bytes = new Integer[2];
    bytes[0] = COMMAND_ID;
    bytes[1] = this.getIntFromBooleans();
    return bytes;
}

@Override
public String getBytesString()
{
    Integer[] bytes = this.getBytes();
    String result = LightCommand.formaByteToHEX(bytes[0]);
    result +=       LightCommand.formaByteToHEX(bytes[1]);
    return result;
}

/**
 * This method converts a byte value containing info of colors to wait, to
 * the boolean values.
 * Example:
 * 0x03 -> notifyRed = true, notifyGreen = true, notifyBlue = false.
 * @return
 */
private void setBooleansFromByte (int aByte)
{
    if((aByte & Color.ColorBytesEnum.RED.getValue()) > 0)
        this.notifyRed = true;

    if((aByte & Color.ColorBytesEnum.GREEN.getValue()) > 0)
        this.notifyGreen = true;

    if((aByte & Color.ColorBytesEnum.BLUE.getValue()) > 0)
        this.notifyBlue = true;
}

/**
 *
 * @return
 */
public int getIntFromBooleans()
{
    int res = 0;
```

```
        if(this.notifyRed)
            res |= Color.ColorBytesEnum.RED.getValue();

        if(this.notifyGreen)
            res |= Color.ColorBytesEnum.GREEN.getValue();

        if(this.notifyBlue)
            res |= Color.ColorBytesEnum.BLUE.getValue();

        return res;
    }

}

/**
 * File: CommandSet.java
 */
package com.ile.moodlight.lightprogram;

/**
 *
 * This class represents the Set Brightness light program command.
 * @author Ilkka Saarelainen
 *
 */
public class CommandSet extends LightCommand
{
    public static final int COMMAND_ID = 0x05;
    private static final String DISPLAY_NAME = "Set brightness";
    private int targetValue;

    /**
     * Constructor to create empty command.
     */
    public CommandSet()
    {

    }
    /**
     * @param aColor Color to
     * @param aTargetColor
     */
    public CommandSet(int aTargetValue)
    {

        this.targetValue = aTargetValue;
    }

    @Override
    public String getDisplayName()
    {
        return "Set brightness to " + this.targetValue;
    }

    @Override
    public int getCommandID()
    {
        return COMMAND_ID;
    }

    public int getTargetValue()
    {
        return targetValue;
    }
```

```
    public void setTargetValue(int targetValue)
    {
        this.targetValue = targetValue;
    }

    /**
     * Returns the string representation of this command
     */
    public String toString()
    {
        return DISPLAY_NAME;
    }

    @Override
    public Integer[] getBytes()
    {
        Integer[] bytes = new Integer[2];
        bytes[0] = COMMAND_ID;
        bytes[1] = this.targetValue;
        return bytes;
    }

    @Override
    public String getBytesString()
    {
        Integer[] bytes = this.getBytes();
        String result = LightCommand.formaByteToHEX(bytes[0]);
        result +=       LightCommand.formaByteToHEX(bytes[1]);
        return result;
    }

}

/**
 * File: CommandWait.java
 */

package com.ile.moodlight.lightprogram;

/**
 *
 * This class represents the Wait light program command.
 * @author Ilkka Saarelainen
 *
 */
public class CommandWait extends LightCommand
{
    public static final int COMMAND_ID = 0x08;
    private static final String DISPLAY_NAME = "Wait";
    private int ticks;

    /**
     * Constructor to create empty command.
     */
    public CommandWait()
    {

    }
    /**
     *
     * @param aTicks
     */
    public CommandWait(int aTicks)
    {
```

```
        this.ticks = aTicks;
    }

    /**
     * Returns the string representation of this command
     */
    @Override
    public String getDisplayName()
    {
        return "Wait " + this.ticks + " ticks.";
    }

    @Override
    public int getCommandID()
    {
        return COMMAND_ID;
    }

    public int getTicks()
    {
        return this.ticks;
    }

    public void setTicks(int aTicks)
    {
        this.ticks = aTicks;
    }

    public String toString()
    {
        return DISPLAY_NAME;
    }

    @Override
    public Integer[] getBytes()
    {
        Integer[] bytes = new Integer[3];
        bytes[0] = COMMAND_ID;
        bytes[1] = (this.ticks >> 8) & 0xFF;
        bytes[2] = this.ticks & 0xFF;
        return bytes;
    }

    @Override
    public String getBytesString()
    {
        Integer[] bytes = this.getBytes();
        String result = LightCommand.formaByteToHEX(bytes[0]);
        result +=       LightCommand.formaByteToHEX(bytes[1]);
        result +=       LightCommand.formaByteToHEX(bytes[2]);
        return result;
    }

}
/**
 * File: CommandWaitColor.java
 */

package com.ile.moodlight.lightprogram;

/**
 *
 * This class represents the Wait Color light program command.
 * @author Ilkka Saarelainen
 *
 */
```

```
public class CommandWaitColor extends LightCommand
{
    public static final int COMMAND_ID = 0x06;
    private static final String DISPLAY_NAME = "Wait color";
    private boolean waitRed;
    private boolean waitGreen;
    private boolean waitBlue;

    /**
     * Constructor to create empty command.
     */
    public CommandWaitColor()
    {
        this.waitRed = false;
        this.waitGreen = false;
        this.waitBlue = false;
    }

    public CommandWaitColor(int aColorsToWait)
    {
        this.setBooleansFromByte(aColorsToWait);
    }

    @Override
    public String getDisplayName()
    {
        //int nOfColorsToWait = this.getNumberOfColorsToWait();
        int nOfColorNamesApplied = 0;
        StringBuilder sb = new StringBuilder();

        if(this.waitRed)
        {
            sb.append("Red");
            nOfColorNamesApplied++;
        }

        if(this.waitGreen)
        {
            if(nOfColorNamesApplied == 1)
                sb.append(", ");

            sb.append("Green");
            nOfColorNamesApplied++;
        }

        if(this.waitBlue)
        {
            if(nOfColorNamesApplied > 0)
                sb.append(", ");

            sb.append("Blue");
            nOfColorNamesApplied++;
        }

        return "Wait color(s): " + sb.toString();
    }

    @Override
    public int getCommandID()
    {
        return COMMAND_ID;
    }

    public boolean getWaitRed()
    {
        return this.waitRed;
```

```
}

public boolean getWaitGreen()
{
    return this.waitGreen;
}

public boolean getWaitBlue()
{
    return this.waitBlue;
}

public void setWaitRed()
{
    this.waitRed = true;
}

public void setWaitGreen()
{
    this.waitGreen = true;
}

public void setWaitBlue()
{
    this.waitBlue= true;
}

/**
 * Returns the string representation of this command
 */
public String toString()
{
    return DISPLAY_NAME;
}

@Override
public Integer[] getBytes()
{
    Integer[] bytes = new Integer[2];
    bytes[0] = COMMAND_ID;
    bytes[1] = this.getIntFromBooleans();
    return bytes;
}

@Override
public String getBytesString()
{
    Integer[] bytes = this.getBytes();
    String result = LightCommand.formaByteToHEX(bytes[0]);
    result +=       LightCommand.formaByteToHEX(bytes[1]);
    return result;
}

/**
 * This method converts a byte value containing info of colors to wait, to
 * the boolean values.
 * Example:
 * 0x03 -> waitRed = true, waitGreen = true, waitBlue = false.
 * @return
 */
private void setBooleansFromByte (int aByte)
{
    if((aByte & Color.ColorBytesEnum.RED.getValue()) > 0)
        this.waitRed = true;

    if((aByte & Color.ColorBytesEnum.GREEN.getValue()) > 0)
```

```
            this.waitGreen = true;

        if((aByte & Color.ColorBytesEnum.BLUE.getValue()) > 0)
            this.waitBlue = true;
    }

    /**
     *
     * @return
     */
    public int getIntFromBooleans()
    {
        int res = 0;

        if(this.waitRed)
            res |= Color.ColorBytesEnum.RED.getValue();

        if(this.waitGreen)
            res |= Color.ColorBytesEnum.GREEN.getValue();

        if(this.waitBlue)
            res |= Color.ColorBytesEnum.BLUE.getValue();

        return res;
    }

}

/**
 * File: LightCommand.java
 */
package com.ile.moodlight.lightprogram;

import java.io.Serializable;

/**
 *
 * This is the abstract base class of all light program commands.
 * @author Ilkka Saarelainen
 *
 */
public abstract class LightCommand implements Serializable
{
    /**
     * Returns the name of this command.
     * @return
     */
    abstract public String getDisplayName();

    abstract public int getCommandID();

    abstract public Integer[] getBytes();

    abstract public String getBytesString();

    abstract public String toString();

    public static final String formaByteToHEX(int aByte)
    {
        String t = String.format("%02x ", aByte);
        return t.substring(t.length() - 3, t.length()).toUpperCase();
    }
}

/**
```

```java
 * File: LightProgram.java
 */

package com.ile.moodlight.lightprogram;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;

import android.os.Environment;
import android.os.Handler;
import android.os.Message;
import android.util.Log;

import com.ile.moodlight.R;
import com.ile.moodlight.activity.TopContainerActivity;

/**
 *
 * This class represents the Light program.
 * @author Ilkka Saarelainen
 *
 */
public class LightProgram extends File
{
    public ArrayList<LightCommand> redSequence;
    public ArrayList<LightCommand> greenSequence;
    public ArrayList<LightCommand> blueSequence;

    public static final Integer RED_SEQ_START = 1;
    public static final Integer GREEN_SEQ_START = 2;
    public static final Integer BLUE_SEQ_START = 3;

    public static final int MSG_LOAD_OK = 1;
    public static final int MSG_LOAD_FAIL = 2;

    public static final String FILE_EXTENSION = ".lp";
    public static final String LIGHT_PROGRAM_DIR =
            Environment.getExternalStorageDirectory().getPath()
            +TopContainerActivity.instance.getResources()
            .getString(R.string.path_lightprogram_dir);

    private boolean loadedFromFile = false;

    public LightProgram(String aPath)
    {
        super(aPath);
    }

    @Override
    public String toString()
    {
        String name = this.getName();
        // Remove .lp -suffix.
        name = name.substring(0, name.length() - 3);
        return name;
    }

    /**
     * Loads a light program from file. </br>
     * </br>
```

```java
 * NOTE: this method uses the Handler mechanism, so it can be called easily
 * from another thread.</br>
 * </br>
 * @param aHandler  - Handler to notify when load is finished.
 * @throws Exception
 */
public boolean load(Handler aHandler)
{
    // Format sequences.
    this.redSequence = new ArrayList<LightCommand>();
    this.greenSequence = new ArrayList<LightCommand>();
    this.blueSequence = new ArrayList<LightCommand>();

    BufferedReader br = null;
    try
    {
         FileReader fr;
         fr = new FileReader(this.getAbsolutePath());
         br = new BufferedReader(fr);

         String sequenceLine;

         // J-valeus comes from LightCommand.Color enum. Do NOT edit.
         // Read all lines.
         for(int j = 1; j <= 3; j++)
         {
             ArrayList<LightCommand> currentSequence;
             // Select correct sequence
             switch(j)
             {
             case 1:
                 currentSequence = this.redSequence;
                 break;
             case 2:
                 currentSequence = this.greenSequence;
                 break;
             case 3:
             default: // Default must be defined. Don't remove this.
                 currentSequence = this.blueSequence;
             }

             sequenceLine = br.readLine();
             // Remove all spaces to get the size of the line.
             sequenceLine = sequenceLine.replaceAll(" ", "");
             int size = sequenceLine.length() / 2;
             int[] byteArray = new int[size];

             // Parse integer values from the line;
             for(int i = 0; i < size; i ++)
             {
                 String hex = sequenceLine.substring(i*2, i*2+2);
                 int value = Integer.parseInt(hex, 16);
                 byteArray[i] = value;
             }

             for(int i = 0; i < byteArray.length; i++)
             {
                 LightCommand command;
                 switch(byteArray[i])
                 {
                 case CommandFade.COMMAND_ID:
                     // Set correct sign to the steps.
                     int steps = byteArray[i+1] << (Integer.SIZE - 8);
                     steps >>= (Integer.SIZE - 8);
                     command = new CommandFade(
                             byteArray[i+2], steps);
```

```
                    // Move i over the next 2 bytes.
                    i += 2;
                    break;
                case CommandSet.COMMAND_ID:
                    command = new CommandSet(byteArray[i+1]);
                    // Move i over the next byte.
                    i += 1;
                    break;
                case CommandWait.COMMAND_ID:
                    int wait = (byteArray[i+1] << 8) | byteArray[i+2];
                    command = new CommandWait(wait);
                    // Move i over the next 2 bytes.
                    i += 2;
                    break;
                case CommandWaitColor.COMMAND_ID:
                    command = new CommandWaitColor(byteArray[i+1]);
                    // Move i over the next byte.
                    i += 1;
                    break;

                case CommandNotifyColor.COMMAND_ID:
                    command = new CommandNotifyColor(byteArray[i+1]);
                    // Move i over the next byte.
                    i += 1;
                    break;
                default:
                    throw new Exception("Unsupported command detected!:"
                            +byteArray[i] +" on line "+j+":"+i);
                }
                //Set command to the sequence.
                currentSequence.add(command);
            }
        }
    }
    catch(Exception ex)
    {
        if(aHandler != null)
        {
            // Notify the handler.
            Message msg = new Message();
            msg.what = MSG_LOAD_FAIL;
            aHandler.sendMessage(msg);
        }
        Log.e("LightProgram", "Error while loading program", ex);
        return false;
    }

    try
    {
        // Close reader.
        br.close();
    }
    catch(Exception ex)
    {
        Log.e("LightProgram", "Error while loading program", ex);
        // Do nothing here.
    }
    if(aHandler != null)
    {
        // Notify the handler.
        Message msg = new Message();
        msg.what = MSG_LOAD_OK;
        aHandler.sendMessage(msg);
    }
    this.loadedFromFile = true;
```

```
        return true;
    }

    public void save() throws Exception
    {
        if(!this.loadedFromFile)
        {
            throw new Exception("Can not use save() -method because the " +
                    "program is not loaded from File!");
        }
        this.saveToFile(this.getAbsolutePath());
    }

    public LightProgram saveAs(String aLightProgramName, boolean aDeleteOld) throws Ex-
ception
    {
        // Check if the file name is changed.
        if(aLightProgramName.equals(this.toString()))
        {
            // If the file name is not change, save() -method can be used.
            this.save();
            return this;
        }

        File f = new File(LIGHT_PROGRAM_DIR + "/" + aLightProgramName + FILE_EXTENSION);

        // Check if the file already exists.
        if(f.exists())
        {
            throw new FileAlreadyFoundException();
        }

        this.saveToFile(LIGHT_PROGRAM_DIR + "/" + aLightProgramName +
                FILE_EXTENSION);

        // Delete the old file.
        if(aDeleteOld)
        {
            try
            {
                this.delete();
            }
            catch (Exception ex)
            {
                ex.printStackTrace();
            }
        }
        LightProgram newInstance = new LightProgram(
                LIGHT_PROGRAM_DIR + "/" + aLightProgramName + FILE_EXTENSION);
        // Load the program from the new file.
        newInstance.load(null);
        return newInstance;
    }

    public void saveToFile(String aFileName) throws Exception
    {
        FileWriter fw;
        fw = new FileWriter(aFileName);
        BufferedWriter br = new BufferedWriter(fw);

        String seq = sequenceToString(this.redSequence);
        br.write(seq + "\r\n");
        seq = sequenceToString(this.greenSequence);
        br.write(seq + "\r\n");
        seq = sequenceToString(this.blueSequence);
        br.write(seq + "\r\n");
```

```
        br.flush();
        br.close();
    }

    private String sequenceToString(ArrayList<LightCommand> aSequence)
    {
        StringBuilder sb = new StringBuilder();
        for(int i = 0; i < aSequence.size(); i++)
        {
            LightCommand c = aSequence.get(i);
            sb.append(c.getBytesString());
        }
        return sb.toString();
    }

    private ArrayList<Integer> sequenceToByteArray(ArrayList<LightCommand> aArray)
    {
        ArrayList<Integer> res = new ArrayList<Integer>();
        for(int i = 0; i < aArray.size(); i++)
        {
            res.addAll(new ArrayList<Integer>(Arrays.asList(aArray.get(i).getBytes())));
        }
        return res;
    }

    /**
     * Initializes a new light program. The file path must be given in the
     * constructor.
     *
     * @throws IOException if the file can not be created.
     */
    public void initNewProgram() throws IOException
    {
        // Create the file.
        this.createNewFile();

        // Write 3 empty lines to the file.
        FileWriter fw;
        fw = new FileWriter(this);
        BufferedWriter br = new BufferedWriter(fw);
        br.write("\r\n\r\n\r\n");
        br.close();

    }

    /**
     * Returns the whole program as a byte array, which can be sent to the light unit.
     * @return
     */
    public ArrayList<Integer> getProgramAsByteArray()
    {

        ArrayList<Integer> bytes = new ArrayList<Integer>();
        // Add all bytes
        bytes.add(RED_SEQ_START);
        bytes.addAll(this.sequenceToByteArray(this.redSequence));
        bytes.add(GREEN_SEQ_START);
        bytes.addAll(this.sequenceToByteArray(this.greenSequence));
        bytes.add(BLUE_SEQ_START);
        bytes.addAll(this.sequenceToByteArray(this.blueSequence));

        //Add start command and program size.
        int size = bytes.size();
        bytes.add(0, TopContainerActivity.Commands.UPLOAD_PROG.getValue());
        bytes.add(1, (size >> 8) & 0xFF); // High byte
        bytes.add(2, size & 0xFF); // Low byte
```

```
        return bytes;
    }

    /**
     * Sends this program to the target device.
     */
    public void sendProgram()
    {
        ArrayList<Integer> bytes = this.getProgramAsByteArray();

        for(int i = 0; i < bytes.size(); i ++)
        {
            TopContainerActivity.connectionHandler.sendData(bytes.get(i));
        }
    }

    /**
     * This type of exception could be thrown for example when trying to create
     * a LightProgram with name that already exists.
     *
     * @author Ilkka Saarelainen
     *
     */
    public class FileAlreadyFoundException extends Exception
    {
        public FileAlreadyFoundException()
        {
            super();
        }
        public FileAlreadyFoundException(String aMessage)
        {
            super(aMessage);
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<!-- File: activity_edit_light_program.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical">

     <EditText
         android:id="@+id/editTextLightProgramName"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:ems="10"
         android:inputType="text">
         <requestFocus />
     </EditText>

     <!-- HR -->
     <ImageView
         android:layout_width="match_parent"
         android:layout_height="@dimen/hr_height"
         android:layout_marginTop="@dimen/hr_top_bottom_margin"
         android:layout_marginBottom="@dimen/hr_top_bottom_margin"
         android:layout_marginLeft="@dimen/hr_margin"
         android:layout_marginRight="@dimen/hr_margin"
         android:background="@color/hr_color"
         />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="@dimen/default_margin"
        android:layout_marginTop="@dimen/default_margin"
        android:layout_marginLeft="@dimen/default_margin"
        android:textSize="@dimen/title_size"
        android:textColor="@color/title_color"
        android:text="@string/title_edit_light_sequences" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btnRedSeq"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="@string/button_red_sequence"
            android:layout_weight="1"/>

        <Space android:layout_width="0dp"
            android:layout_height="1dp"
            android:layout_weight="1"/>

    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btnGreenSeq"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="@string/button_green_sequence"
```

```
            android:layout_weight="1"/>
        <Space android:layout_width="0dp"
            android:layout_height="1dp"
            android:layout_weight="1"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/btnBlueSeq"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="@string/button_blue_sequence"
            android:layout_weight="1"/>
        <Space android:layout_width="0dp"
            android:layout_height="1dp"
            android:layout_weight="1"/>
    </LinearLayout>

    <Space android:layout_height="0dp"
        android:layout_width="fill_parent"
        android:layout_weight="1"/>

    <!-- HR -->
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="@dimen/hr_height"
        android:layout_marginTop="20dp"
        android:layout_marginBottom="@dimen/hr_top_bottom_margin"
        android:layout_marginLeft="@dimen/hr_margin"
        android:layout_marginRight="@dimen/hr_margin"
        android:background="@color/hr_color"
        />



    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="@dimen/default_margin">

        <Button
            android:id="@+id/btnSaveLightProg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:layout_marginRight="@dimen/hr_margin"
            android:text="@string/button_save_light_program" />

        <Button
            android:id="@+id/btnDeleteLightProg"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginLeft="@dimen/hr_margin"
            android:text="@string/button_delete_ligh_program" />

     </RelativeLayout>
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<!-- File: activity_edit_sequence.xml -->
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:dslv="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
xmlns:app="http://schemas.android.com/apk/res/com.ile.moodlight">

    <Button
        android:id="@+id/btnSaveSequence"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_marginBottom="10dp"
        android:layout_marginRight="10dp"
        android:text="@string/button_save_sequence" />

    <com.mobeta.android.dslv.DragSortListView
        android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_above="@+id/btnSaveSequence"
        android:layout_margin="10dp"
        android:dividerHeight="5dp"
        android:fastScrollEnabled="true"
        android:paddingBottom="0dp"
        android:paddingLeft="0dp"
        android:paddingTop="0dp"
        android:paddingRight="0dp"
        dslv:click_remove_id="@id/click_remove"
        dslv:collapsed_height="2dp"
        dslv:drag_enabled="true"
        dslv:drag_handle_id="@id/drag_handle"
        dslv:drag_scroll_start="0.33"
        dslv:drag_start_mode="onDown"
        dslv:float_alpha="0.4"
        dslv:max_drag_scroll_speed="0.5"
        dslv:remove_enabled="true"
        dslv:remove_mode="clickRemove"
        dslv:slide_shuffle_speed="0.3"
        dslv:sort_enabled="true"
        dslv:track_drag_sort="false"
        dslv:use_default_controller="true" />

    <Button
        android:id="@+id/btnAddNewCommand"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="10dp"
        android:text="@string/button_addNewCommand" />

</RelativeLayout>

<?xml version="1.0" encoding="utf-8"?>
<!-- File: activity_new_command.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

<!-- Row 0 -->

    <LinearLayout
        android:layout_height="wrap_content"
```

```
    android:layout_width="match_parent"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/text_new_command_type"/>

    <Spinner
        android:id="@+id/spinner_command_types"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:layout_marginLeft="@dimen/default_margin"/>
</LinearLayout>

<!-- Row 1 -->
<LinearLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:orientation="horizontal">

    <TextView
     android:id="@+id/textTargetValue"
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:text="@string/text_new_command_target_value"
     android:layout_weight="5"/>

 <EditText
     android:id="@+id/editTextTargetValue"
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:inputType="numberSigned"
     android:layout_weight="5"/>
</LinearLayout>

<!-- Row 2 -->
<LinearLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textStepSize"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="@string/text_new_command_step_size"
        android:layout_weight="5"/>

    <EditText
        android:id="@+id/editTextStepSize"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:inputType="numberSigned"
        android:layout_weight="5"/>
</LinearLayout>

<!-- Row 3 -->
<LinearLayout
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:orientation="horizontal">

    <Space
        android:layout_width="0dp"
```

```
            android:layout_height="match_parent"
            android:layout_weight="1" />
        <CheckBox
            android:id="@+id/checkBoxWaitColor1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="-ColorName-"
            android:layout_weight="4" />
        <Space
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1" />
    </LinearLayout>

    <!-- Row 4 -->
    <LinearLayout
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:orientation="horizontal">

        <Space
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1" />
        <CheckBox
            android:id="@+id/checkBoxWaitColor2"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="-ColorName-"
            android:layout_weight="4" />
        <Space
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1" />
    </LinearLayout>

    <LinearLayout android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp">

        <Button
            android:id="@+id/btnNewCommandCancel"
            style="?android:attr/buttonStyleSmall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/button_new_command_cancel" />

        <Space
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight="1" />

        <Button
            android:id="@+id/btnNewCommandOK"
            style="?android:attr/buttonStyleSmall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/button_new_command_ok" />

    </LinearLayout>
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<!-- File: activity_new_light_program.xml -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <EditText
        android:id="@+id/editTextNewProgramName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:ems="10"
        android:text="@string/text_new_program_default_name">

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/btnNewProgramOK"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/editTextNewProgramName"
        android:layout_below="@+id/editTextNewProgramName"
        android:layout_marginTop="62dp"
        android:text="@string/button_new_program_ok" />

</RelativeLayout>


<?xml version="1.0" encoding="utf-8"?>
<!-- File: activity_edit_light_program.xml -->
<!-- Copyright (C) 2009 The Android Open Source Project

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
     See the License for the specific language governing permissions and
     limitations under the License.

     Edited by: Ilkka Saarelainen
-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >
    <TextView android:id="@+id/title_paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_paired_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1"
```

```
    />
    <TextView android:id="@+id/title_new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/new_devices"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="2"
    />
    <Button android:id="@+id/button_scan"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_scan"
    />
</LinearLayout>

<?xml version="1.0" encoding="utf-8"?>
<!-- File: activity_top_container.xml -->
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".TopContainerActivity" />

<?xml version="1.0" encoding="utf-8"?>
<!-- File: device_name.xml -->
<!-- Copyright (C) 2009 The Android Open Source Project

     Licensed under the Apache License, Version 2.0 (the "License");
     you may not use this file except in compliance with the License.
     You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

     Unless required by applicable law or agreed to in writing, software
     distributed under the License is distributed on an "AS IS" BASIS,
     WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
     See the License for the specific language governing permissions and
     limitations under the License.

     Edited by: Ilkka Saarelainen
-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="5dp"
/>

<?xml version="1.0" encoding="utf-8"?>
<!-- File: device_name.xml -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:id="@+id/btnNewProgram"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true"
            android:layout_marginTop="20dp"
            android:layout_marginBottom="@dimen/default_margin"
            android:layout_marginLeft="@dimen/default_margin"
            android:text="@string/button_new_light_program" />

        <Button
            android:id="@+id/btnEditProgram"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_toRightOf="@id/btnNewProgram"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="@dimen/default_margin"
            android:layout_marginLeft="@dimen/default_margin"
            android:text="@string/button_edit_light_program" />

        <Button
            android:id="@+id/btnUseProgram"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:layout_marginTop="20dp"
            android:layout_marginBottom="@dimen/default_margin"
            android:layout_marginLeft="@dimen/default_margin"
            android:text="@string/button_send_light_program"
            android:layout_marginRight="@dimen/default_margin"/>

        <!-- HR -->
        <ImageView
            android:id="@id/hr"
            android:layout_width="match_parent"
            android:layout_height="@dimen/hr_height"
            android:layout_marginTop="@dimen/hr_top_bottom_margin"
            android:layout_marginBottom="@dimen/hr_top_bottom_margin"
            android:layout_marginLeft="@dimen/hr_margin"
            android:layout_marginRight="@dimen/hr_margin"
            android:background="@color/hr_color"
            android:layout_above="@id/btnEditProgram"
            />
        >
        <ListView
            android:id="@android:id/list"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:choiceMode="singleChoice"
            android:focusable="true"
            android:layout_above="@id/hr">
        </ListView>
    </RelativeLayout>

    <?xml version="1.0" encoding="utf-8"?>
    <!-- File: fragment_maintenance.xml -->
    <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >


        <LinearLayout
             android:layout_width="match_parent"
```

```
 android:layout_height="wrap_content"
 android:orientation="vertical">

<TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_marginBottom="@dimen/default_margin"
     android:layout_marginTop="@dimen/default_margin"
     android:layout_marginLeft="@dimen/default_margin"
     android:textSize="@dimen/title_size"
     android:textColor="@color/title_color"
     android:text="@string/title_upload_flash" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btnFlashLoad"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_load_flash"
        android:layout_margin="@dimen/default_margin"/>
    <Button
        android:id="@+id/btnFlashUpload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_upload_flash"
        android:layout_margin="@dimen/default_margin"
        android:enabled="false"/>

</LinearLayout>

 <TextView
     android:id="@+id/textFlashUploadStatus"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_marginBottom="@dimen/default_margin"
     />

 <!-- HR -->
 <ImageView
     android:layout_width="match_parent"
     android:layout_height="@dimen/hr_height"
     android:layout_marginTop="@dimen/hr_top_bottom_margin"
     android:layout_marginBottom="@dimen/hr_top_bottom_margin"
     android:layout_marginLeft="@dimen/hr_margin"
     android:layout_marginRight="@dimen/hr_margin"
     android:background="@color/hr_color"
     />

 <TextView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_marginBottom="@dimen/default_margin"
     android:layout_marginTop="@dimen/default_margin"
     android:layout_marginLeft="@dimen/default_margin"
     android:textSize="@dimen/title_size"
     android:textColor="@color/title_color"
     android:text="@string/title_change_pin" />

 <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
```

```
        <EditText
            android:id="@+id/editTextBluetoothPin"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:maxLength="16"
            android:inputType="number" >
        </EditText>

        <Button
           android:id="@+id/btnChangePin"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:text="@string/button_change_pin"
           android:layout_margin="@dimen/default_margin"
           android:enabled="false"/>
    </LinearLayout>

    <TextView
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginBottom="@dimen/default_margin"
         android:layout_marginTop="@dimen/default_margin"
         android:layout_marginLeft="@dimen/default_margin"
         android:text="@string/title_pin_help" />

    <!-- HR -->
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="@dimen/hr_height"
        android:layout_marginTop="@dimen/hr_top_bottom_margin"
        android:layout_marginBottom="@dimen/hr_top_bottom_margin"
        android:layout_marginLeft="@dimen/hr_margin"
        android:layout_marginRight="@dimen/hr_margin"
        android:background="@color/hr_color"
        />

    <TextView
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginBottom="@dimen/default_margin"
         android:layout_marginTop="@dimen/default_margin"
         android:layout_marginLeft="@dimen/default_margin"
         android:textSize="@dimen/title_size"
         android:textColor="@color/title_color"
         android:text="@string/title_change_name" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/editTextBluetoothName"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:ems="10"
            android:maxLength="16">
        </EditText>

        <Button
            android:id="@+id/btnChangeName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/button_change_pin"
            android:layout_margin="@dimen/default_margin"
```

```
                android:enabled="false"/>
        </LinearLayout>
        <TextView
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:layout_marginBottom="@dimen/default_margin"
             android:layout_marginTop="@dimen/default_margin"
             android:layout_marginLeft="@dimen/default_margin"
             android:text="@string/title_name_help" />

    </LinearLayout>
</ScrollView>

<?xml version="1.0" encoding="UTF-8"?>
<!-- File: fragment_static_color.xml -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <FrameLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/colorPickerContainer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center" >
    </FrameLayout>

    <Button
        android:id="@+id/btnTogglePower"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="@dimen/ambilwarna_spacer"
        android:text="@string/button_toggle_power" />

    <SeekBar
        android:id="@+id/seekBarRed"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/btnTogglePower"
        android:max="255"
        android:progress="100"
        android:progressDrawable="@drawable/seekbar_progress_red"
        android:thumb="@drawable/seek_thumb_normal" />

    <SeekBar
        android:id="@+id/seekBarGreen"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/seekBarRed"
        android:max="255"
        android:progress="100"
        android:layout_marginTop="@dimen/seekbar_spacing"
        android:progressDrawable="@drawable/seekbar_progress_green"
        android:thumb="@drawable/seek_thumb_normal" />

    <SeekBar
        android:id="@+id/seekBarBlue"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/seekBarGreen"
```

```
            android:max="255"
            android:progress="100"
            android:layout_marginTop="@dimen/seekbar_spacing"
            android:progressDrawable="@drawable/seekbar_progress_blue"
            android:thumb="@drawable/seek_thumb_normal" />

        <RelativeLayout
            android:id="@+id/ambilwarna_viewContainer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/seekBarBlue"
            android:layout_centerHorizontal="true"
            android:clipToPadding="false"
            android:paddingBottom="@dimen/ambilwarna_spacer"
            android:paddingLeft="@dimen/ambilwarna_spacer"
            android:paddingRight="@dimen/ambilwarna_spacer"
            android:paddingTop="@dimen/ambilwarna_spacer" >

            <yuku.ambilwarna.AmbilWarnaKotak
                android:id="@+id/ambilwarna_viewSatBri"
                android:layout_width="@dimen/ambilwarna_hsvWidth"
                android:layout_height="@dimen/ambilwarna_hsvHeight"
                android:layerType="software" />

            <ImageView
                android:id="@+id/ambilwarna_viewHue"
                android:layout_width="@dimen/ambilwarna_hueWidth"
                android:layout_height="@dimen/ambilwarna_hsvHeight"
                android:layout_marginLeft="@dimen/ambilwarna_spacer"
                android:layout_toRightOf="@id/ambilwarna_viewSatBri"
                android:scaleType="fitXY"
                android:src="@drawable/ambilwarna_hue" />

            <ImageView
                android:id="@+id/ambilwarna_cursor"
                android:layout_width="9dp"
                android:layout_height="9dp"
                android:scaleType="matrix"
                android:src="@drawable/ambilwarna_cursor" />

            <ImageView
                android:id="@+id/ambilwarna_target"
                android:layout_width="15dp"
                android:layout_height="15dp"
                android:scaleType="matrix"
                android:src="@drawable/ambilwarna_target" />
        </RelativeLayout>

        <ToggleButton
            android:id="@+id/toggleColorCorrection"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignLeft="@+id/ambilwarna_viewContainer"
            android:layout_alignParentTop="true"
            android:textOn="@string/button_toggle_color_correction"
            android:textOff="@string/button_toggle_color_correction" />

    </RelativeLayout>

    <?xml version="1.0" encoding="utf-8"?>
    <!-- File: list_item_click_remove.xml -->
    <RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="@dimen/item_height">
```

```
    <ImageView
        android:id="@id/drag_handle"
        android:background="#FF0000"
        android:layout_width="@dimen/item_height"
        android:layout_height="@dimen/item_height"/>

    <ImageView
        android:id="@id/click_remove"
        android:layout_width="@dimen/item_height"
        android:layout_height="@dimen/item_height"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:background="@drawable/delete_x" />

    <TextView
        android:id="@+id/textCommand"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/item_height"
        android:layout_alignParentTop="true"
        android:layout_toLeftOf="@id/click_remove"
        android:layout_toRightOf="@id/drag_handle"
        android:gravity="center_vertical"
        android:paddingLeft="8dp"
        android:text="text"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: colors.xml -->
<resources>
    <color name="blue">#33b5e5</color>
    <color name="darkblue">#0099cc</color>
    <color name="red">#ff4444</color>
    <color name="grayStart">#adadad</color>
    <color name="grayEnd">#ffffff</color>
    <color name="black">#000000</color>
    <color name="bg_handle_pressed">#606060</color>
    <color name="hr_color">#808080</color>
    <color name="title_color">#F5F5F5</color>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: dimens.xml -->
<resources>
    <dimen name="item_height">55dp</dimen>
    <dimen name="section_div_height">20dp</dimen>
    <dimen name="seekbar_spacing">10dp</dimen>
    <dimen name="hr_height">2dp</dimen>
    <dimen name="hr_margin">5dp</dimen>
    <dimen name="hr_top_bottom_margin">6dp</dimen>
    <dimen name="default_margin">5dp</dimen>
    <dimen name="title_size">20dp</dimen>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: ids.xml -->
<resources xmlns:android="http://schemas.android.com/apk/res/android">
    <item type="id" name="drag_handle" />
    <item type="id" name="click_remove" />
    <item type="id" name="hr" />
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: strings.xml -->
<resources>
    <string name="app_name">Mood Light</string>
    <string name="title_activity_main">Mood Light</string>
    <string name="title_fragment_static_color">Static color</string>
    <string name="title_fragment_light_program">Light Programs</string>
    <string name="title_fragment_maintenance">Maintenance</string>
    <string name="title_paired_devices">Paired devices</string>
    <string name="title_other_devices">Unpaired devices</string>
    <string name="button_scan">Scan</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_select_device">Select device</string>
    <string name="scanning">Scanning...</string>
    <string name="select_device">Select device</string>
    <string name="none_found">No devices found!</string>
    <string name="none_paired">No paired devices.</string>
    <string name="hello_world">Hello world!</string>
    <string name="title_activity_light_program_edit">Edit light program</string>
    <string name="button_red_sequence">Red sequence</string>
    <string name="button_green_sequence">Green sequence</string>
    <string name="button_blue_sequence">Blue sequence</string>
    <string name="button_save_light_program">Save changes</string>
    <string name="title_activity_edit_light_sequence">Edit Light Sequence</string>
    <string name="button_save_sequence">OK</string>
    <string name="button_new_command_ok">OK</string>
```

```
    <string name="button_new_command_cancel">Cancel</string>
    <string name="text_new_command_type">Command type</string>
    <string name="text_new_command_target_value">Target value</string>
    <string name="text_new_command_step_size">Step size</string>
    <string name="button_send_light_program">Select</string>
    <string name="button_edit_light_program">Edit</string>
    <string name="button_new_light_program">New</string>
    <string name="title_activity_new_command">New Command</string>
    <string name="title_activity_edit_command">Edit Command</string>
    <string name="button_addNewCommand">Add new command</string>
    <string name="button_toggle_power">Power off</string>
    <string name="button_load_flash">Load flash</string>
    <string name="button_upload_flash">Upload Flash</string>
    <string name="title_upload_flash">Flash upgrade</string>
    <string name="title_change_pin">Change bluetooth password</string>
    <string name="title_change_name">Change bluetooth name</string>
    <string name="button_change_pin">Change</string>
    <string name="title_pin_help">Only numbers are allowed. Max lenght is 16.</string>
    <string name="title_name_help">ASCII characters allowed. Max lenght is 16.</string>
    <string name="path_lightprogram_dir">/MoodLight/LightPrograms</string>
    <string name="toast_error_program_name_already_in_use">The given name is already in
use!</string>
    <string name="toast_notice_program_save_ok">Saved.</string>
    <string name="red_name">Red</string>
    <string name="green_name">Green</string>
    <string name="blue_name">Blue</string>
    <string name="title_activity_new_light_program_dialog">Program Name:</string>
    <string name="text_new_program_default_name">New Program</string>
    <string name="button_new_program_ok">OK</string>
    <string name="error_while_creating_new_lp">Unknown error while creating the
file.</string>
    <string name="title_edit_light_sequences">Edit color sequences.</string>
    <string name="button_delete_ligh_program">Delete</string>
    <string name="error_while_uploading_lp">Unknown error while loading the pro-
gram!</string>
    <string name="notify_lp_sent_successfully">Program sent to the device!</string>
    <string name="button_toggle_color_correction">Color correction</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: seekbar_progress_bg_blue.xml -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item>
        <clip>
            <shape>
                <gradient
                    android:startColor="#60101040"
                    android:endColor="#A00000FF"
                    android:angle="0"
                    />
            </shape>
        </clip>
    </item>
</layer-list>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: seekbar_progress_bg_green.xml -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item>
        <clip>
            <shape>
                <gradient
                    android:startColor="#6010FF10"
                    android:endColor="#A000FF00"
                    android:angle="0"
                    />
            </shape>
        </clip>
    </item>
</layer-list>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: seekbar_progress_bg_red.xml -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item>
        <clip>
            <shape>
                <gradient
                    android:startColor="#60401010"
                    android:endColor="#A0FF0000"
                    android:angle="0"
                    />
            </shape>
        </clip>
    </item>
</layer-list>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- File: seekbar_progress_blue.xml -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@android:id/background">
        <nine-patch
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:src="@drawable/seek_bg2_9"
            android:dither="true"
         />
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip>
```

```
            <shape>
                <gradient
                    android:startColor="#80028ac8"
                    android:centerColor="#80127fb1"
                    android:centerY="0.75"
                    android:endColor="#a004638f"
                    android:angle="270"
                />
            </shape>
        </clip>
    </item>
    <item
        android:id="@android:id/progress"
        android:drawable="@drawable/seekbar_progress_bg_blue"
    />
</layer-list>


<?xml version="1.0" encoding="utf-8"?>
<!-- File: seekbar_progress_green.xml -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@android:id/background">
        <nine-patch
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:src="@drawable/seek_bg2_9"
            android:dither="true"
         />
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip>
            <shape>
                <gradient
                    android:startColor="#80028ac8"
                    android:centerColor="#80127fb1"
                    android:centerY="0.75"
                    android:endColor="#a004638f"
                    android:angle="270"
                />
            </shape>
        </clip>
    </item>
    <item
        android:id="@android:id/progress"
        android:drawable="@drawable/seekbar_progress_bg_green"
    />
</layer-list>


<?xml version="1.0" encoding="utf-8"?>
<!-- File: seekbar_progress_red.xml -->
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@android:id/background">
        <nine-patch
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:src="@drawable/seek_bg2_9"
            android:dither="true"
         />
    </item>
    <item android:id="@android:id/secondaryProgress">
        <clip>
            <shape>
                <gradient
                    android:startColor="#80028ac8"
                    android:centerColor="#80127fb1"
                    android:centerY="0.75"
                    android:endColor="#a004638f"
```

```
                    android:angle="270"
                />
            </shape>
        </clip>
    </item>
    <item
        android:id="@android:id/progress"
        android:drawable="@drawable/seekbar_progress_bg_red"
    />
</layer-list>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ile.moodlight"
    android:versionCode="1"
    android:versionName="0.0.1">
<!-- File: AndroidManifest.xml -->

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="11" />

    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@android:style/Theme.Holo" >
        <activity
            android:name=".activity.TopContainerActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".activity.SelectDeviceActivity"
            android:configChanges="orientation|keyboardHidden"
            android:label="@string/title_activity_select_device"
            android:theme="@android:style/Theme.Holo.Dialog" />
        <activity
            android:name=".activity.EditLightProgramActivity"
            android:label="@string/title_activity_light_program_edit" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="TopContainerActivity" />
        </activity>
        <activity
            android:name=".activity.EditLightSequenceActivity"
            android:label="@string/title_activity_edit_light_sequence" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="LightProgramEditActivity" />
        </activity>
        <activity
            android:name=".activity.NewCommandActivity"
            android:configChanges="orientation|keyboardHidden"
            android:label="@string/title_activity_new_command"
            android:theme="@android:style/Theme.Holo.Dialog" />
        <activity
            android:name=".activity.NewLightProgramDialogActivity"
            android:label="@string/title_activity_new_light_program_dialog"
            android:configChanges="orientation|keyboardHidden"
```

```
            android:theme="@android:style/Theme.Holo.Dialog"  >
        </activity>
    </application>

</manifest>
```

```
/*****************************************************************************
*
* Atmel Corporation
*
* File              : bootloader.c
* Compiler          : IAR C 3.10C Kickstart, AVR-GCC/avr-libc(>= 1.2.5)
* Revision          : $Revision: 1.7 $
* Date              : $Date: Tuesday, June 07, 200 $
* Updated by        : $Author: raapeland $
* Modified by       : Ilkka Saarelainen, 2.4.2013
*
* Support mail      : avr@atmel.com
*
* Target platform   : All AVRs with bootloader support
*
* AppNote           : AVR109 - Self-programming
*
* Description   : This Program allows an AVR with bootloader capabilities to
*                 Read/write its own Flash/EEprom. To enter Programming mode
*                 an input pin is checked. If this pin is pulled low, programming mode
*                 is entered. If not, normal execution is done from $0000
*                 "reset" vector in Application area.
*
* Preparations  : Use the preprocessor.xls file for obtaining a customized
*                 defines.h file and linker-file code-segment definition for
*                 the device you are compiling for.
*****************************************************************************/
#include "defines.h"
#include "serial.h"
#include "flash.h"

#define ADDR_T unsigned int

unsigned char BlockLoad(unsigned int size, unsigned char mem, ADDR_T *address);
void BlockRead(unsigned int size, unsigned char mem, ADDR_T *address);

/* BLOCKSIZE should be chosen so that the following holds: BLOCKSIZE*n = PAGESIZE,
where n=1,2,3... */
#define BLOCKSIZE PAGESIZE


#ifdef __ICCAVR__
#  define C_TASK __C_task
#else /* ! __ICCAVR__ */
#  define C_TASK /**/
#endif /* __ICCAVR__ */

void WDT_off(void)
{
            asm("cli");
            wdt_reset();
            /* Clear WDRF in MCUSR */
            MCUSR &= ~(1<<WDRF);
            /* Write logical one to WDCE and WDE */
            /* Keep old prescaler setting to prevent unintentional time-out */
            WDTCSR |= (1<<WDCE) | (1<<WDE);
            /* Turn off WDT */
            WDTCSR = 0x00;
            asm("sei");
}

C_TASK int main(void)
{
```

```
            // Disable watchdog as recommended in data sheet page 54.
            WDT_off();

    ADDR_T address;
    unsigned int temp_int;
    unsigned char val;

    /* Initialization */
    void (*funcptr)( void ) = 0x0000; // Set up function pointer to RESET vector.
    PROGPORT |= (1<<PROG_NO); // Enable pull-up on PROG_NO line on PROGPORT.
    initbootuart(); // Initialize UART.

            sendchar('B');
            sendchar('O');
            sendchar('O');
            sendchar('T');

    /* Branch to bootloader or application code? */
    if( !(PROGPIN & (1<<PROG_NO)) ) // If PROGPIN is pulled low, enter programmingmode.
    {
        /* Main loop */
        for(;;)
        {
            val=recchar(); // Wait for command character.

            // Check autoincrement status.
            if(val=='a')
            {
                sendchar('Y'); // Yes, we do autoincrement.
            }

            // Set address.
            else if(val=='A') // Set address...
            { // NOTE: Flash addresses are given in words, not bytes.
                address=(recchar()<<8) | recchar(); // Read address high and low byte.
                sendchar('\r'); // Send OK back.
            }

            // Chip erase.
            else if(val=='e')
            {
                for(address = 0; address < APP_END;address += PAGESIZE)
                { // NOTE: Here we use address as a byte-address, not word-address, for
convenience.
                    _WAIT_FOR_SPM();
#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
                    _PAGE_ERASE( address );
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif
                }

                sendchar('\r'); // Send OK back.
            }

            // Check block load support.
            else if(val=='b')
                            {
                                    sendchar('Y'); // Report block load supported.
                                    sendchar((BLOCKSIZE>>8) & 0xFF); // MSB first.
                                    sendchar(BLOCKSIZE&0xFF); // Report BLOCKSIZE
(bytes).
                        }
```

```
            // Start block load.
                        else if(val=='B')
                        {
                            temp_int = (recchar()<<8) | recchar(); // Get block size.
                                     val = recchar(); // Get memtype.
                                         sendchar( BlockLoad(temp_int,val,&address) );
// Block load.
                            }


                            // Start block read.
                        else if(val=='g')
                        {
                            temp_int = (recchar()<<8) | recchar(); // Get block size.
                                     val = recchar(); // Get memtype
                                     BlockRead(temp_int,val,&address); // Block read
                        }
            // Read program memory.
            else if(val=='R')
            {
                // Send high byte, then low byte of flash word.
                _WAIT_FOR_SPM();
                _ENABLE_RWW_SECTION();
#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
                sendchar( _LOAD_PROGRAM_MEMORY( (address << 1)+1 ) );
                sendchar( _LOAD_PROGRAM_MEMORY( (address << 1)+0 ) );
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif

                address++; // Auto-advance to next Flash word.
            }


            // Write program memory, low byte.
            else if(val=='c')
            { // NOTE: Always use this command before sending high byte.
                temp_int=recchar(); // Get low byte for later _FILL_TEMP_WORD.
                sendchar('\r'); // Send OK back.
            }


            // Write program memory, high byte.
            else if(val=='C')
            {
                temp_int |= (recchar()<<8); // Get and insert high byte.
                _WAIT_FOR_SPM();
#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
                _FILL_TEMP_WORD( (address << 1), temp_int ); // Convert word-address to
byte-address and fill.
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif
                address++; // Auto-advance to next Flash word.
                sendchar('\r'); // Send OK back.
            }
```

```
            // Write page.
            else if(val== 'm')
            {
                if( address >= (APP_END>>1) ) // Protect bootloader area.
                {
                    sendchar('?');
                } else
                {
                    _WAIT_FOR_SPM();
#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
                    _PAGE_WRITE( address << 1 ); // Convert word-address to byte-address
and write.
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif
                }

                sendchar('\r'); // Send OK back.
            }
            // Enter and leave programming mode.
            else if((val=='P')||(val=='L'))
            {
                sendchar('\r'); // Nothing special to do, just answer OK.
            }


            // Exit bootloader.
            else if(val=='E')
            {
                _WAIT_FOR_SPM();
                _ENABLE_RWW_SECTION();
                sendchar('\r');
                funcptr(); // Jump to Reset vector 0x0000 in Application Section.
            }


            // Get programmer type.
            else if (val=='p')
            {
                sendchar('S'); // Answer 'SERIAL'.
            }


            // Return supported device codes.
            else if(val=='t')
            {
#if PARTCODE+0 > 0
                sendchar( PARTCODE ); // Supports only this device, of course.
#endif /* PARTCODE */
                sendchar( 0 ); // Send list terminator.
            }


            // Set LED, clear LED and set device type.
            else if((val=='x')||(val=='y')||(val=='T'))
            {
                recchar(); // Ignore the command and it's parameter.
                sendchar('\r'); // Send OK back.
            }

            // Return programmer identifier.
            else if(val=='S')
            {
```

```
                sendchar('A'); // Return 'AVRBOOT'.
                sendchar('V'); // Software identifier (aka programmer signature) is al-
ways 7 characters.
                sendchar('R');
                sendchar('B');
                sendchar('O');
                sendchar('O');
                sendchar('T');
            }


            // Return software version.
            else if(val=='V')
            {
                sendchar('1');
                sendchar('5');
            }


            // Return signature bytes.
            else if(val=='s')
            {

                sendchar( SIGNATURE_BYTE_3 );
                sendchar( SIGNATURE_BYTE_2 );
                sendchar( SIGNATURE_BYTE_1 );
            }


            // The last command to accept is ESC (synchronization).
            else if(val!=0x1b)                  // If not ESC, then it is unrecog-
nized...
            {
                sendchar('?');
            }
        } // end: for(;;)
    }
    else
    {
        _WAIT_FOR_SPM();
        _ENABLE_RWW_SECTION();
        funcptr(); // Jump to Reset vector 0x0000 in Application Section.
    }
} // end: main

unsigned char BlockLoad(unsigned int size, unsigned char mem, ADDR_T *address)
{
    unsigned char buffer[BLOCKSIZE];
    unsigned int data;
    ADDR_T tempaddress;

    // EEPROM memory type.
    if(mem=='E')
    {
        /* Fill buffer first, as EEPROM is too slow to copy with UART speed */
        for(tempaddress=0;tempaddress<size;tempaddress++)
            buffer[tempaddress] = recchar();

        /* Then program the EEPROM */
        _WAIT_FOR_SPM();
             for( tempaddress=0; tempaddress < size; tempaddress++)
             {
                     EEARL = *address; // Setup EEPROM address
            EEARH = ((*address) >> 8);
            EEDR = buffer[tempaddress]; // Get byte.
            EECR |= (1<<EEMPE); // Write byte.
```

```
            EECR |= (1<<EEPE);
            while (EECR & (1<<EEPE)) // Wait for write operation to finish.
                ;

                                        (*address)++; // Select next EEPROM byte
        }

        return '\r'; // Report programming OK
    }

    // Flash memory type.
             else if(mem=='F')
    { // NOTE: For flash programming, 'address' is given in words.
        (*address) <<= 1; // Convert address to bytes temporarily.
        tempaddress = (*address);  // Store address in page.

        do
                          {
            data = recchar();
            data |= (recchar() << 8);
#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
            _FILL_TEMP_WORD(*address,data);
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif
            (*address)+=2; // Select next word in memory.
            size -= 2; // Reduce number of bytes to write by two.
        } while(size); // Loop until all bytes written.

#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
             _PAGE_WRITE(tempaddress);
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif
             _WAIT_FOR_SPM();
             _ENABLE_RWW_SECTION();

        (*address) >>= 1; // Convert address back to Flash words again.
        return '\r'; // Report programming OK
    }

    // Invalid memory type?
    else
    {
        return '?';
    }
}


void BlockRead(unsigned int size, unsigned char mem, ADDR_T *address)
{
    // EEPROM memory type.
    if (mem=='E') // Read EEPROM
    {
        do
        {
            EEARL = *address; // Setup EEPROM address
            EEARH = ((*address) >> 8);
            (*address)++; // Select next EEPROM byte
            EECR |= (1<<EERE); // Read EEPROM
```

```
            sendchar(EEDR); // Transmit EEPROM dat ato PC

            size--; // Decrease number of bytes to read
        } while (size); // Repeat until all block has been read
    }

    // Flash memory type.
             else if(mem=='F')
             {
        (*address) <<= 1; // Convert address to bytes temporarily.

        do
        {
#ifdef __ICCAVR__
#pragma diag_suppress=Pe1053 // Suppress warning for conversion from long-type address
to flash ptr.
#endif
            sendchar( _LOAD_PROGRAM_MEMORY(*address) );
            sendchar( _LOAD_PROGRAM_MEMORY((*address)+1) );
#ifdef __ICCAVR__
#pragma diag_default=Pe1053 // Back to default.
#endif
            (*address) += 2; // Select next word in memory.
            size -= 2; // Subtract two bytes from number of bytes to read
        } while (size); // Repeat until all block has been read

        (*address) >>= 1; // Convert address back to Flash words again.
    }
}

/*
 * File: defines.h
 */
/* definitions generated by preprocessor, copy into defines.h */
#ifndef     PPINC
#define     _ATMEGA328P  // device select: _ATMEGAxxxx
#define     _B1024       // boot size select: _Bxxxx (words), powers of two only
#ifdef      __ICCAVR__
#include    "iom328.h"
#endif
#if         __GNUC__
#include    <avr/io.h>
#endif

#include <avr/wdt.h>

#define F_CPU 8000000UL

/* define pin for enter-self-prog-mode */
#define     PROGPORT     PORTB
#define     PROGPIN      PINB
#define     PROG_NO      PORTB7

/* baud rate register value calculation */
#define     CPU_FREQ     8000000
#define     BAUD_RATE    9600
#define     BRREG_VALUE 103

/* definitions for UART control */
#define     BAUD_RATE_LOW_REG        UBRR0L
#define     UART_CONTROL_REG         UCSR0B
#define     ENABLE_TRANSMITTER_BIT   TXEN0
#define     ENABLE_RECEIVER_BIT      RXEN0
#define     UART_STATUS_REG          UCSR0A
```

```
#define      TRANSMIT_COMPLETE_BIT    TXC0
#define      RECEIVE_COMPLETE_BIT     RXC0
#define      UART_DATA_REG            UDR0

/* definitions for SPM control */
#define      SPMCR_REG    SPMCSR
#define      PAGESIZE     128
#define      APP_END      30720

/* definitions for device recognition */
#define      PARTCODE
#define      SIGNATURE_BYTE_1         0x1E
#define      SIGNATURE_BYTE_2         0x95
#define      SIGNATURE_BYTE_3         0x14

/* indicate that preprocessor result is included */
#define      PPINC
#endif

/*****************************************************************************
*
* Atmel Corporation
*
* File              : flash.h
* Compiler          : IAR C 3.10C Kickstart, AVR-GCC/avr-libc(>= 1.2.5)
* Revision          : $Revision: 1.7 $
* Date              : $Date: Tuesday, June 07, 200 $
* Updated by        : $Author: raapeland $
*
* Support mail      : avr@atmel.com
*
* Target platform   : All AVRs with bootloader support
*
* AppNote           : AVR109 - Self-programming
*
* Description       : Flash operations for AVR109 Self-programming
*****************************************************************************/

#if defined(__ICCAVR__)

/* IAR Embedded Workbench */
#  include <inavr.h>

#  define _GET_LOCK_BITS() __AddrToZByteToSPMCR_LPM( (void __flash *) 0x0001, 0x09 )
#  define _GET_LOW_FUSES() __AddrToZByteToSPMCR_LPM( (void __flash *) 0x0000, 0x09 )
#  define _GET_HIGH_FUSES() __AddrToZByteToSPMCR_LPM( (void __flash *) 0x0003, 0x09 )
#  define _GET_EXTENDED_FUSES() __AddrToZByteToSPMCR_LPM( (void __flash *) 0x0002, 0x09
)
#  define _SET_LOCK_BITS(data) __DataToR0ByteToSPMCR_SPM( data, 0x09 )
#  define _ENABLE_RWW_SECTION() __DataToR0ByteToSPMCR_SPM( 0x00, 0x11 )

#  define _WAIT_FOR_SPM() while( SPMCR_REG & (1<<SPMEN) );

#  ifndef LARGE_MEMORY
#    define _LOAD_PROGRAM_MEMORY(addr) __load_program_memory( (const unsigned char
__flash *) (addr) )
#    define _FILL_TEMP_WORD(addr,data) __AddrToZWordToR1R0ByteToSPMCR_SPM( (void __flash
*) (addr), data, 0x01 )
#    define _PAGE_ERASE(addr) __AddrToZByteToSPMCR_SPM( (void __flash *) (addr), 0x03 )
#    define _PAGE_WRITE(addr) __AddrToZByteToSPMCR_SPM( (void __flash *) (addr), 0x05 )
#  else /* LARGE_MEMORY */
#    define _LOAD_PROGRAM_MEMORY(addr) __extended_load_program_memory( (const unsigned
char __farflash *) (addr) )
#    define _FILL_TEMP_WORD(addr,data) __AddrToZ24WordToR1R0ByteToSPMCR_SPM( (void
__farflash *) (addr), data, 0x01 )
```

```
#    define _PAGE_ERASE(addr) __AddrToZ24ByteToSPMCR_SPM( (void __farflash *) (addr),
0x03 )
#    define _PAGE_WRITE(addr) __AddrToZ24ByteToSPMCR_SPM( (void __farflash *) (addr),
0x05 )
#  endif /* LARGE_MEMORY */

#elif __GNUC__ > 0

/* AVR-GCC/avr-libc */
#  include <avr/boot.h>
#  include <avr/pgmspace.h>

#  if defined(GET_LOCK_BITS)    /* avr-libc >= 1.2.5 */
#    define _GET_LOCK_BITS() boot_lock_fuse_bits_get(GET_LOCK_BITS)
#    define _GET_LOW_FUSES() boot_lock_fuse_bits_get(GET_LOW_FUSE_BITS)
#    define _GET_HIGH_FUSES() boot_lock_fuse_bits_get(GET_HIGH_FUSE_BITS)
#    define _GET_EXTENDED_FUSES() boot_lock_fuse_bits_get(GET_EXTENDED_FUSE_BITS)
#  endif /* defined(GET_LOCK_BITS) */
#  define _SET_LOCK_BITS(data) boot_lock_bits_set(~data)
#  define _ENABLE_RWW_SECTION() boot_rww_enable()

#  define _WAIT_FOR_SPM() boot_spm_busy_wait()

#  ifndef LARGE_MEMORY
#    define _LOAD_PROGRAM_MEMORY(addr) pgm_read_byte_near(addr)
#  else /* LARGE_MEMORY */
#    define _LOAD_PROGRAM_MEMORY(addr) pgm_read_byte_far(addr)
#  endif /* LARGE_MEMORY */
#  define _FILL_TEMP_WORD(addr,data) boot_page_fill(addr, data)
#  define _PAGE_ERASE(addr) boot_page_erase(addr)
#  define _PAGE_WRITE(addr) boot_page_write(addr)

#else
#  error "Don't know your compiler."
#endif

/*****************************************************************************
*
* Atmel Corporation
*
* File              : serial.h
* Compiler          : IAR C 3.10C Kickstart, AVR-GCC/avr-libc(>= 1.2.5)
* Revision          : $Revision: 1.7 $
* Date              : $Date: Tuesday, June 07, 200 $
* Updated by        : $Author: raapeland $
*
* Support mail      : avr@atmel.com
*
* Target platform   : All AVRs with bootloader support
*
* AppNote           : AVR109 - Self-programming
*
* Description       : Header file for serial.c
*****************************************************************************/

#include "defines.h"


void initbootuart(void)
{

    BAUD_RATE_LOW_REG = BRREG_VALUE;
    UART_CONTROL_REG = (1 << ENABLE_RECEIVER_BIT) |
    (1 << ENABLE_TRANSMITTER_BIT); // enable receive and transmit
```

```
    UART_STATUS_REG |= (1 << U2X0); //Double speed
}


void sendchar(unsigned char c)
{
    UART_DATA_REG = c;                                      // prepare transmission
    while (!(UART_STATUS_REG & (1 << TRANSMIT_COMPLETE_BIT)));// wait until byte
sendt
    UART_STATUS_REG |= (1 << TRANSMIT_COMPLETE_BIT);          // delete TXCflag
}


unsigned char recchar(void)
{
    while(!(UART_STATUS_REG & (1 << RECEIVE_COMPLETE_BIT)));  // wait for data
    return UART_DATA_REG;
}
```