Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Gašper Volf

Naprava za svetlobne efekte, vodena preko sistema Bluetooth

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Branko Šter

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Napišite aplikacijo za pametne telefone z operacijskim sistemom Android, ki služi za nadzor naprave za svetlobne efekte. Napravo naj nadzoruje radijsko na osnovi standarda Bluetooth. Opišite uporabljeni mikrokrmilnik na napravi ter parametre Bluetooth povezave. Opišite tudi delovanje aplikacije.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Gašper Volf sem avtor diplomskega dela z naslovom:

Naprava za svetlobne efekte, vodena preko sistema Bluetooth

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Branka Štera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. septembra 2015 Podpis avtorja:

Hvala mentorju, dr. Branku Šteru za strokovno pomoč ter nasvete. Hvala družini in prijateljem za podporo in prenašanje moje muhavosti tekom izdelave diplomske naloge. Posebno se zahvaljujem svoji sestri, ki me je prepričala, da je študij računalništva pravi zame.

To my future self.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	aplikacijski programski vmesnik
CPE	Central Processing Unit	centralna procesna enota
ROM	Read-Only Memory	bralni pomnilnik
RAM	Random Access Memory	bralno-pisalni pomnilnik
EEPROM	Electrically Erasable	električno izbrisljivi
	Programmable Read-Only Memory	programljivi bralni pomnilnik
GPIO	General Purpose Input/Output	splošno-namenski vhodi/izhodi
PSP	Interupt Service Routine (ISR)	prekinitveno-servisni program
GUI	Graphical User Interface	grafični uporabniški vmesnik

Povzetek

Ljudje se po naravi radi zabavamo. Na vsako dobro zabavo spadata glasna glasba in ples. Vse skupaj se še dodatno popestri, če se pleše v temi in zraven utripajo luči, ki polepšajo vzdušje. Na trgu je na voljo veliko izdelkov, ki lahko popestrijo dogajanje na plesišču, vendar so dragi, ali pa imajo omejene funkcionalnosti. Da bi popestril dogajanje na lokalnih praznovanjih in zabavah, sem v sklopu diplomske naloge razvil program za krmiljenje naprave za svetlobne efekte. Da pa bi bilo vse skupaj lažje za uporabo, sem razvil še aplikacijo za mobilne telefone z operacijskim sistem Android, ki omogoča upravljanje z napravo za svetlobne efekte preko standarda Bluetooth.

Ključne besede: mikrokrmilnik, svetlobni efekti, Bluetooth, Android.

Abstract

People love to have fun. One of the best entertainment options are parties. All good parties include loud music and dancing. Dancing is enriched by darkness and flashing lights, that brighten up the atmosphere. There are a lot of light shows available on the market. But they are either very expensive or lack features. To spice up local celebrations and parties, I developed a program for controlling a light show device as part of the thesis. To make the usage of the device as simple as possible I also developed an application for Android phones, that allows controlling of the light show device using the Bluetooth standard.

Keywords: microcontroller, light show, Bluetooth, Android.

Poglavje 1

Uvod

Prijatelj je za zaključni izdelek v srednji šoli izdelal napravo za svetlobne učinke. Napravo smo uporabljali na raznovrstnih praznovanjih, da smo malo popestrili plesišče. Ker pa si je bilo potrebno zapomniti, katera tipka ima katero funkcijo, saj so vse enake, me je prijatelj vprašal, če bi bilo možno narediti računalniški program za oddaljen nadzor. Po krajšem iskanju po medmrežju sem ugotovil, da bi se dalo, vendar bi bilo potrebno zamenjati obstoječi mikrokrmilnik ter dodati Bluetooth sprejemnik. Prijatelj je bil nad odgovorom navdušen in tako se je rodila ideja za mojo diplomsko nalogo.

V okviru diplomske naloge sem moral najprej prenesti vse funkcionalnosti starega mikrokrmilnika (AT89C52) na novega (ATMega64). Potem sem moral dodati nove funkcionalnosti, med katere spada komunikacija z Bluetooth vmesnikom ter določiti protokol za komuniciranje med mikrokrmilnikom ter Android aplikacijo. Nazadnje sem se lotil Android aplikacije.

V poglavju 2 je opisana naprava za svetlobne efekte in njene ključne funkcionalnosti. V poglavju 3 opisujem mikrokrmilnik, njegove lastnosti ter potrebne korake in funkcije za zagotovitev potrebnih funkcionalnosti naprave. Poglavje 4 opisuje operacijski sistem Android ter opis aktivnosti, ki jih moja aplikacija ponuja. Na koncu sledi še zaključek, kjer so opisani problemi, s katerimi sem se srečal tekom diplomskega dela, ter možne izboljšave naprave za svetlobne efekte in Android aplikacije.

Poglavje 2

Naprava za svetlobne efekte

2.1 Opis

Naprava za svetlobne učinke je 8-kanalna, kar pomeni, da je nanjo mogoče priklopiti 8 žarnic. Poleg žarnic ima tudi izhod za stroboskop.

Naprava ima 16 tipk, ki omogočajo upravljanje z lučkami, hitrost utripanja in podobno. Ima tudi LED ekran, na katerem se prikazuje trenutno izvajani program, korak programa in hitrost utripanja lučk. Poleg ekrana ima še 8 LED diod, ki služijo kot indikator trenutno prižganih luči. Na sliki 2.2 si lahko ogledamo sprednjo ploščo naprave.

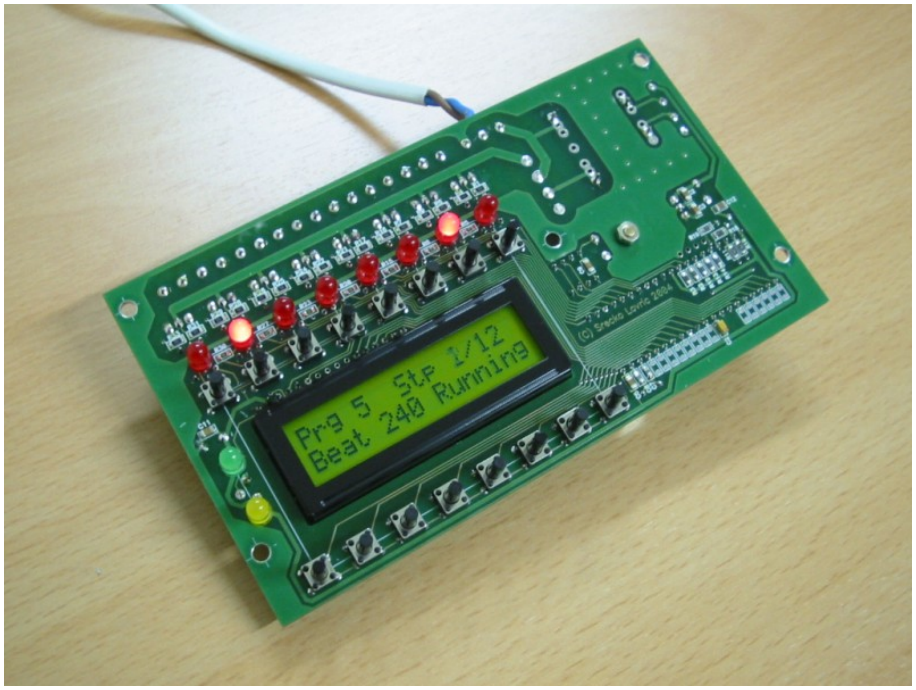
S prijateljem sva naredila tudi lesen zaboj, ki ima vgrajene nosilce za žarnice in ventilatorje. Ventilatorji so potrebni za hlajenje, saj se žarnice močno segrevajo in bi lahko prišlo do požara. Slika 2.1 prikazuje napravo v delovanju.

2.2 Funkcionalnosti

Naprava ima 3 načine delovanja: način takta, ročni način in način programiranja.

V načinu takta se izvaja programe po taktu, ki ga določi uporabnik. Program je sestavljen iz zaporedja korakov. Vsak korak ima definirano, katere

Slika 2.1: Naprava za svetlobne efekte v delovanju



Slika 2.2: Sprednja stran naprave za svetlobne efekte

izmed osmih luči so prižgane in katere ugasnjene. Takt nam pove, kolikokrat v minuti se bo zamenjal korak. Minimalna vrednost takta je 10, maksimalna 800, privzeta ob vklopu naprave pa je 200.

V ročnem načinu se koraki ne spreminjajo po taktu, ampak s pritiski na tipko.

Programsko je v napravo vpisanih 10 programov, v načinu programiranja pa lahko vpišemo lastne programe. Prostora je za 30 programov, vsak ima lahko do največ 60 korakov.

Da pa bi bilo upravljanje z napravo lažje, sem razvil tudi aplikacijo za pametne telefone Android. Na mikrokrmilnik naprave je priključen Bluetooth sprejemnik/oddajnik, ki komunicira s telefonom.

Poglavje 3

Mikrokrmilnik

Na mikrokrmilnik lahko gledamo kot na majhen računalnik, ki ima vse svoje sestavne dele vgrajene v eno samo integrirano vezje. Četudi se tega ne zavedamo, so nepogrešljivi del našega vsakdana. Večina naprav, ki jih uporabljamo za lajšanje vsakodnevnih tegob, pa naj bo to priprava hrane, pomivanje posode ali pa pranje perila, je krmiljena z enim ali večimi mikrokrmilniki.

3.1 Splošno

Skoraj vsak mikrokrmilnik je sestavljen iz sledečih elementov: centralne procesne enote (CPE), pomnilnika ter GPIO zatičev (GPIO, ang. General Purpose Input Output).

Ponavadi ima mikrokrmilnik dve vrsti pomnilnika, in sicer: ROM (ang. Read Only Memory) in RAM (ang. Random Access Memory). Med bralne pomnilnike spadajo: EPROM (ang. Erasable Programmable Read Only Memory), EEPROM (ang. Electrically Erasable Read Only Memory) ter Flash, ki se uporablja za shranjevanje programa.

3.2 Mikrokrmilnik ATmega64

ATmega64 je mikrokrmilnik ameriškega podjetja Atmel, ki ima sedež v zvezni državi Kaliforniji. Podjetje je bilo ustanovljeno leta 1984. Večino svojega dobička ustvari prav s prodajo mikrokrmilnikov. Poleg mikrokrmilnikov družine AVR, Atmel proizvaja še mikroprocesorje z arhitekturo ARM. Med ostale produkte podjetja spadajo še kartice za brezžično povezavo (Wi-Fi), ekrani na dotik, pomnilniški moduli itd.

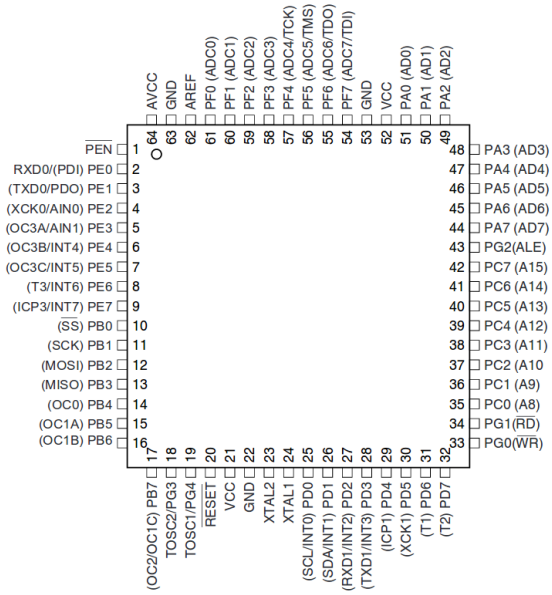
3.2.1 Lastnosti

ATmega64 je 8-bitni mikrokrmilnik tipa RISC (ang. Reduced Instruction Set Computer), kar pomeni, da je nabor strojnih ukazov dokaj majhen, le 130, in da jih večina traja enako število urinih period.

Razpolaga s 64-Kbajtnim programljivim pomnilnikom flash, ki se uporablja za shranjevanje programa. Vgrajen ima tudi 4-Kbajtni SRAM (Static random-access memory), ki služi kot glavni pomnilnik. Razpolaga tudi z 2-Kbajtnim EEPROMom, ki lahko ohrani svoje stanje tudi, ko ni napajanja in se uporablja za shranjevanje konfiguracij programa in podobno.

Med perifernimi funkcionalnostmi mikrokrmilnika najdemo dva 8-bitna ter dva 16-bitna števca, dva programljiva USART vmesnika in SPI vmesnik tipa gospodar/suženj, ki služi za programiranje mikrokrmilnika.

ATmega64 ima 64 zatičev, od tega jih je 56 GPIO. GPIO zatiči so združeni v 7 portov, v vsakem jih je po 8. Porti se imenujejo PORTA, PORTB, PORTC, PORTD, PORTE, PORTF ter PORTG. Vsak izmed njih ima po 3 registre. Istoimenski register služi nastavljanju vrednosti zatičev. Register DDRx (ang. Data Direction Register) zatiče na portu x nastavi kot vhodne ali izhodne, register PINx (ang. Port x Input Pins Address) pa služi branju trenutne vrednosti zatičev na portu x. Podrobno razporeditev zatičev na čipu najdemo na sliki 3.1. Nekateri izmed GPIO zatičev imajo tudi dodatne funkcionalnosti. Služijo lahko kot zunanje prekinitve ob spremembi stanja zatiča ali pa kot USART vmesniki.



Slika 3.1: Razporeditev zatičev na ATmega64

3.3 LED diode

LED diode in izhodi za žarnice so priključeni na PORTC. Za LED diode ni potrebno prav veliko konfiguracije. Vse zatiče PORTC je potrebno nastaviti kot izhod, kar se stori z vpisom samih enic v register DDRC. Ko je izhod zatiča, na katerega je vezana dioda, enak 0, dioda sveti, ko je izhod enak 1, je dioda ugasnjena.

3.4 Tipke

Tipke so priključene na PORTF. Zatiče PORTF je potrebo nastaviti kot vhodne. To storimo z zapisom ničel v register DDRF. Ob pritisku tipke dobimo na zatič logično 0, zato moramo za vseh 8 zatičev vklopiti notranje upore, ki zatič držijo na logični 1, dokler tipka ni pritisnjena. To storimo z vpisom enic v register PORTF.

Ker pa na vsak zatič prideta po dve tipki, potrebujemo še kontrolna

izhodna zatiča, v našem primeru sta to zatiča 0 ter 1 na PORTG. Ko je ničti postavljen na 1, je izbrana spodnja vrsta tipk, ko pa je prvi na 1, je izbrana zgornja vrsta tipk. Če sta oba zatiča hkrati postavljena na 1, nam to določa nedefinirano stanje.

Branje tipk oziroma zaznavanje pritiska na tipko je implementirano s časovnikom. Inicializacija časovnika:

```
void initTimer(){
        /* nastavi vrednost primerjevalnega registra */
        OCR1A = 1700;
        /* za primerjevalni register nastavi OCR1A */
        TCCR1B |= (1 << WGM12);
        /* omogoci prekinitev */
        TIMSK  |= (1 << OCIE1A);
        /* nastavi uro na (glavno uro) / 1024 in zacni */
        TCCR1B |= (1 << CS12) | (1 << CS10);
}
```

Ura časovnika je nastavljena na 16 MHz / 1024. Pri 15625 Hz časovnik doseže vrednost 1700 približno 9 krat na sekundo. Takrat se sproži prekinitev in pokliče se sledeči prekinitveno-servnisi program (PSP):

```
ISR(TIMER1_COMPA_vect){
        bit_clear(PORTG, _BV(1));
        bit_set(PORTG, _BV(0));
        if (mode != mode_prog) {
                if (PINF != 0xff){
                        cli();
                        doAction(PINF);
                        sei();
                        return;
                }
        } else {
                if (PINF != 0xff) {
```

```
                cli();
                doActionProg(PINF);
                sei();
                return;
            }
            bit_clear(PORTG, _BV(0));
            bit_set(PORTG, _BV(1));
            if (PINF != 0xff) {
                cli();
                doActionProg(~PINF);
                sei();
                return;
            }
        }
}
```

Prekinitveno-servisni program najprej preveri, v katerem načinu je naprava. Če ni v načinu programiranja, potem se vedno uporablja samo spodnja vrsta tipk. Ko pa je v načinu programiranja, pa najprej preveri spodnjo vrsto in nato še zgornjo vrsto tipk. Če zazna pritisk tipke, to je, ko niso vsi zatiči PORTF enaki 1, izključi prekinitve in pokliče funkcijo, ki izvede akcijo glede na pritisnjeno tipko.

3.5 LED ekran

LED ekran je priključen na zatiče PORTD, razen osvetlitve, ki se nahaja na PORTB. Natančnejšo porazdelitev zatičev si lahko ogledamo v tabeli 3.1.

LED ekran deluje v 4-bitnem načinu, kar pomeni, da sta za prenos enega znaka potrebni 2 pošiljanji. Po vsakem pošiljanju je potrebno ekranu sporočiti, da je pošiljanje končano, kar se stori s postavitvijo zatiča E (ang. enable) na ekranu za vsaj 5 milisekund. Pošiljati je mogoče ukaze in znake. Pri pošiljanju znaka mora biti zatič ekrana Rs (ang. reset) v visokem stanju, pri

Tabela 3.1: Razporeditev zatičev LED ekrana

Zatič ekrana	Zatič mikrokrmilnika
Osvetlitev	PORTB 7
Rs	PORTD 7
E	PORTD 6
DB4	PORTD 5
DB5	PORTD 4
DB6	PORTD 1
DB7	PORTD 0

ukazu pa v nizkem. Koda za krmiljenje LED ekrana:

```
void sendNibbleLCD(uint8_t data){
        PORTD &= ~(0x33); //pocisti podatkovne zatice ekrana
        if(data & _BV(4)) bit_set(PORTD, _BV(LCD4));
        if(data & _BV(5)) bit_set(PORTD, _BV(LCD5));
        if(data & _BV(6)) bit_set(PORTD, _BV(LCD6));
        if(data & _BV(7)) bit_set(PORTD, _BV(LCD7));
}

void ledPulse(){
        bit_set(PORTD, _BV(LCD_E));
        _delay_ms(5);
        bit_clear(PORTD, _BV(LCD_E));
}

void sendByteLED(uint8_t data){
        sendNibbleLCD(data);
        ledPulse();
        data <<= 4;
        sendNibbleLCD(data);
        ledPulse();
```

```
}

void charLCD(uint8_t ch){
        bit_set(PORTD, _BV(LCD_Rs));
        sendByteLED(ch);
}

void cmdLCD(uint8_t cmd){
        bit_clear(PORTD, _BV(LCD_Rs));
        sendByteLED(cmd);
}

void initLCD(){
        cmdLCD(0x33); // initialize
        cmdLCD(0x32); // nastavi 4 bitni vhod
        cmdLCD(0x28); // 2 vrstici po 18 znakov
        cmdLCD(0x0C); // onemogoci kazalec( 0c0e omogoci)
        cmdLCD(0x06); // smer kazalca
        cmdLCD(0x01); // pocisti ekran
        _delay_ms(100);
}

void cursorLCD(char pos){
        char cmd;
        if(pos < 0x10) cmd = 0x80;
        else cmd = 0xc0;
        cmd += (pos & 0x0f);
        cmdLCD(cmd);
}

void stringLCD(char* text){
```

```
        while(*text){
                charLCD(*text++);
        }
}
```

3.6 Bluetooth

Na mikrokrmilnik je priključen tudi Bluetooth oddajnik / sprejemnik, in sicer na zatiča 2 in 3 PORTD. Ta dva zatiča poleg GPIO služita tudi kot vhod in izhod za USART1. USART (ang. Universal Synchronous / Asynchronous Receiver / Transmitter) je naprava, ki jo sedaj vključujejo že skoraj vsi mikrokrmilniki in služi za prenos podatkov. Ponavadi se uporablja v povezavi s komunikacijskimi standardi. Tako je tudi v mojem primeru. USART uporabljam v povezavi s standardom RS-232.

Zatič 2 na PORTD predstavlja Rx zatič (sprejemni, ang. Receive) USART1 vmesnika na mikrokrmilniku, zatič 3 pa Tx zatič(oddajni, ang. Transmit). Zatič Rx mikrokrmilnika je vezan na zatič Tx na Bluetooth modulu, Tx mikrokrmilnika pa na Rx modula.

V mikrokrmilniku je potrebno nastaviti parametre, ki so potrebni za pošiljanje. Bluetooth modul ima privzeto hitrost prenosa (ang. baud rate) 9600, zato je potrebno na to vrednost nastaviti tudi hitrost prenosa USARTa v mikrokrmilniku. Mikrokrmilnik za hitrost prenosa uporablja vrednost registra UBBR (ang. USART Baud Rate Register). Vanj je potrebno vpisati vrednost izraza

$$\frac{f}{16BAUD} - 1,$$

kar je v našem primeru enako 76. Nastaviti je potrebno tudi število podatkovnih bitov in zaključnih bitov. Vse to se naredi v naslednji funkciji:

```
        void uart_init(void){
                /* hitrost prenosa */
                UBRR1H = 0;
```

```
        UBRR1L = 76;
        /* 8 pod. bitov, 1 kon. bit */
        UCSR1C=(1<<UCSZ11)|(1<<UCSZ10);
        /* omogocimo posiljanje ter prekinitve */
        UCSR1B=(1<<TXEN1)|(1<<RXEN1)|(1<<RXCIE1);
}
```

Protokol, ki se uporablja za dekodiranje, je preprost. Vedno se pošiljata / sprejemata po dva bajta. Prvi bajt je kontrolni, ki določa akcijo, drugi bajt pa je parameter akcije.

Poglavje 4

Android aplikacija za nadzor naprave

4.1 Android

Android je operacijski sistem namenjen predvsem napravam z ekranom na dotik, kot so mobilni telefoni ter tablični računalniki. Razvija ga ameriški spletni gigant Google.

Operacijski sistem Android temelji na Linuxovem jedru in je napisan v programskih jezikih C in Java. Za pisanje novih aplikacij ponuja SDK (ang. Software development kit). SDK vsebuje razhroščevalnik (ang. debugger), knjižnice, emulator Android naprav, dokumentacijo ter vzorce kode.

Android aplikacije so podobne spletnim aplikacijam. Večina spletnih aplikacij je sestavljenih iz več podstrani, ki skupaj tvorijo celoto. Podobno je tudi pri Android aplikacijah, le da se tukaj podstrani imenujejo aktivnosti (ang. activity). Vsaka aktivnost ima tudi svojo postavitev (ang. layout).

4.2 Aplikacija za nadzor naprave

Aplikacija je sestavljena iz treh aktivnosti: glavne aktivnosti, aktivnosti za nadzor in aktivnosti za pisanje programov.

4.2.1 Glavna aktivnost

Glavna aktivnost je prva, ki se prikaže ob zagonu aplikacije. Ima dva gumba za vstop v eno izmed ostalih aktivnosti.

Ob pritisku na gumb za vstop v aktivnost za pisanje programov se aktivnost odpre takoj, pri pritisku na gumb za vstop v aktivnost za nadzor pa se pred tem izvede še nekaj akcij. Najprej se preveri, če je Bluetooth vmesnik na telefonu vključen. Če ni, uporabniku ponudi opcijo za vklop le-tega, drugače nadaljuje. V naslednjem koraku se poskuša aplikacija povezati na napravo za svetlobne učinke. Če ji to ne uspe, izpiše napako, ob uspehu pa se odpre aktivnost za nadzor.

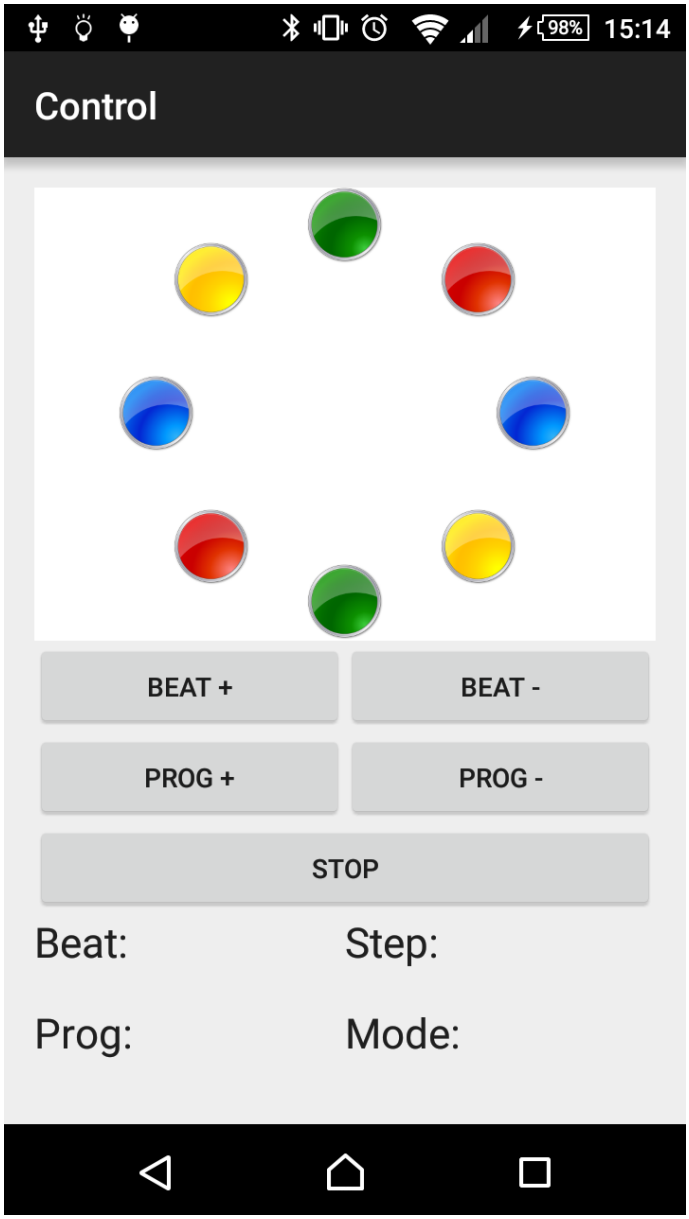
4.2.2 Aktivnost za nadzor naprave

Postavitev aktivnosti je sestavljena iz dveh delov. Polovico ekrana zasedajo slike 8 barvnih luči, razporejenih v krog, ki je enaka raporeditvi na napravi. V drugi polovici pa se nahajajo gumbi ter labele. Izgled prikazuje slika 4.1.

Ob vsaki spremembi koraka na napravi se preko Bluetooth povezave pošljeta dva bajta. Prvi bajt nam sporoči, da je prišlo do spremembe prižganih / ugasnjenih luči in da moramo spremeniti slike luči v naši aktivnosti. Drugi bajt pa nam pove, katere luči so prižgane in katere ugasnjene. Ker je luči ravno 8, tako kot število bitov v bajtu, lahko na bite gledamo kot na indikator. Če je prvi bit v bajtu 1, pomeni da je prva luč ugasnjena, če pa je 0, pa pomeni, da je prižgana. Enako velja za vse ostale bite. Glede na bit, ki nam predstavlja luč, nastavimo sliko pogleda na prižgano ali ugasnjeno.

Ob pritisku na gumb se pošljeta dva bajta na napravo. Prvi bajt določa akcijo, ki se na napravi izvede, drugi bajt pa parameter akcije. Na ta način je mogoče nastaviti hitrost utripanja lučk, program, ki se izvaja, in zaustavitev oziroma zagon programa.

Labele prikazujejo stanje naprave. Ob vsaki spremembi stanja naprava pošlje informacijo o novem stanju, ki se zapiše v labelo.



Slika 4.1: Aktivnost za nadzor naprave

4.2.3 Aktivnost za pisanje programov

Za razliko od aktivnosti za nadzor je ta aktivnost dostopna tudi brez povezave Bluetooth do naprave. Postavitev je sestavljena iz dveh delov. Prvi del je enak, kot v aktivnosti za nadzor, v drugem delu pa se nahaja šest gumbov.

Aktivnost je namenjena pregledovanju, pisanju in spreminjanju programov. S pritiskom na gumb se premikamo po korakih programa. Slike luči nam prikazujejo njihovo stanje v trenutnem koraku. S pritiskom na sliko luči se stanje zamenja iz aktivnega v neaktivno ali obratno. Ob pritisku na gumb za shranjevanje se spremembe programa zapišejo naravnost v napravo za svetlobne učinke. Če pa aplikacija ni povezana z napravo, se program zapiše v sistem Android in se avtomatično prenese na napravo ob prvi povezavi.

4.2.4 Bluetooth

Branje toka podatkov (ang. data stream) je blokirana akcija, kar pomeni, da se med branjem ne more izvajati nobena druga akcija. Če bi to počeli v isti niti, kjer se izvaja grafični uporabniški vmesnik (GUI), med branjem vmesnik ne bi bil odziven. Ta problem rešimo tako, da branje toka podatkov prestavimo v novo nit.

S tem smo rešili problem neodzivajočega uporabniškega vmesnika, vendar tako kot vedno v računalništvu, z rešitvijo enega problema ustvarimo nov problem. Preko vmesnika Bluetooth dobivamo podatke, ki spreminjajo izgled uporabniškega vmesnika, vendar uporabniški vmesnik lahko spreminja le glavna nit. Pojavi se vprašanje, kako podatke iz ene niti prenesemo v drugo nit. Sistem Android za ta problem ponuja elegantno rešitev. V istem javanskem razredu, ki definira aktivnost, se definira tudi obdelovalnik (ang. handler). Obdelovalniku je potrebno podati nit, ki skrbi za grafični vmesnik.

Obdelovalnik dobi sporočilo od niti, ki bere iz toka podatkov. Sporočilo vsebuje polje bajtov. Vsak bajt iz polja zapišemo v vrsto (ang. queue). Vrsta je implementirana z javanskim razredom LinkedList.

Koda obdelovalnika v aktivnosti za nadzor:

```
public Handler handler = new Handler(Looper.getMainLooper()) {
 public void handleMessage(Message msg) {
  Bundle bundle = msg.getData();
   byte[] bytes = bundle.getByteArray("msg");
   int size = bundle.getInt("size");
   for (int i=0; i<size; i++) {
        queue.addLast((int)bytes[i]);
   }
   while (queue.size() > 1) {
    int[] temp = {queue.pop(), queue.pop()};
    String status = decodeBytes(temp);

    String[] split = status.split(" ");
    switch (split[0]) {
     case "beat":
      labelBeat.setText("Beat: " + split[1]);
      break;
     case "prog":
      labelProg.setText("Prog: " + split[1]);
      break;
     case "step":
      labelStep.setText("Step: " + split[1]);
      break;
     case "running":
      stop_start.setText((split[1].equals("0")) ? "Start" : "Stop");
      break;
     case "mode":
      labelMode.setText("Mode: " + split[1]);
      break;
     case "lights":
      setLights(Integer.parseInt(split[1]));
```

```
        break;
       default:
        Toast.makeText(getApplicationContext(), "Error",
         Toast.LENGTH_SHORT).show();
     }
    }
   }
  private void setLights(int lights) {
   for (int i=lightsArray.length-1; i>=0; i--) {
    if (((lights >> i) & 1) == 0)
     lightsArray[i].setImageResource(drawablesArray[(i % 4) * 2]);
    else
     lightsArray[i].setImageResource(drawablesArray[((i % 4) * 2) + 1]);
   }
  }
};
```

V niti, kjer beremo tok podatkov, ob inicializaciji nastavimo obdelovalnik na obdelovanik nadzorne aktivnosti. V zanki samo beremo in pošiljamo prebrane bajte v obdelavo obdelovalniku.

Koda branja iz vhodnega toka podatkov:

```
public void run(){
 int b;
 int bytes;
 byte[] buffer = new byte[1024];
 Bundle bundle = new Bundle();
 while (handler == null) {}
 while (true){
  String result = null;
   try {
    bytes = in.read(buffer);
    Message msg = handler.obtainMessage();
```

```
      bundle.putByteArray("msg", buffer);
      bundle.putInt("size", bytes);
      msg.setData(bundle);
      handler.sendMessage(msg);
     } catch (IOException e) {
      e.printStackTrace();
      break;
    }
   }
}
```

Poglavje 5

Zaključek

V sklopu diplomske naloge sem razvil program za mikrokrmilnik ATmega64, ki krmili napravo za svetlobne efekte. Program omogoča spreminjanje hitrosti utripanja lučk, način utripanja, pisanje novih programov in pošiljanje ter sprejemanje ukazov preko vmesnika Bluetooth. Poleg programa za mikrokrmilnik je bila razvita tudi Android aplikacija, ki omogoča nadzor naprave ter pisanje novih programov utripanja. Upam, da bo naprava prisotna na čimveč praznovanjih in da se bodo ljudje ob utripanju lučk zabavali.

Izpolnil sem vse cilje, ki sem si jih zadal ob pričetku izdelave. S končnim izdelkom sem zadovoljen. Edina težava, ki sem jo opazil, je velika stopnja napak pri pošiljanju iz naprave na telefon. Pričakovana stopnja napak je okoli 0,2%, dejanska stopnja pa je okoli 3-5%. Problem sem poskušal rešiti s spreminjanjem parametrov Bluetooth komunikacije in s spreminjanjem kode za sprejemanje v aplikaciji za telefon. Oba poskusa sta bila neuspešna. Stopnjo napak mi je v obeh primerih uspelo povečati, a nikoli zmanjšati.

Obstaja precej funkcionalnosti, ki bi se jih lahko dodalo. Med drugim tudi možnost priklopa večih stroboskopov, posebni programi, namenjeni samo utripanju stroboskopa, program za krmiljenje naprave za računalnik itd.

Ob programiranju sem se zabaval in hkrati naučil veliko novih spretnosti. S programiranjem mikrokrmilnika v jeziku C še nisem imel izkušenj, prvič pa sem se lotil tudi izdelave aplikacije za operacijski sistem Android.

Skratka, možnosti za izboljšavo naprave za svetlobne efekte je neomejeno. Verjetnost, da bom katero izmed njih tudi implementiral, je kar velika.

Literatura

[1] ATmega64/L datasheet. [Online]. Dosegljivo: http://www.atmel.com/Images/Atmel-2490-8-bit-AVR-Microcontroller-ATmega64-L_datasheet.pdf. [Dostopano 5.9.2015].

[2] Elliot Williams. Make: AVR Programming. Addison-Wesley, 2014.

[3] Develop Apps - Android Developers. [Online]. Dosegljivo: http://developer.android.com/develop/index.html.[Dostopano 7.9.2015].

[4] Nicolas Gramlich. andbook! Android Programming.[Online]. Dosegljivo: http://andbook.anddev.org/files/andbook.pdf. [Dostopano 22.8.2015] .

[5] GNU General Public Licence. [Online]. Dosegljivo: https://www.gnu.org/copyleft/gpl.html. [Dostopano 20. 9. 2014].