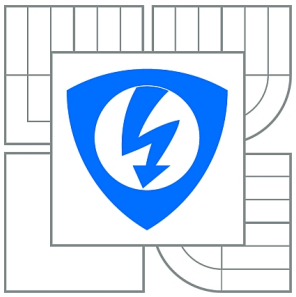
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF BIOMEDICAL ENGINEERING

INDIKACE POZICE MIKROTITRAČNÍ DESTIČKY

INDICATION OF THE MICROPLATES POSITION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

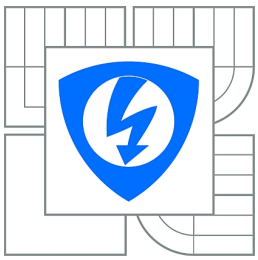
AUTOR PRÁCE — GABRIELA PAPAJOVÁ

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SEKORA

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav biomedicínského inženýrství

Bakalářská práce

bakalářský studijní obor
Biomedicínská technika a bioinformatika

Studentka: Gabriela Papajová ***ID:*** 154645
Ročník: 3 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Indikace pozice mikrotitrační destičky

POKYNY PRO VYPRACOVÁNÍ:

1) Seznamte se s provedením mikrotitračních destiček pro pipetování. 2) Navrhněte systém pro indikaci pozice při pipetování. 3) Navrhněte softwarový přípravek (GUI) pro výběr pozice a pořadí pipetování. 4) Navrhněte hardwarový přípravek, který bude přijímat informace o pozici z navrženého GUI a tyto zobrazí pod mikrotitrační destičkou. 5) Realizujte softwarovou i hardwarovou část a ověřte správnost funkce.

DOPORUČENÁ LITERATURA:

[1] BARRETT, Steven F. Embedded systems design with the Atmel AVR microcontroller. San Rafael?: Morgan, 2010, xvi, 300 s. Synthesis lectures on digital circuits and systems. ISBN 978-1-60845-127-2.
[2] IBRAHIM, Dogan. Advanced PIC microcontroller projects in C: from USB to ZIGBEE with the 18F series. Amsterdam: Newnes/Elsevier, 2008, xiv, 544 s. ISBN 978-0-7506-8611-2.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 29.5.2015

Vedoucí práce: Ing. Jiří Sekora
Konzultanti bakalářské práce:

prof. Ing. Ivo Provazník, Ph.D.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce se zabývá indikací pozice mikrotitrační destičky pro usnadnění orientace při pipetování látek. To zahrnuje návrh hardwaru, blokový diagram a obslužný software. Další důležitá část se zabývá softwarovým designem grafického uživatelského rozhraní a řízením mikrokontroléru. Finální produkt je hardwarový přípravek, který používá 96 dvoubarevných LED k indikaci pozice mikrotitrační destičky. Součástí práce je i uživatelská příručka pro snadnou orientaci v programu. Tento přípravek bude používán na lékařské fakultě Masarykovy univerzity v Brně.

KLÍČOVÁ SLOVA

Indikace pozice, mikrokontrolér, mikrotitrační destička, LED pole, Arduino

ABSTRACT

This bachelor thesis deals with designing and constructing a device which can indicate position of the microplates. This includes hardware design, block chart and utility software. Next important section deals with software design of graphical user interfaces and how to control the microcontroller. The final product is a hardware tool that uses array of 96 two-color LED to indicate a position in the microplate. This work also contains a user guide for simple orientation in the program. This product will be operated at Faculty of Medicine, Masaryk University in Brno.

KEYWORDS

Indication of position, microcontroller, microplate, LED array, Arduino

PAPAJOVÁ, G. *Indikace pozice mikrotitrační destičky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 38 s. Vedoucí bakalářské práce Ing. Jiří Sekora.

PROHLÁŠENÍ

Prohlašuji, že jsem svoji bakalářskou práci na téma Indikace pozice mikrotitrační destičky vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědoma následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

V Brně dne 25. května 2015 .....

podpis autorky

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Jiřímu Sekorovi za metodické vedení, připomínky a užitečné rady při vytváření mé bakalářské práce. Děkuji také MUDr. Michalovi Jurajdovi Ph.D. za důležité podněty, které pomohly ke zkvalitnění této práce.

V Brně dne 25. května 2015 .....

podpis autorky

Obsah

Seznam obrázků

Úvod

Tématem bakalářské práce je indikace pozice mikrotitrační destičky pro usnadnění orientace při pipetování látek.

Bakalářská práce je rozdělena na softwarovou a hardwarovou část. Softwarová část je reprezentována grafickým uživatelským rozhraním v jazyce Java a softwarem v mikrokontroléru pro ovládání matice LED. Hardwarová část je tvořena 96 dvoubarevnými diodami ovládanými platformou Arduino s mikrokontrolérem ATmega328.

Tento přípravek by měl usnadnit a zkvalitnit práci laboratorního personálu při pipetování do mikrotitrační destičky tím, že zabrání pracovníkovi udělat nevědomou chybu. Laboratorní personál má vizuálně kontrolu světlem LED pod mikrotitrační destičkou a tím, že si předem samotný titrační plán vytvoří v obslužném programu. Přípravek je „stand-alone“ zařízení, to znamená, že pracuje bez použití osobního počítače. Další výhodou je také možnost uložení pozic napipetovaných vzorků a jejich zpětné otevření kdykoliv je potřeba.

Grafické uživatelské rozhraní i hardware je vytvořen uživatelsky co nejvíce přístupný a jednoduchý. Hardware je napájen přes externí adaptér ze sítě a jeho mikrokontrolér má nahraný program, který čeká na vložení µSD karty s vytvořeným titračním plánem. Dále samotný hardware má tlačítko, díky kterému může obsluha postupně rozsvěcovat diody bez obsluhy počítače dle vybraného režimu a barvy. Součástí bakalářské práce je i uživatelská příručka pro lepší orientaci v obslužném programu.

1 Analýza požadavků zařízení

Tento přípravek má pomoci pracovníkovi s indikací pozice na mikrotitrační destičce pomocí LED umístěných pod ní. Vzhledem k tomu, že se většinou pipetuje 3 a více různých látek, tak jsou vybrány dvoubarevné LED, kde je na výběr z červené, zelené a oranžové barvy.

Samotná mikrotitrační destička má rozměry 86 x 128 mm a vyplňuje ji 96 jamek (8 řad po 12 sloupcích). Tyto destičky se využívají k mnoha metodám v medicíně, chemii i biologii. K seznámení se základními metodami, při kterých se tyto destičky používají, slouží následující odkaz na literaturu [16].

Z dalších požadavků se také nesmí opomenout režimy rozsvěcování LED. Požadováno bylo rozsvěcování bod po bodu, triplety (tj. tři po sobě jdoucí body, které jsou rozsvíceny najednou), vše najednou, po řádcích a po sloupcích.

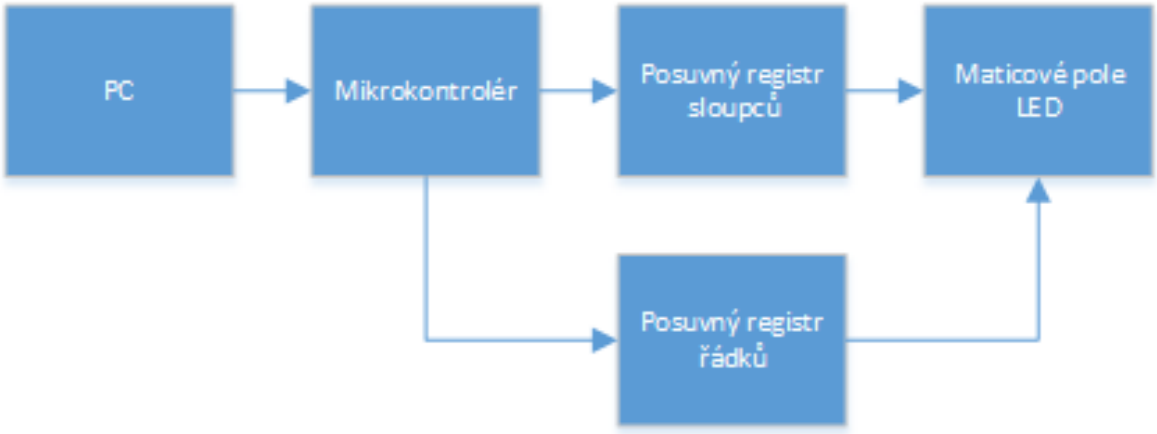
Další požadavek byl, aby zařízení fungovalo bez obsluhy počítače, proto zde je využíváno μSD karty, která se vkládá přímo do přípravku.

Dále je možnost uložení vytvořeného titračního plánu a jeho zpětné otevření v obslužném programu.

2 Návrh a popis řešení

2.1 Blokové schéma

Rozpis jednotlivých bloků a jejich význam v obvodu je rozepsán v následující kapitole. Výsledné blokové schéma je na obrázku 2.1.



Obr. 2.1: Blokové schéma obvodu

2.2 Popis blokového schématu

Osobní počítač

První částí blokového schématu je blok PC, který představuje počítač, na kterém bude spuštěný obslužný program pro tvorbu pipetovacího plánu. Tato funkcionalita je podrobněji popsána v kapitole 4.

Mikrokontrolér

Druhou částí schématu je mikrokontrolér ATmega328, který bude na základě vytvořeného pipetovacího plánu uloženého na micro SD kartě (dále µSD), příslušné LED rozsvěcovat. Mikrokontrolér ATmega328 je součástí otevřené elektronické platformy Arduino [8]. Detaily a vlastnosti platformy Arduino jsou popsány v technické dokumentaci [1].

Posuvný registr sloupcový

Třetí částí schématu jsou tři 8bitové posuvné registry pro aktivaci sloupců. Aby byl zvýšen komfort obsluhy při pipetování různých vzorků, je umožněno využít tří barev diod – zelené, červené a oranžové. Prvních 12 bitů registrů aktivuje katody LED pro

červenou barvu a dalších 12 bitů aktivuje druhé katody LED pro zelenou barvu. Pokud dojde k aktivaci obou katod jedné diody zároveň, tak výsledná barva bude kombinací obou, tj. oranžová. Vybraným typem posuvného registru je 74LS595N, který je detailně popsán v technické dokumentaci [7]. Příklad použití a zapojení registru bylo převzato z knihy *C programming for Arduino* [4].

Posuvný registr řádkový

Čtvrtou částí je jeden 8bitový posuvný registr, který bude postupně aktivovat řádky matice jeden po druhém. Řádky musí být aktivovány velmi rychle a to alespoň s frekvencí větší než 100 Hz (tj. 8,3 Hz na řádek), čímž se s ohledem na setrvačnost LED bude lidskému oku zdát, že je obraz na diodové matici stabilní. Vybraným typem posuvného registru je opět 74LS595N.

Maticové pole LED

Poslední částí blokového schématu je maticové pole 96 dvoubarevných LED v provedení SMD, které je umístěné přesně pod jamkami mikrotitrační destičky. Každá dioda má dvě katody, z nichž jedna aktivuje zelenou barvu a druhá barvu červenou. Při aktivaci obou katod dojde ke smíchání barev a tím výslednému rozsvícení diody v oranžové barvě. Příklad řízení LED modulu byl převzat z následující literatury [3].

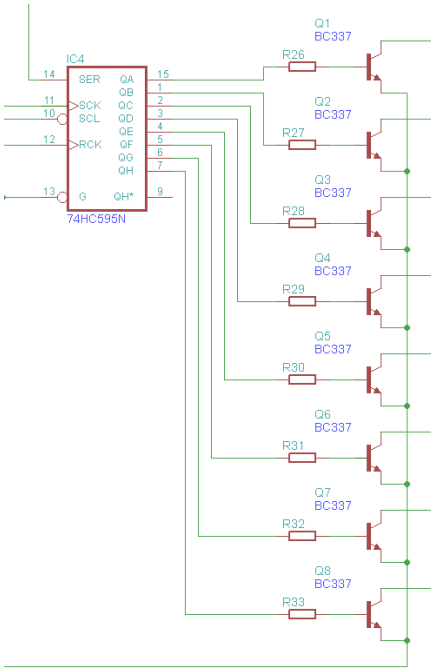
2.3 Elektrické schéma zapojení

Srdcem celého obvodu je deska s platformou Arduino a mikrokontrolérem ATmega328, který bude přes své vstupně/výstupní rozhraní řídit celý obvod. Platforma Arduino [6] byla vybrána z důvodu snadné implementace funkcí, které jsou využity při řešení. Samotný obvod se bude skládat z 96 dvoubarevných LED, čtyř 8bitových posuvných registrů, tlačítka, 33 rezistorů a 8 bipolárních tranzistorů typu NPN.

Výslednou funkcí obvodu bude rozsvěcování požadovaných LED v matici dle zvolené barvy a režimu. K tomu, aby bylo tohoto požadavku dosaženo, musí být LED uspořádány do matice o velikosti 8x12.

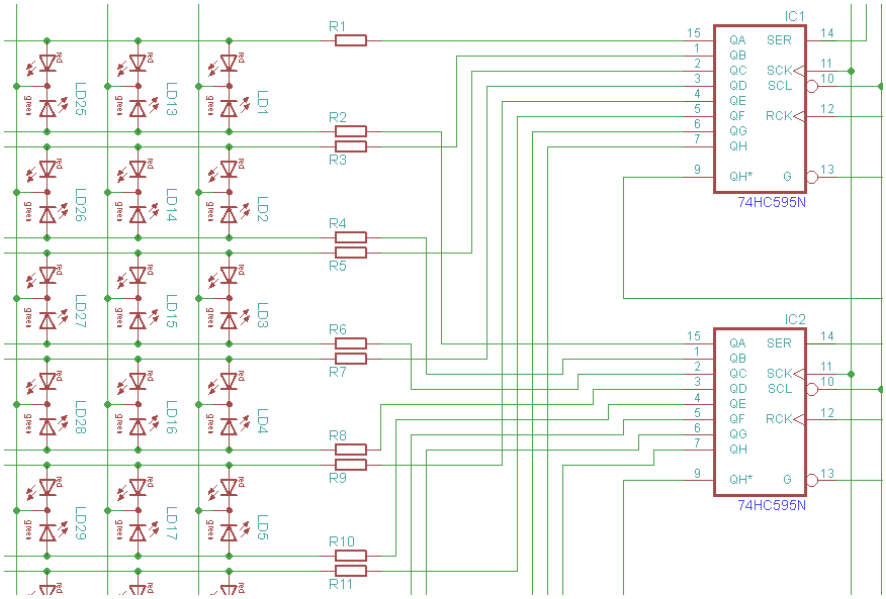
V každém řádku matice dojde k propojení anody LED a každý takto vytvořený řádek je připojen přes tranzistor a odpor k výstupu 8bitového posuvného registru, viz obr. 2.2. Tento posuvný registr odesílá na své výstupy jedničku, kterou „posouvá“ postupně od prvního pinu k poslednímu s frekvencí, kterou mu nastavíme. Tím dosahujeme postupného aktivování řádků. Na tento registr navazuje pole osmi bipolárních NPN tranzistorů BC337-40 [13], které se starají o odvod proudu z elektrického obvodu a jeho uzemnění. Toto přidání tranzistorů mezi registr a diody bylo zvoleno i z toho důvodu, aby se výstupy registru ochránily vůči nežádoucímu

vstupnímu proudu, jelikož tento typ registru nemá absorpční vlastnosti a hlavně, aby nebyly proudově zatíženy jeho výstupy.



Obr. 2.2: Část obvodu s posuvným registrem a NPN tranzistory

Katody pro červené zbarvení LED se taktéž propojí a toto propojení vytvoří pomyslný sloupec. Pomyslný proto, že jsou ještě katody pro zelené zbarvení LED, které je potřeba také propojit. Příklad části obvodu je na obr. 2.3. Na tomto obrázku lze vidět, že všechny výstupy registru vpravo nahoře jsou připojeny na katody aktivující červenou barvu a všechny výstupy z druhého registru jsou připojeny na katody aktivující zelenou barvu.

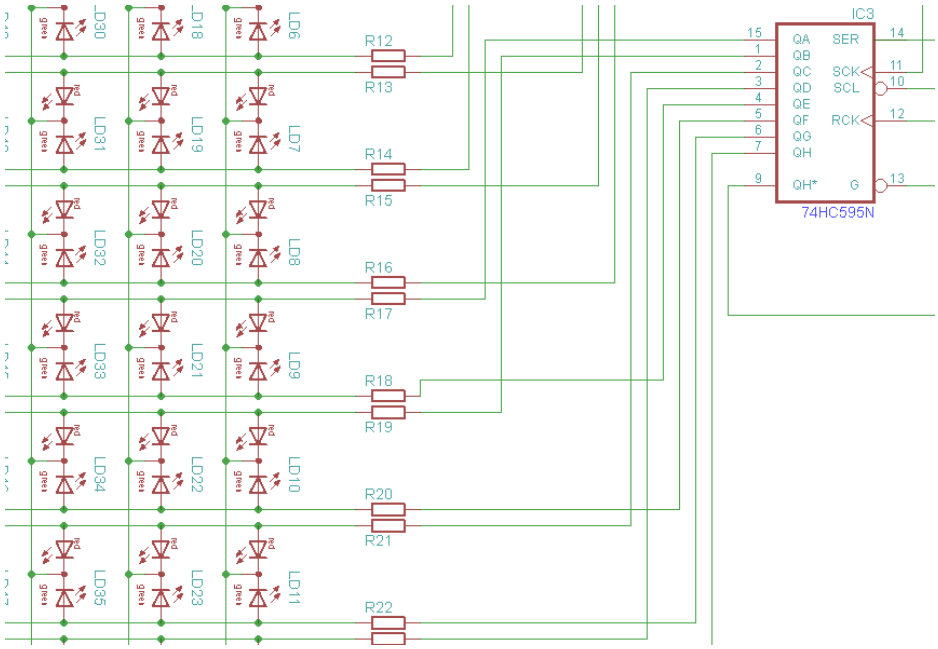


Obr. 2.3: Registry pro aktivaci katod LED

Následně všech dvanáct „červených sloupců“ bude připojeno přes odpor k výstupům prvního 8bitového posuvného registru a k 4 výstupům z třetího 8bitového registru.

Na dalších 4 výstupech třetího registru a na všech 8 výstupech druhého registru je opět přes odpor připojeno všech dvanáct „zelených sloupců“. Tyto tři registry se tedy budou starat o aktivaci barvy. Zapojení registrů bylo převzato z článku na oficiálních stránkách Arduina [14]. Výsledné schéma zapojení je v příloze A.

Poslední ze tří registrů pro aktivaci sloupců bude z prvních 4 výstupů aktivovat katody jedné barvy a z dalších 4 výstupů katody druhé barvy. Názorná ukázka zapojení registru pro aktivaci obou katod je na obr. 2.4.



Obr. 2.4: Posuvný registr pro aktivaci katod obou barev

Hodnoty odporů jsou vypočteny jako:

$$R = \frac{Uz - Ud}{Id} \tag{1}$$

kde U_Z je napětí zdroje, U_D napětí na diodě a I_D je proud tekoucí diodou.

Napětí zdroje U_Z = 5 V, napětí na diodě U_D = 2 V a proud tekoucí diodou I_D = 10 mA [12]. Po dosazení do vzorce (1) vyšla hodnota odporu R = 300 Ω. Nejbližší hodnota odporu je 330 Ω, která bude použita v obvodu.

Dále se musí vypočítat hodnoty odporů před tranzistory. Nejprve se musí spočítat proud tekoucí bází tranzistoru, k tomu slouží následující vzorec:

$$\mathrm{Ib} = \frac{\mathrm{Ice}}{\beta} \tag{2}$$

kde I_{ce} je proud tekoucí částí kolektor-emitor a β- proudový zesilovací činitel.

V technické dokumentaci je uveden proud tekoucí částí koletor-emitor I_{ce} = 20 mA a β = 250. Po dosazení do vzorce (2) je hodnota spínacího proudu I_b = 80 μA.

$$\mathrm{R} = \frac{Uz - Ube}{Ib} \tag{3}$$

kde U_z je napětí na zdroji, U_{be}- napětí na části kolektor-emitor a I_b spínací proud tekoucí bází.

K výpočtu odporů před tranzistory použijeme vzorec (3), do něj se dosadí U_z= 5 V, U_{be} = 0,7 V a I_b= 80 μA. Vstupní část tranzistoru musí mít velký odpor, proto vyšla hodnota R_t = 53 750 Ω, nejblíže tomu je hodnota reálného odporu R = 56 kΩ, který bude použitý v obvodu.

2.4 Řídící software pro mikrokontrolér

Mikrokontrolér ATmega328 v platformě Arduino bude řízen programem, který je sestavený v jazyce C a je stabilně uložen v RAM mikrokontroléru.

Standartní Arduino program se skládá ze dvou hlavních částí. Jsou jimi metody *setup()* a *loop()* [9]. V metodě *setup()* se provádí inicializace programu. Nastavují se zde hlavně vstupně-výstupní piny mikrokontroléru a případně další pomocné proměnné. V metodě *loop()* je hlavní běh programu. Tato metoda běží v nekonečném cyklu.

V našem případě v metodě *setup()* nastavíme tři výstupní piny. Pin 5 pro latch, pin 4 pro hodinový signál a pin 7 pro data posuvných registrů. Vstupní pin 8 je pouze jeden pro tlačítko. Nastavení pinů je dokumentováno v následujícím kódu:

```
// nastav piny
 pinMode(latchPin, OUTPUT);
 pinMode(clockPin, OUTPUT);
 pinMode(dataPin, OUTPUT);
 pinMode(buttonPin, INPUT);
```

Pro μSD kartu není potřeba inicializovat speciální piny, jsou již nativně vyvedeny na piny 11 (MOSI), 12 (MISO) a 13 (CLK). Dále si zde nastavíme pole hodnot jednotlivých řádků. Každý prvek pole obsahuje decimální hodnotu, která reprezentuje jeden z výstupů posuvného registru pro řádky:

```
// pole hodnot radku
 radky[0] = 1;   // 00000001b
```

```
  radky[1] = 2;   // 00000010b
  radky[2] = 4;   // 00000100b
  radky[3] = 8;   // 00001000b
  radky[4] = 16;  // 00010000b
  radky[5] = 32;  // 00100000b
  radky[6] = 64;  // 01000000b
  radky[7] = 128; // 10000000b
```

V této metodě také načítáme soubor z SD karty. Popis funkcí SD knihovny je popsán v následujícím odkazu [10]. Pokud je soubor „matice.txt“ nalezen, načteme z něho režim zobrazování, matici LED a případně pole pořadí rozsvěcování diod, pokud je pro daný režim relevantní. Běh programu vidíme v následujícím kódu:

```
// re-open the file for reading:
  soubor = SD.open("matice.txt");
  if (soubor) {
    Serial.println("Nacten soubor matice.txt:");

    // dvourozmerne char-pole pro cislo poradi
    char cc[2];
    // cislo poradi led diody
    int cisloLed;
    // cislice ukazujici na pozici dvourozmerneho char-pole
    int cislice = 0;

    // cti ze souboru dokud to jde
    while (soubor.available()) {
       // pokud bylo dosazeno vsech radku pri cteni matice,
       // prejdi ke cteni poradi
      if (radek == radkuCelkem)
      {
        cc[cislice] = soubor.read();
        // dalsi radek?
        if (cc[cislice] == '\n')
        {
          // vynuluj pozici cislice
          cislice = 0;
          // preved char-pole na cislo
          cisloLed = atoi(cc); //atol()convert string to long
          Serial.println(cisloLed);
          // pokud je nactena nejaka hodnota
          //if ( cisloLed != 0 )
          //{
```

```
            // zapis cislo do pole poradi
            poradi[celkemPoradi] = cisloLed;
            celkemPoradi++;
            //}
          }
          else
          {
            cislice++;
          }
        }
        else
        {
          char c = soubor.read();
          // je nactena led dioda?
          if (c == '0' || c == '1' || c == '2' || c == '3')
          {
            matice[sloupec][radek] = c;
            sloupec++;
            if (sloupec > sloupcuCelkem - 1)
            {
              sloupec = 0;
              radek++;
              if (radek == radkuCelkem)
              {
                // dummy cteni znaku konce radku
                c = soubor.read();
                c = soubor.read();
              }
            }
          }
          else
          {
            // je nacteny rezim?
            if (c == 'B' || c == 'T' || c == 'S' || c ==
'R' ||  c == 'V')
            {
              rezim = c;
            }
            else
            {
              if (c == '\n')
              {
                // odradkovani
```

```
          }
        }
      }
    }
  }
  // zavri soubor
  soubor.close();
```

Zde je část programu ošetřující, že není soubor na SD kartě:

```
} else {
  // nepovedlo se otevrit soubor
  Serial.println("soubor se nepovedlo otevrit");
}
```

V metodě *loop()* rozsvěcujeme diody podle načteného režimu. K tomu slouží čtyři pomocné metody, jejichž název napovídá jejich funkčnost – *rozsvitSloupecDiod(sloupec), rozsvitRadekDiod(radek), rozsvitVybraneDiody(sloupec, radek)* a *rozsvitVsechnyDiody()*. Režim „V“ spustí metodu pro rozsvícení všech diod. Režim „B“ spustí metodu pro rozsvícení vybrané diody v pořadí. Režim „T“ spustí metodu pro rozsvícení vybrané diody v cyklu, celkem třikrát. Režim „S“ spustí metodu pro rozsvícení celého sloupce. A nakonec režim „R“ spustí metodu pro rozsvícení celého řádku.

```
// rozsvit vsechno
  if (rezim == 'V')
  {
    rozsvitVsechnyDiody( );
  }
  // rozsvecuj bod po bodu
  if (rezim == 'B')
  {
    sloupec = poradi[aktualniPoradi] % sloupcuCelkem;
    radek   = poradi[aktualniPoradi] / radkuCelkem;
    rozsvitVybraneDiody(sloupec, radek);
  }
  // rozsvit triplet
  if (rezim == 'T')
  {
    for (int i = 0; i < 3; i++)
    {
      sloupec = poradi[aktualniPoradi + i] % sloupcuCelkem;
```

```
      radek   = poradi[aktualniPoradi + i] / radkuCelkem;
      rozsvitVybraneDiody(sloupec, radek);
    }
  }
  // rozsvit sloupec
  if (rezim == 'S')
  {
    rozsvitSloupecDiod(aktualniSloupec);
  }
  // rozsvit radek
  if (rezim == 'R')
  {
    rozsvitRadekDiod(aktualniRadek);
  }
```

Metoda *rozsvitRadekDiod(radek)* si načte hodnotu z pole řádků. Nastaví všechny výstupy registru pro červenou barvu do logické jedničky a výstupy registru pro zelenou barvu do logické nuly, aby diody svítily pouze červeně. Pak nastaví latch pin do logické nuly, aby se povolil zápis do registrů, a odešle hodnoty do registrů. Poté nastaví latch pin zpět do logické jedničky.

```
void rozsvitRadekDiod(int radek) {

  // nacti hodnotu radku
  registrRadek = radky[radek];

  registrCervena = 255;
  registrZelena = 0;

  // zapis hodnoty do registru
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, registrRadek);
  shiftOut(dataPin, clockPin, MSBFIRST, registrZelena);
  shiftOut(dataPin, clockPin, MSBFIRST, registrCervena);
  digitalWrite(latchPin, HIGH);

  // zpozdeni v us
  delayMicroseconds(2000);
}
```

Metoda *rozsvitSloupecDiod(sloupec)* nastaví všechny výstupy registru řádků do logické jedničky, aby svítily všechny řádky ve stejný okamžik. Registru pro červenou

barvu se nastaví hodnota podle vstupního parametru aktuálního sloupce. Registr pro zelenou barvu bude opět neaktivní.

```
void rozsvitSloupecDiod(int sloupec) {

  // projed kazdy radek

  // nacti hodnotu radku
  registrRadek = 255;

  // cervena
  registrCervena = pow(2, sloupec);
  registrZelena = 0;

  // zapis hodnoty do registru
  digitalWrite(latchPin, LOW);
  shiftOut(dataPin, clockPin, MSBFIRST, registrRadek);
  shiftOut(dataPin, clockPin, MSBFIRST, registrZelena);
  shiftOut(dataPin, clockPin, MSBFIRST, registrCervena);
  digitalWrite(latchPin, HIGH);

  // zpozdeni v us
  delayMicroseconds(2000);
}
```

Metoda *rozsvitVybraneDiody(sloupec, radek)* nastaví výstup registru řádků podle vstupního parametru požadovaného řádku a načte si z pole matice látku pro požadovaný sloupec a řádek. Pokud je hodnota látky „1“, tedy červená, nastaví se výstup registru pro červenou barvu podle vstupního parametru požadovaného sloupce. Registr zelené barvy bude neaktivní. Pokud je hodnota látky „2“, tedy zelená, nastaví se výstup registru pro zelenou barvu podle vstupního parametru požadovaného sloupce a registr červené barvy bude neaktivní. V případě látky „3“, tedy oranžové, budou nastaveny výstupy obou registrů podle vstupního parametru požadovaného sloupce. Pokud je látka „0“, budou oba registry neaktivní.

```
void rozsvitVybraneDiody(int sloupec, int radek) {

  // typ latky
  char latka;

  // nacti hodnotu radku
```

```
registrRadek = radky[radek];

registrZelena  = 0;
registrCervena = 0;

// nacti hodnotu latky z matice
latka = matice[sloupec][radek];

// cervena?
if (latka == '1')
{
  registrCervena += pow(2, sloupec);
  registrZelena += 0;
}

// zelena?
if (latka == '2')
{
  registrCervena += 0;
  registrZelena += pow(2, sloupec);
}

// oranzova?
if (latka == '3')
{
  registrCervena += pow(2, sloupec);
  registrZelena  += pow(2, sloupec);
}

// vypnuta?
if (latka == '0')
{
  registrCervena += 0;
  registrZelena += 0;
}

// zapis hodnoty do registru
digitalWrite(latchPin, LOW);
shiftOut(dataPin, clockPin, MSBFIRST, registrRadek);
shiftOut(dataPin, clockPin, MSBFIRST, registrZelena);
shiftOut(dataPin, clockPin, MSBFIRST, registrCervena);
digitalWrite(latchPin, HIGH);
```

```
    // zpozdeni v us
    delayMicroseconds(2000);

    registrZelena  = 0;
    registrCervena = 0;
  }
```

Metoda *rozsvitVsechnyDiody()* postupně projde celou matici pomocí dvou cyklů. V cyklu řádků se bude načítat pouze hodnota aktuálního řádku a po vykonání cyklu sloupců se odešlou hodnoty na výstupy registrů. V cyklu sloupců se načte hodnota látky. Rozpoznání látky a nastavení hodnoty konkrétního registru sloupců se provádí úplně stejně jako v předchozí metodě. V tomto případě je ale potřeba v cyklu sloupců posčítat jednotlivé hodnoty pro konkrétní registr. Nechceme totiž rozsvěcovat jednotlivé diody v každém cyklu, ale díky posuvnému registru máme tu výhodu, že se dají rozsvítit všechny sloupce naráz. Tudíž stačí hodnoty registrům posílat při každém cyklu řádků. Další výhodou bude výrazné snížení problikávání diod, které by bylo zřejmé díky rychlému přepínání diod v každém jednotlivém cyklu sloupců.

```
  void rozsvitVsechnyDiody() {

    // typ latky
    char latka;

    // projed kazdy radek
    for (int radek = 0; radek < radkuCelkem; radek++)
    {

      // nacti hodnotu radku
      registrRadek = radky[radek];

      // projed kazdy sloupec
      for (int sloupec = 0; sloupec < sloupcuCelkem;
sloupec++)
      {

        // nacti hodnotu latky z matice
        latka = matice[sloupec][radek];

        // cervena?
        if (latka == '1')
```

```
      {
        registrCervena += pow(2, sloupec);
        registrZelena += 0;
      }

      // zelena?
      if (latka == '2')
      {
        registrCervena += 0;
        registrZelena += pow(2, sloupec);
      }

      // oranzova?
      if (latka == '3')
      {
        registrCervena += pow(2, sloupec);
        registrZelena  += pow(2, sloupec);
      }

      // vypnuta?
      if (latka == '0')
      {
        registrCervena += 0;
        registrZelena += 0;
      }
    }

    // zapis hodnoty do registru
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, registrRadek);
    shiftOut(dataPin, clockPin, MSBFIRST, registrZelena);
    shiftOut(dataPin,clockPin, MSBFIRST, registrCervena);
    digitalWrite(latchPin, HIGH);

    // zpozdeni v us
    delayMicroseconds(2000);

    registrZelena  = 0;
    registrCervena = 0;
  }
}
```

Kontroluje se zde také, zda nebylo stisknuto tlačítko, které by provedlo svou akci podle zvoleného režimu.

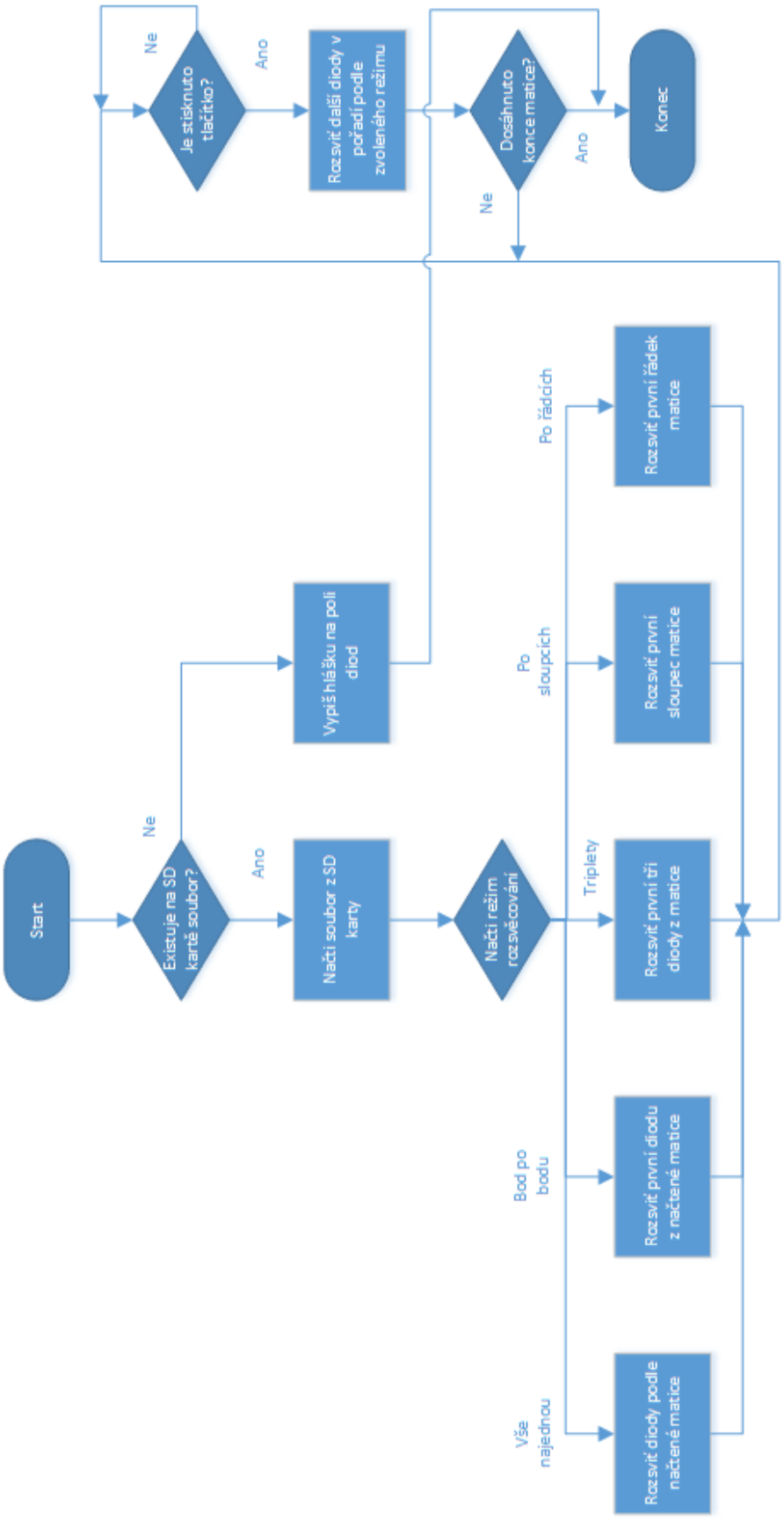
```
// nacti uroven tlacitka
tlacitko = digitalRead(buttonPin);
```

Pokud je hodnota tlačítka v logické jedničce, došlo k jeho sepnutí. Pro lepší reakci na tlačítko je vhodné nastavit nějaké zpoždění, např. 500ms. Pokud bylo tlačítko stisknuto v režimu sloupců, zvýší se aktuální pozice sloupce o jeden. V případě dosažení konce matice se pokračuje od začátku. Režim řádků funguje v principu stejně jako režim sloupců. Pokud je režim rozsvěcování bod po bodu, zvýší se ukazatel pozice na poli pořadí. Pokud bylo dosaženo konce pole pořadí, začíná se opět od začátku. V případě režimu tripletů je princip stejný jako u režimu bod po bodu, jen s tím rozdílem, že se zvýší ukazatel pozice na poli pořadí o tři místa.

```
if (tlacitko == HIGH)
  {
    // cekani pul vteriny z duvodu lepsi reakce na tlacitko
    delay(500);
    // rezim sloupcu
    if (rezim == 'S')
    {
      aktualniSloupec++;
      if (aktualniSloupec == sloupcuCelkem)
      {
        aktualniSloupec = 0;
      }
    }
    // rezim radku
    if (rezim == 'R')
    {
      aktualniRadek++;
      if (aktualniRadek == radkuCelkem)
      {
        aktualniRadek = 0;
      }
    }
    // triplety
    if (rezim == 'T')
    {
      aktualniPoradi += 3;
```

```
        }
        // vse ostatni
        else
        {
          aktualniPoradi++;
        }
        //  pokud  bylo  prekroceno  pole  poradi,  vynuluj
ukazatel pozice
        if (aktualniPoradi >= celkemPoradi)
        {
          aktualniPoradi = 0;
        }
      }
    }
```

Názorný příklad běhu programu je vidět na Obr. 2.5.



Obr. 2.5: Vývojový diagram programu pro mikrokontrolér

3 Grafické uživatelské rozhraní

Software pro grafické uživatelské rozhraní byl naprogramován v jazyce Java.

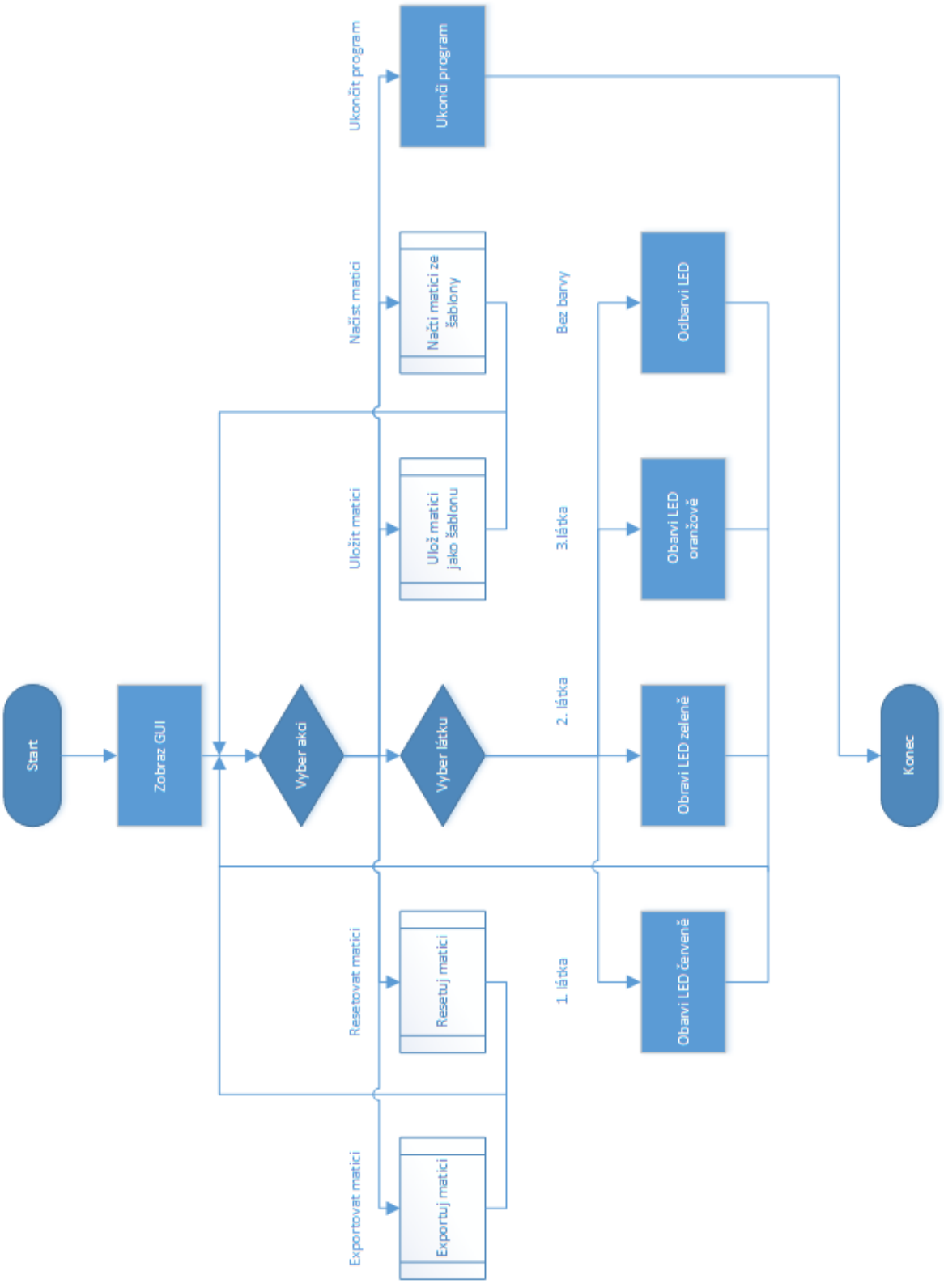
Požadavky na program byly, aby byl schopný vykreslit matici 8x12, která bude představovat LED pole. Dále uživateli poskytne nezbytně nutné funkce pro pohodlné ovládání programu. Těmi budou funkce pro uložení matice do souboru, načtení matice ze souboru, resetování matice, uložení matice na µSD kartu pro používání přípravku bez PC. Pomocí grafického uživatelského rozhraní si uživatel navolí pozice, barvy a režim rozsvěcování, který se uloží do matice a ta je nahrána na µSD kartu, která je uživatelem vložena do platformy Arduino a načtena mikrokontrolérem ATmega328. Potom si uživatel jednoduše pomocí hardwarového tlačítka rozsvěcuje a zhasíná příslušné diody podle zvoleného titračního programu a zvolené barvy.

Tato základní funkcionalita je znázorněna pomocí vývojového diagramu na obrázku 3.1. Rozpis jednotlivých funkcí pak následuje na obrázcích 3.2 – 3.5.

3.1 Funkce hlavního programu

Po spuštění programu se uživateli zobrazí grafické rozhraní s maticí a obslužnými tlačítky. Uživatel má možnost si vybrat z několika akcí. Má na výběr exportovat matici na µSD kartu, resetovat aktuální nastavení matice, uložit aktuální stav matice do souboru jako šablonu, načíst uloženou matici ze šablony nebo ukončit program. Dále má na výběr několik látek, které jsou reprezentovány barvou. Tyto barvy pak používá k označování diod matice.

Vývojový diagram hlavního programu je zobrazen na obr. 3.1.

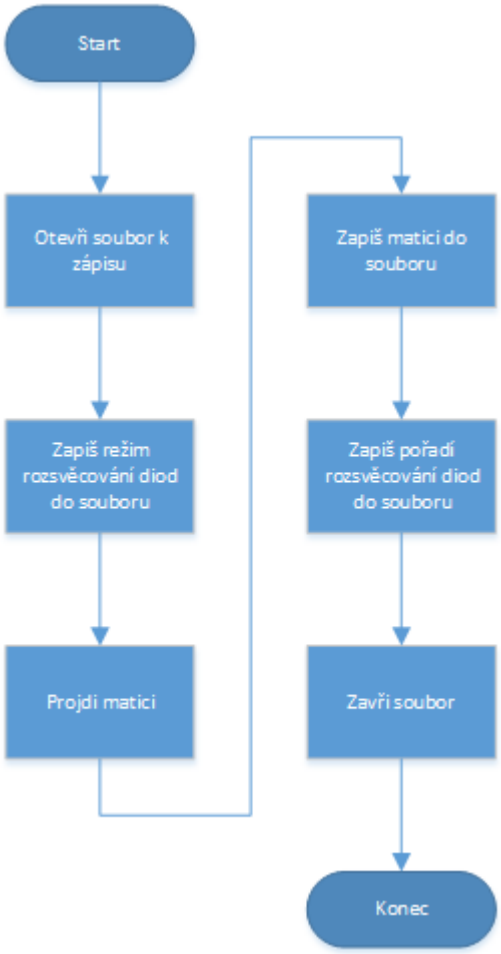


Obr. 3.1: Vývojový diagram hlavního programu

3.2 Exportuj matici

Po zvolení akce pro export matice se uživateli zobrazí okno s výběrem souboru. Po výběru a otevření souboru se zapíše nejprve zvolený režim rozsvěcování diod. Režimy jsou reprezentovány jedním velkým písmenem. Jsou jimi písmena B (bod po bodu), T (triplety), R (po řádcích), S (po sloupcích) a V (vše najednou). Po zapsání režimu se v cyklu projde celá matice a zapíše se do souboru. Do souboru se zapisují vždy všechny diody a každá dioda je reprezentována číslem. Číslo 0 značí neobarvenou diodu, číslo 1 barvu červenou, číslo 2 barvu zelenou a číslo 3 barvu oranžovou. Pokud zvolený režim rozsvěcování diod umožňuje rozsvěcování diod v pořadí jejich označení, je tato posloupnost zapsána na konec souboru. Poté je soubor k zápisu uzavřen.

Vývojový diagram exportování matice je zobrazen na obr. 3.2.



Obr. 3.2: Exportuj matici

3.3 Resetuj matici

Akce resetování matice projde v cyklu celou matici a odbarví všechny diody. V programové reprezentaci matice to znamená, že se do pole matice zapíší samé nuly. Pokud byl zvolený režim rozsvěcování diod, který podporuje rozsvěcování diod v pořadí jejich označení, je toto pole pořadí také inicializováno.

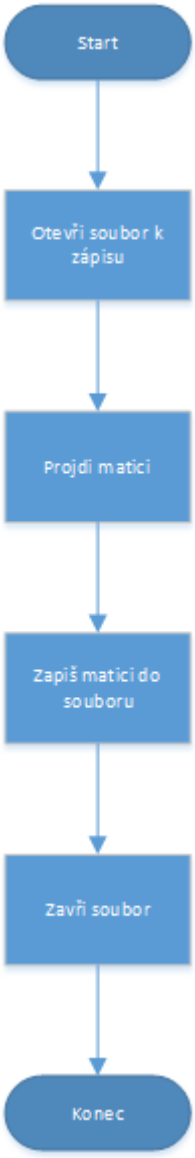
Vývojový diagram resetování matice je zobrazen na obr. 3.3.

Obr. 3.3: Resetuj matici

3.4 Ulož matici jako šablonu

Po výběru této akce se uživateli zobrazí okno s výběrem souboru. Po otevření souboru k zápisu se projde v cyklu celá matice a zapíše se do souboru ve stejném formátu jako při exportu na µSD kartu. Po zápisu celé matice se soubor uzavře.

Vývojový diagram uložení matice je zobrazen na obr. 3.4.



Obr. 3.4: Ulož matici jako šablonu

3.5 Načti matici ze souboru

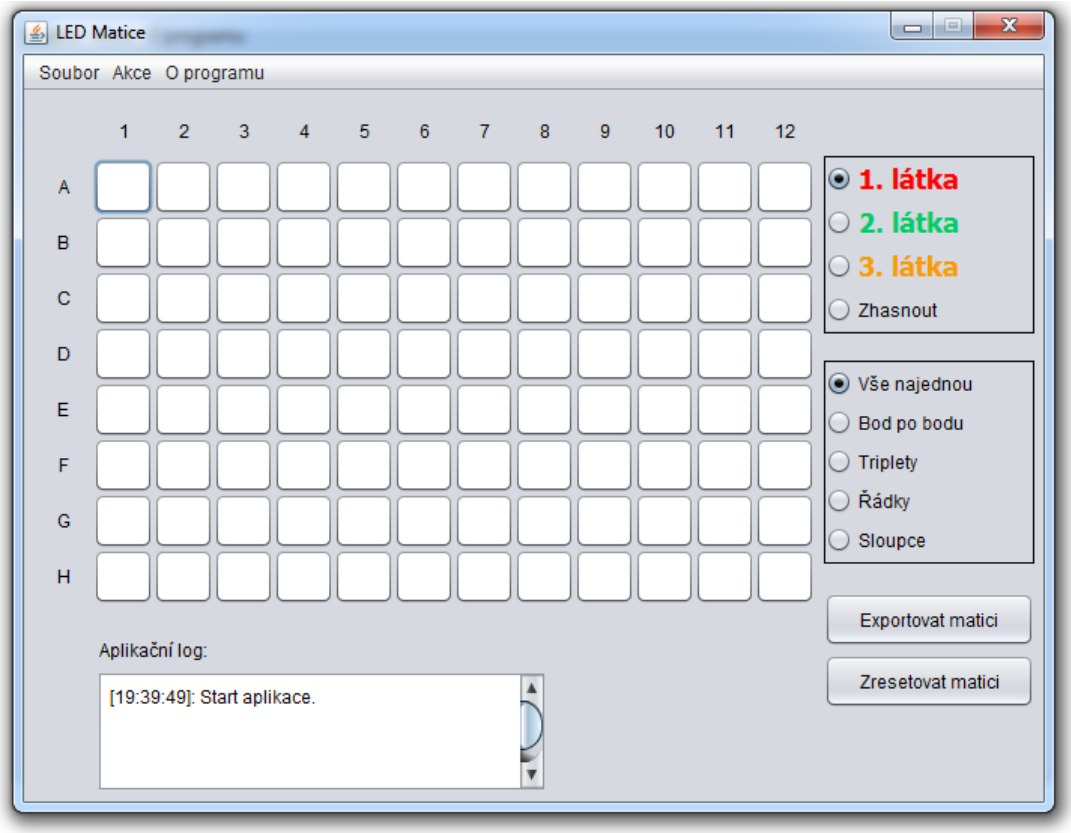
Stejně jako v předchozím případě, se i zde po výběru této akce uživateli zobrazí okno s výběrem souboru. Po otevření souboru ke čtení se projde v cyklu celý soubor a načtené hodnoty se uloží do lokálního pole matice. Po zpracování celého souboru se soubor uzavře a matice v uživatelském rozhraní se překreslí podle načteného nastavení.

Vývojový diagram načtení matice je zobrazen na obr. 3.5.

Obr. 3.5: Načti matici ze souboru

3.6 Popis grafického uživatelského rozhraní

Grafická část programu je tvořena několika dialogovými okny (JDialog), z nichž to hlavní bylo navrženo následovně, viz Obr. 3.6.



Obr. 3.6: Hlavní okno programu

V horní části okna se nachází uživatelské menu (JMenuBar) s nabídkou pro otevření šablony, ukončením programu, výběrem akce pro reset nebo export matice na µSD kartu a jako poslední je informační okno s kontaktem na autora programu.

Největší plochu okna pak tvoří panel (JPanel), na který jsou dynamicky umísťována tlačítka (JButtony) reprezentující LED. Panel má definovanou mřížku o rozměrech 13x9, což je o jeden sloupec a jeden řádek víc, než původní matice diod. To z toho důvodu, že jeden řádek a jeden sloupec reprezentují popisky. Po pravé straně jsou pak dvě skupiny přepínačů (JRadioButtony), z nichž ta první určuje, jakou barvou se budou jednotlivé tlačítka obarvovat po kliknutí na ně, a ta druhá určuje, jakým způsobem se budou LED rozsvěcovat po stisku tlačítka na hardwaru. Tlačítka (JButtony) po pravé straně pak suplují funkcionalitu uživatelského menu.

A nakonec pod panelem matice je umístěn aplikační log (JTextArea), kde se budou uživateli vypisovat důležité informační hlášky nebo případné chyby.

3.7 Popis logiky programu

Panel LED matice má nadefinovanou Post-Init metodu addLeds(), která přidává tlačítka na panel. V této metodě se inicializuje lokální seznam (HashMap) tlačítek ledList<ID tlačítka, tlačítko>, do kterého jsou vložena všechna vytvořená tlačítka pro pozdější snadné vyhledávání a manipulaci s nimi. Informace a popisy základních metod a funkcí jsou v následující literatuře [5][2]. Dále je zde cyklus pro vytvoření klikatelných popisků sloupců, které jsou tvořeny klasickými JLabely s metodami pro kliknutí myši *mousePressed()* a pro změnu kurzoru při najetí na popisek *mouseEntered()*. Metoda pro kliknutí myši volá další metodu *ledLightColumn()*, jejímž parametrem je číslo sloupce, který se po kliknutí obarví.

Po vykreslení popisků sloupců přichází na řadu samotné vykreslení tlačítek. Ovšem každý první sloupec v mřížce panelu je osazen opět JLabelem reprezentujícím označení řádku. Má stejné metody jako vytvořené popisky pro sloupce, ovšem se liší v metodě volané po stisku tlačítka myši. Tentokrát se volá metoda *ledLightRow()* s parametrem čísla řádku, který se po kliknutí obarví. Samotná tlačítka jsou pak řešená klasickými JButtony s výchozím bílým pozadím a metodou *actionPerformed()* volající metodu *ledLight()* po stisku tlačítka.

Pokud by se mělo jít do detailu metody *ledLight()*, která nastavuje barvu tlačítku, tak v této metodě se zjišťuje, které ze zvolených radiobuttonů pro výběr barvy je zaškrtnuté a nastaví si podle toho lokální proměnnou *value*. Tu pak předává dál do metody *setLight()*, která už nastavuje konkrétní barvu pozadí na základě předávané hodnoty metodou *setBackground()* třídy JButton. Po nastavení barvy pozadí tlačítka se zavolá metoda *setLedValue()*, jejími parametry jsou ID tlačítka a barva pozadí. V této metodě se zapíše barva pozadí do globální proměnné *matrix[][]*, která reprezentuje LED matici o velikosti 8x12. Tato proměnná vždy přesně kopíruje stav označení LED (tlačítek na panelu).

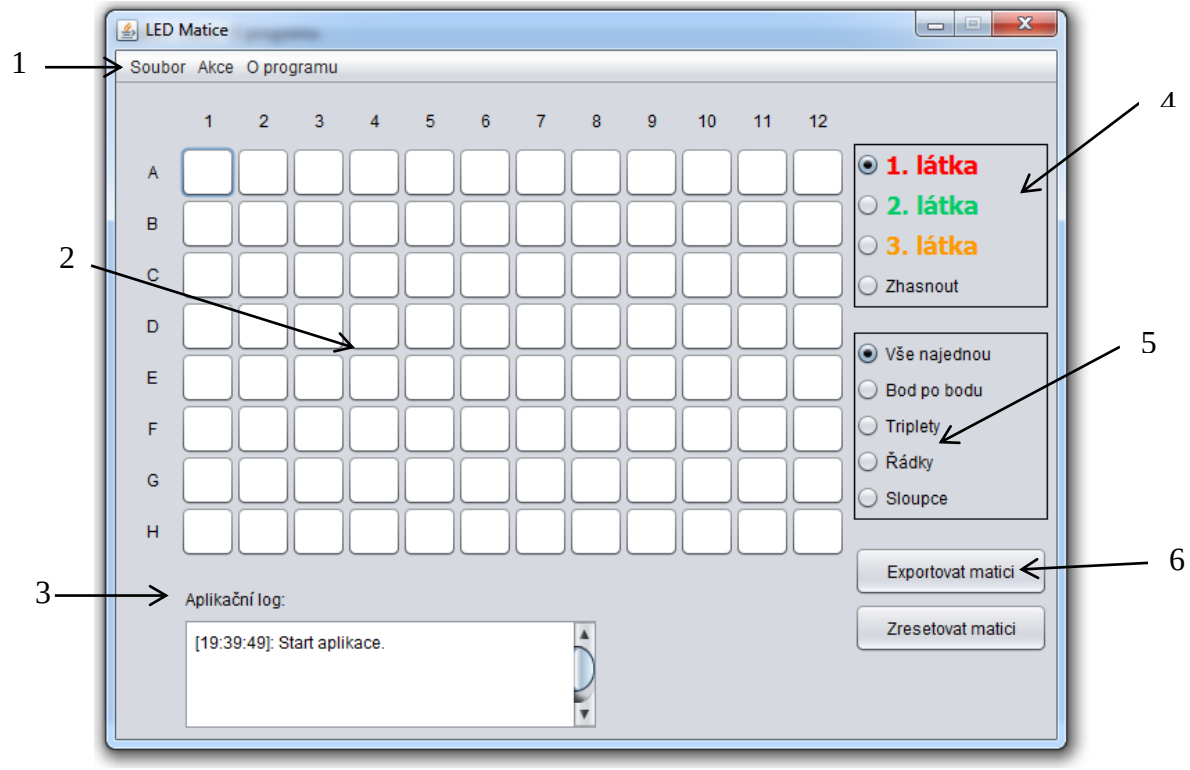
Metoda pro reset matice *resetMatrixPopup()*, která je volána při stisku tlačítka či výběrem z menu, vyvolá popup okno pro potvrzení akce. Pokud je akce potvrzená, zavolá se metoda *resetMatrix()*, která prochází v cyklu pole *matrix[][]*, kam ukládá výchozí nulovou hodnotu, a ze seznamu diod vybírá jednotlivá tlačítka, která metodou *setLight()* obarvuje na výchozí bílou barvu.

Metoda *saveMatrix()* je určená pro uložení aktuálního stavu tlačítek do souboru. Ta volá přidruženou metodu *pickFile()*, která zobrazí popup okno s komponentou JFileBrowser výběr souboru a vrací vybranou cestu k souboru. Tato cesta je pak použitá k otevření souboru pro zápis a v jednoduchém cyklu zapíše obsah proměnné *matrix[][]* do binárního souboru. Po skončení cyklu soubor zavírá.

Metoda *openMatrix()* je určená k načtení uloženého stavu tlačítek ze souboru. V jádru funguje velice podobně jako metoda *saveMatrix()*, ovšem s tím rozdílem, že z otevřeného souboru zapisuje hodnoty do proměnné *matrix[][]*. Po dosažení konce souboru se soubor zavře a je potřeba ještě projít všechna tlačítka a na základě hodnot v proměnné *matrix[][]* je patřičně obarvit. Principy návrhu GUI jsou popsány v následujícím odkazu [15].

4 Práce s uživatelským rozhraním

Po spuštění programu se objeví následující uživatelské rozhraní, viz obr. 4.1.

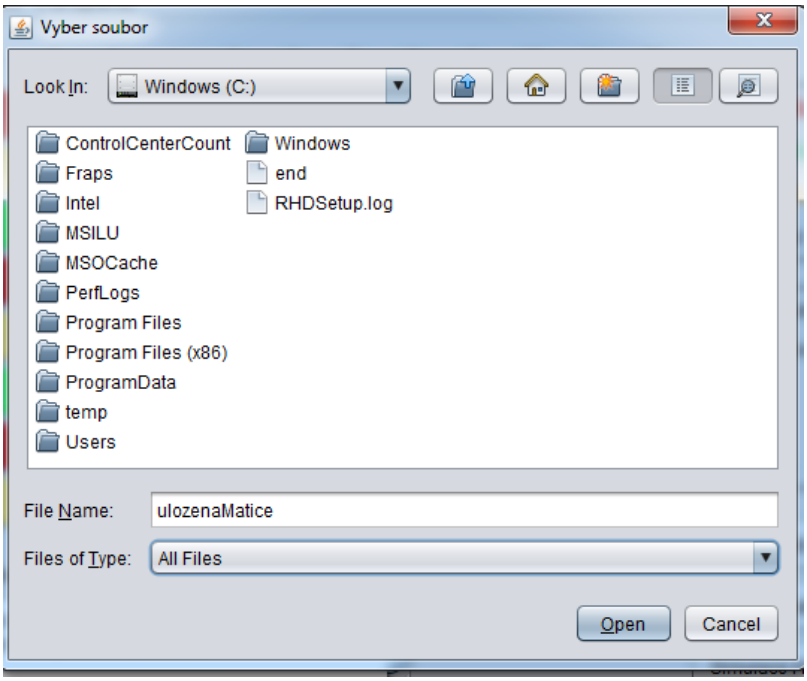


Obr. 4.1: Uživatelské rozhraní

Uživatelské rozhraní se rozděluje na následující oblasti:

1) Hlavní menu

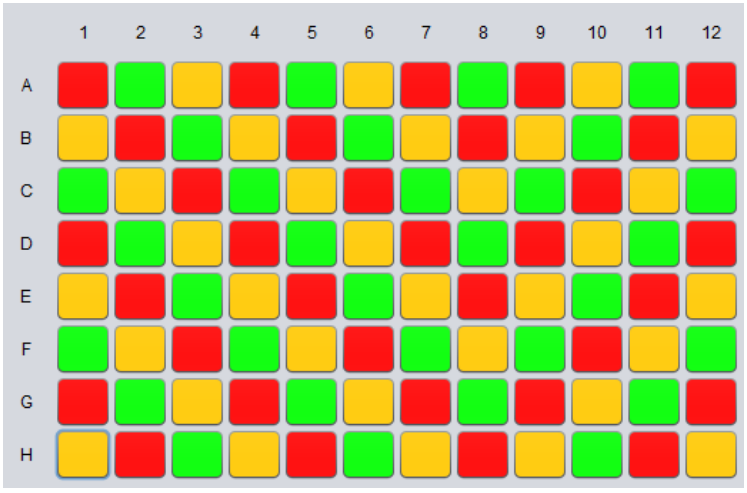
Hlavní menu obsahuje 3 nabídky. První je nabídka *Soubor*, která má možnost otevřít již uloženou šablonu nebo uložit nově vytvořenou, viz obr. 4.1. V obou případech je nutno vybrat soubor z disku. Druhá je nabídka *Akce*, která umožňuje exportovat matici na µSD kartu nebo ji resetovat. Třetí nabídka *O programu* obsahuje kontaktní informace na tvůrce programu.



Obr. 4.2: Výběr souboru

2) Maticové pole

Maticové pole (viz obr. 4.3) obsahuje 12 sloupců značených čísly a 8 řádků značených písmeny. Každá buňka matice je reprezentovaná stisknutelným tlačítkem. Po stisku tohoto tlačítka se buňka zbarví dle vybrané látky. Dále je zde možnost obarvit celý řádek nebo sloupec kliknutím na příslušné číslo sloupce nebo písmeno řádku.



Obr. 4.3: Obarvené maticové pole

3) Aplikační log

V této oblasti se vypisují informační nebo chybové hlášky.

4) Výběr látky

K dispozici jsou 3 látky, z nichž každá je reprezentována určitou barvou. Dále je zde možnost pro odbarvení již obarvené buňky.

5) Režim obarvování na hardwaru

První režim *Vše najednou* odesílá informace o pozici naráz, to znamená, že po uložení na µSD kartu a načtení karty přípravkem se obarví všechny zvolené diody najednou. Druhý režim *Bod po bodu* odesílá informace o pozici po jednom, tedy po načtení karty přípravek čeká na stisk hardwarového tlačítka a postupně rozsvěcuje jednotlivé diody v takovém pořadí, v jakém byly uloženy. Třetí režim *Triplety* ukládá informace o pozici po třech buňkách a opět po stisku hardwarového tlačítka jsou rozsvěcovány po třech v takovém pořadí, v jakém byly uloženy. Čtvrtý režim *Řádky* a pátý režim *Sloupce* umožňují po stisku hardwarového tlačítka rozsvěcovat jeden sloupec za druhým nebo jeden řádek za druhým bez nutnosti obarvovat buňky na panelu.

6) Funkční tlačítka

První tlačítko *Exportovat matici* otevře vyskakovací okno, kde si uživatel vybere uložení souboru na µSD kartu, kterou pak fyzicky vloží do přípravku. Tento soubor obsahuje informace o pozici, barvě LED a režimu rozsvěcování LED. Druhé tlačítko *Zresetovat matici* uvede maticové pole do původního (neobarveného) stavu.

5 Diskuze

Během návrhu přístroje byly provedeny změny v hardwarovém zapojení. V původním návrhu byl navržen obvod s použitím čítače, který zajišťoval rozsvěcování řádků. Avšak testování hardware s čítačem vykazovalo chyby v podobě špatné kontroly časování, kdy časovač rozsvěcel jiné sloupce LED.

Lepší možnosti řízení vykazovaly posuvné registry, které byly v prvotním návrhu využity pouze pro řízení řádků matice LED. Z toho důvodu byl časovač vyměněn za 8bitový posuvný registr. Z důvodu řízení 12 sloupců a 8 řádků byla matice transponována tak, aby bylo možné řídit 8bitovým registrem 8 řádků a třemi 8bitovými registry bezezbytku 12 + 12 sloupců (tj. 12 pro zelenou a 12 pro červenou barvu). Tato varianta řešení umožnila přesné časování a tím bezproblémové řízení matice bod po bodu a příslušné barvy.

Též řídicí program pro mikrokontrolér došel během řešení práce značné změny, kdy se měl mikrokontrolér ovládat (resp. programovat) přímo z počítače pomocí knihovny JArduino [11] v programovacím jazyku Java. Tento návrh měl slabinu v tom, že by byl mikrokontrolér buďto přímo programován, nebo by musel mít paralelně samostatně běžící program, který by pouze přebíral vstupy z počítače (konfiguraci matice), zařízení by ale v takovém případě nebylo typu „stand-alone“. Z toho důvodu je finální řešení pomocí µSD karty, kdy se na kartu předem uloží do souboru matice z obslužného Java programu a soubor se načte z karty přímo v přípravku.

Po softwarové stránce nastal problém pouze s ovládáním registrů, kdy byly nedopatřením kontinuálně otvírány registry k zápisu vícekrát, než bylo potřeba, což způsobilo zkratování latch pinů. Toto bylo opraveno a registry jsou otevřeny pro zápis pouze jednou.

Závěr

Cílem této práce bylo navrhnout a sestavit přípravek, který bude indikovat pozice mikrotitrační destičky pro pipetování. Dále sestavit řídicí program pro mikrokontrolér, který bude tento přípravek ovládat a obslužný program pro uživatele.

V prvé řadě jsem se musela seznámit s provedením mikrotitrační destičky, samotným pipetováním, ovládáním mikrokontroléru a analyzovat požadavky uživatelů na takovou aplikaci.

Celý koncept se skládá jak z části hardwarové tak softwarové. Celé zařízení je „stand-alone“, což znamená, že funguje samostatně bez připojení k počítači. V obslužném programu, tedy grafickém uživatelském rozhraní, si uživatel volí pipetovací plán, konkrétně určí barvu z výběru tří možných barev a titrační režim, podle které se LED budou rozsvěcovat. Tento pipetovací plán uživatel nahraje na µSD kartu a vloží do přípravku. Celé zařízení bude napájeno pomocí externího napájecího adaptéru. Pro mikrokontrolér ATmega328 byl sestaven a odladěn řídicí program, který načte údaje z µSD karty a nastaví příslušný titrační režim a napovídá uživateli postup pipetování pomocí LED. Uživatel další postup pipetovacího plánu ovládá pomocí hardwarového tlačítka na přípravku.

Pro sestavení obslužného uživatelského programu jsem zvolila programovací jazyk Java. Jazyk Java byl zvolen jednak z důvodu zajištění kompatibility programu na různých platformách operačních systémů, dále proto, že jsem viděla možnost a příležitost se zdokonalit v objektově orientovaném programování. Jako vývojové prostředí pro sestavení programu bylo zvoleno prostředí NetBeans, se kterým jsem měla dřívější zkušenosti s návrhem jiného grafického rozhraní.

Návrh užitečného přípravku byl pro mě velice významný hlavně z hlediska rozšíření mých znalostí z oblasti mikroprocesorové techniky a programovacího jazyku Java a C. Domnívám se, že zadání bylo splněno, po případné realizaci popsaného přípravku jako celku na samostatné desce plošných spojů s osazením SMD součástkami se jistě bude jednat o užitečnou aplikaci.

Literatura

[1] Arduino UNO. [online]. [cit. 2014-12-23]. Dostupné z: http://arduino.cc/en/Main/arduinoBoardUno

[2] ARNOLD, K. 1996. *Java programming language*. Massachusetts: Addison-Wesley, 373 s. ISBN 02-016-3455-4.

[3] BARTEK, Tomáš. 2014. *Rozhraní pro ovládání průmyslových modulů LED*. 51 l.

[4] BAYLE, Julien. *C programming for Arduino: learn how to program and use Arduino boards with a series of engaging examples, illustrating each core concept*. Birmingham: Packt Pub., 2013, s. 266-268.

[5] DARWIN, Ian F. 2004. *Java cookbook*. 2nd ed. Sebastopol, CA: O'Reilly, 829 p. ISBN 05-960-0701-9.

[6] EVANS, Brian. *Beginning Arduino programming*. New York: Distributed to the book trade worldwide by Springer Science+ Business Media, c2011, chapter 1: Getting Started, s. 1-5. ISBN 1430237775.

[7] INSTRUMENTS, Texas. *8-BIT SHIFT REGISTERS WITH OUTPUT LATCHES* [online]. roč. 1981, 2014 [cit. 2014-12-23]. Dostupné z: http://www.ti.com/lit/ds/symlink/sn54ls595.pdf

[8] MARGOLIS, Michael. *Arduino cookbook*. 2nd ed. Sebastopol: O'Reilly, 2011, chapter 1: Getting Started, ISBN 9780596802479.

[9] MARGOLIS, Michael. *Arduino cookbook*. 2nd ed. Sebastopol: O'Reilly, 2011, chapter 2: Making the Sketch Do your Bidding. ISBN 9780596802479.

[10] MARTIN EVANS, Joshua Noble. *Arduino in action*. Shelter Island, N. Y.: London: Oreilly & Associates Inc, 2013, s. 65-67. ISBN 9781617290244.

[11] MORIN, Brice. *JArduino* [online]. [cit. 2015-05-17]. Dostupné z: https://github.com/SINTEF-9012/JArduino/wiki

[12] PLATT, Charles. *Encyclopedia of electronic components*. volumes. ISBN 97814493341852.

[13] STMICROELECTRONICS. *BC337-40: SMALL SIGNAL NPN TRANSISTORS* [online]. [cit. 2015-05-22]. Dostupné z: http://pdf.datasheetcatalog.com/datasheet/stmicroelectronics/8853.pdf

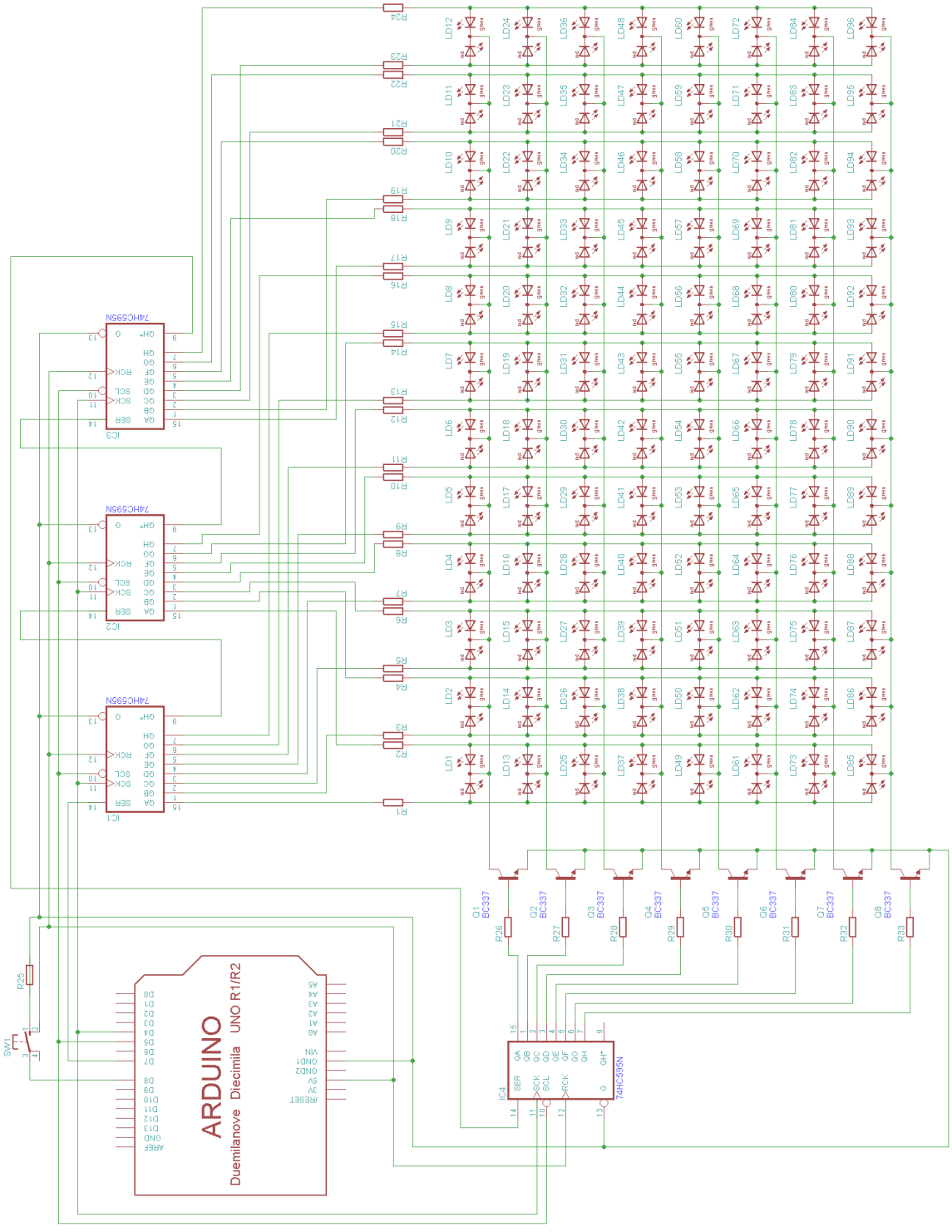
[14] STORR, Wayne. Simple LED Flasher. [online]. 2014-12-22 [cit. 2014-12-23]. Dostupné z: http://www.electronics-tutorials.ws/counter/simple-led-flasher.html

[15] ŠPERKA, Martin. *HCI design and GUI programming in Java*. 1st ed. Prague: Wolters Kluwer, 2014, Chapter 4: GUI PROGRAMMING PRINCIPLES AND TOOLS, s. 68-90. ISBN 978-80-7478-567-2.

[16] YE, S a Ian N DAY. 2003. *Microarrays: applications in biomedical sciences*. New York: Distributed in the U.S.A. by Springer-Verlag, ix, 178 p. ISBN 18-599-6074-X.

Seznam příloh

A CELKOVÉ SCHÉMA ZAPOJENÍ



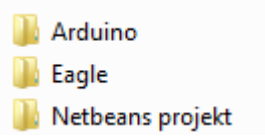
B SEZNAM SOUČÁSTEK

Označení	Hodnota	Popis
IC1 – IC4	74HC595N	8bitový posuvný registr
R1 – R24	330R	Rezistory před diodami
R25	1k	Rezistor před tlačítkem
R26 – R33	56k	Rezistory před tranzistorem
Q1 – Q8	BC337	Bipolární tranzistor NPN
SW1		Tlačítko
LD1 – LD96		Dvoubarevná LED

C OBSAH PŘILOŽENÉHO CD

Bakalářská práce: Gabriela_Papajova_BP.pdf

V kořenovém adresáři CD jsou umístěny následující složky.



Složka *Arduino* obsahuje zdrojový kód pro HW přípravek.

Složka *Eagle* obsahuje elektrické schéma celého obvodu.

Složka *Netbeans projekt* obsahuje Java projekt GUI pro tvorbu pipetovacího plánu.