**ZÁPADOČESKÁ UNIVERZITA V PLZNI**
**FAKULTA ELEKTROTECHNICKÁ**

**KATEDRA ELEKTROMECHANIKY A VÝKONOVÉ ELEKTRONIKY**

# BAKALÁŘSKÁ PRÁCE

## Ovládání prvků diodového pole pomocí LabVIEW

**Michal Knedlík** **2018**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
**Fakulta elektrotechnická**
Akademický rok: **2017/2018**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Michal KNEDLÍK** |
| Osobní číslo: | **E15B0064P** |
| Studijní program: | **B2612 Elektrotechnika a informatika** |
| Studijní obor: | **Elektrotechnika a energetika** |
| Název tématu: | **Ovládání prvků diodového pole pomocí LabVIEW** |
| Zadávající katedra: | **Katedra elektromechaniky a výkonové elektroniky** |

Zásady pro vypracování:

1. Navrhněte s využitím DAQ karty rozmístění LED s ohledem na jejich vyzařovací úhel a vzájemnou polohu na panelu.
2. Navrhněte algoritmus adresace portů karty a jejich časování pro diodovou matici.
3. Doplňte algoritmus o převod ASCII na signálové úrovně DAQ. Respektujte parametry vybraných LED.
4. Rozšiřte program o funkce přesunu znaků a jejich modifikaci při změně vstupu.
5. Navrhněte další možné rozšíření této úlohy.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah kvalifikační práce: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Vlach J., Havlíček J., Vlach M.: Začínáme s LabVIEW, kniha, BEN - technická literatura, Praha, 2008.**

Vedoucí bakalářské práce: **Ing. Pavel Štekl, Ph.D.**
Katedra teoretické elektrotechniky

Datum zadání bakalářské práce: **10. října 2017**

Termín odevzdání bakalářské práce: **7. června 2018**

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

Prof. Ing. Václav Kůs, CSc.
vedoucí katedry

V Plzni dne 10. října 2017

## Abstrakt

Tato práce byla vytvořena pro demonstraci možností LabVIEW a hardwaru od firmy National Instruments. Se zaměřením na vývoj LED zobrazovače, řídícího algoritmu, a výběr vhodných LED diod.

Práce měla několik vývojových fází. Nejprve bylo nutné navrhnout propojení LED diod a otestovat jej na nepájivém kontaktním poli. Poté bylo možné navrhnout algoritmus, který umožňuje zobrazení znaků a slov zadaných z klávesnice počítače. Slova nebo znaky mohou být posouvány v horizontálním nebo vertikálním směru. Nakonec bylo možné připravit finální podobu LED pole. Což zahrnovalo vývoj desky plošného spoje a tisk některých konstrukčních částí v 3D tiskárně.

## Klíčová slova

Led pole, LabVIEW, ascii znaky, LED dioda, algoritmus, blokové schéma, čelní panel

## Abstract

This work was created for reason to demonstrate the possibilities of using LabVIEW and the hardware from National Instruments. Focusing on the development of LED display, controlling algorithm and choice of useable LED diodes.

This work has gone through some development phases. At first it was necessary to develop the connection of LEDs and tested it on non-soldering field. The second thing was to develop the algorithm, which enables us to display the characters and words that were entered from computer keyboard. Moreover, the characters and the words could be moved on the LED array in horizontal or vertical direction. After that it was possible to prepared the final appearance of LED array, which included the development of printed circuit board and to print some construction parts in the 3D printer.

## Key words

LED array, LabVIEW, ascii characters, LED diode, algorithm, Block diagram, Front panel

# Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 29.5.2018 Michal Knedlík

# Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Pavlovi Šteklovi, Ph.D. za cenné rady a metodické vedení práce. Dále bych chtěl poděkovat za cenné rady Ing. Michalovi Chalušovi v oblasti programového řešení a také Ing. Davidovi Rotovi, Ph.D. za umožnění tisku konstrukčních částí na 3D tiskárně.

# Obsah

# Úvod

Tuto práci jsem si vybral především, kvůli programovému prostředí *LabVIEW*, které mně jako elektrotechnikovi připadá názornější a logičtější než běžné textové programovací jazyky.

Předkládaná práce je zaměřena na programové řešení přesunu ascii znaků z klávesnice a jejich zobrazení na navržené LED matici s využitím dostupné karty USB – 6501. Algoritmus odchytává požadavky uživatele, které zpracuje a nastaví vhodné parametry pro zobrazení a posun znaků nebo slov s čímž souvisí adresace digitálních výstupů karty a řízení aktivace jednotlivých bodů pole.

Text je rozdělen do tří hlavních částí. První část se zabývá představením programového prostředí *LabVIEW*. Základními částmi *Front panel, Block diagram*… Parametry použitých karet od firmy *National Instruments*. Popisem základních struktur a logiky.

Druhá část je zaměřena na popis vytvořeného algoritmu. Princip adresace kanálů karty a řízení aktivace příslušných LED diod. Zmiňuji zde rozdíly použitých karet USB-6008 a USB-6501.

Třetí část se zabývá návrhem diodového pole. Od prvotního testovacího zapojení na nepájivém kontaktním poli, až po návrh výsledné desky plošného spoje maticového zobrazovače. Popisuji zde výběr LED diod a jejich rozmístěním na plošném spoji.

# Seznam symbolů a zkratek

ASCII…….. American Standard Code for Information Interchange
(americký standardní kód pro výměnu informací)

CAD…… Computer Aided Design (počítačem podporovaný návrh)

DAQ…… Data AcQuistion (systém sběru dat)

DLP……. Digital Light Projection (vytvrzení světelným zářením)

DC……... direct current (stejnosměrný proud)

3D…… trojdimenzionální

html……. HyperText Markup Language
(značkovací jazyk pro tvorbu webových stránek)

I (cd) ……svítivost

$I_{fmax}$…….. maximální hodnota propustného proudu

$I_R$……. závěrný proud

FDM…… Fused Depostition Modeling (modelování pomocí roztaveného vlákna)

FI-FO……... First In First Out

GND ……... ground ( 0 potenciál)

LabVIEW ..... Laboratory Virtual Instruments Engineering Workbench
(laboratorní pracoviště virtuálních přístrojů)

LCD…… Liquid Crystal Display (displej z tekutých krystalů)

LED ..... Light Emitting Diode (světlo vyzařující dioda)

LOM…… Laminated Object Manufacturing
(výstavba modelu pomocí vrstvení plátů materiálu)

NPN……. bipolární tranzistor s elektronovou vodivostí

PCB..... Printed Circuit Board (deska plošného spoje)

PVC……. Polyvinylchlorid

Si……. křemík

SLS…….. Selective Laser Sintering (Selektivní laserové slinování)

SubVI…….. pod program ( pod VI)

$U_{fmax}$…… maximální hodnota napětí v propustném směru

$U_R$…… závěrné napětí

USB ..... Universal Serial Bus (USB sběrnice)

VI…… Virtual Instrument (virtuální přístroj, název souboru)

$\Delta t$ …… časový úsek

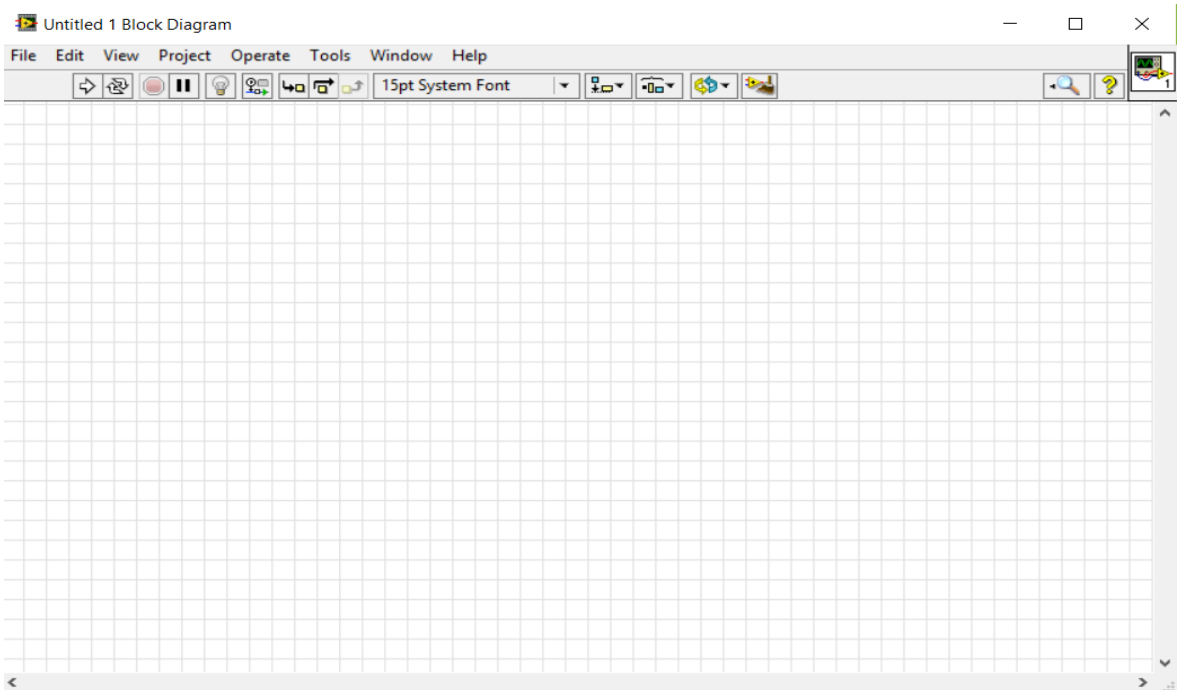
# 1 Vývojové prostředí *LabVIEW*

Pojem *LabVIEW* vznikl jako zkratka z *Laboratory Virtual Instruments Engineering Workbench* v překladu „laboratorní pracoviště virtuálních přístrojů“. A je produktem firmy *National Instruments* založené roku 1976. Již z názvu plyne obrovská výhoda tohoto prostředí. Existencí virtuálních přístrojů firma *National Instruments* vyzdvihuje svůj přínos. Ušetří se náklady za skutečné přístroje a v prostředí *LabVIEW* se každý inteligentní člověk naučí programovat sám. *LabVIEW* usnadňuje práci technickým pracovníkům, protože nemusí znát syntaxi např. textově orientovaných jazyků a lze celkem snadno dohledat funkci jednotlivých bloků. Je zřejmé, že k snímání dat je zapotřebí hardware např. karta *DAQ* (karta pro sběr dat). Další výhodou tohoto systému je grafické programovací prostředí, které může být více názorné a tedy i rychleji upravovatelné na rozdíl od direktivních programovacích jazyků. [1]

## 1.1 Prostředí *LabVIEW*

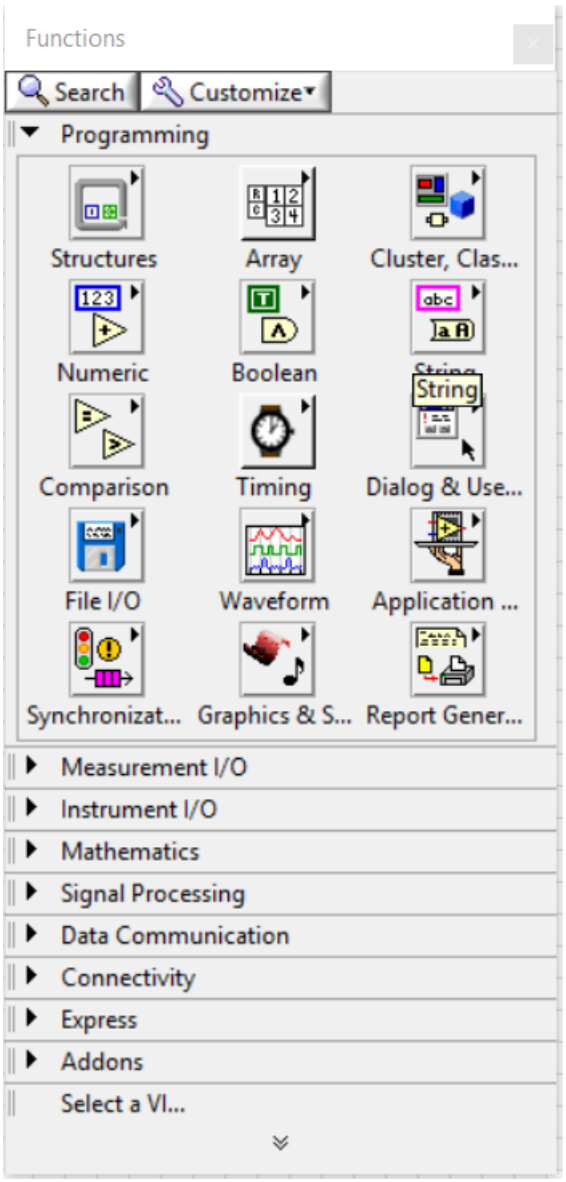
- **Okno blokového diagramu (*Block diagram*)**

Je nejdůležitější část virtuálního přístroje, kde programátor vytvoří vlastní kód. Vybere požadované bloky, které spojuje pomocí logických spojů (*wires*). Složité a rozsáhlé k*ódy* může programátor rozložit do podprogramů, které se v tomto prostředí nazývají *SubVI*.




**Obr. 1 Block diagram**

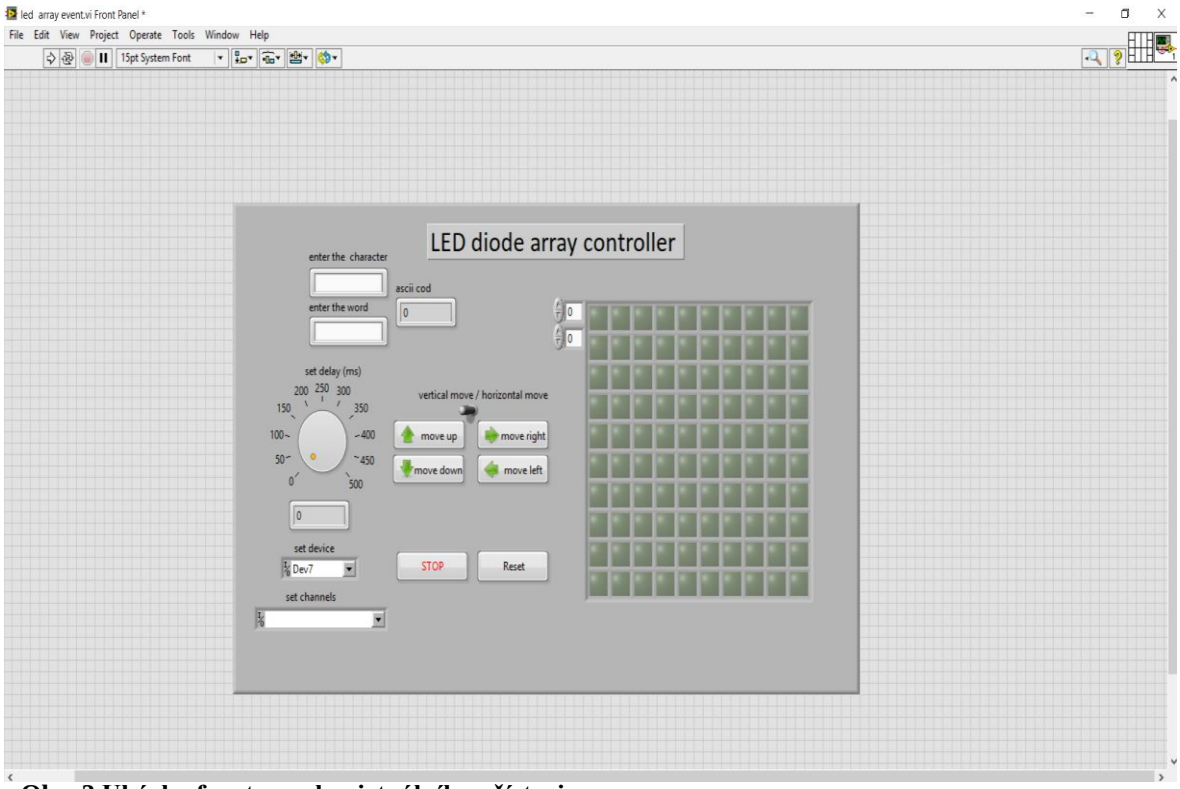
V horní liště **Obr. *1*** vidíme ikony pro řízení běhu programu. Dále pak ikony pro odstraňování chyb tzv. debugging, nástroje pro změnu formátu textů, nebo posun ikon také nástroje pro vyhledávání a zobrazení nápovědy. Při stisknutí pravého tlačítka myši lze vyvolat paletu funkcí, která obsahuje potřebné části programu respektive blokového diagramu. Např. pod ikonou *Structures* lze najít *while* cyklus, *for* cyklus a další…




**Obr. 2 Nabídka functions**

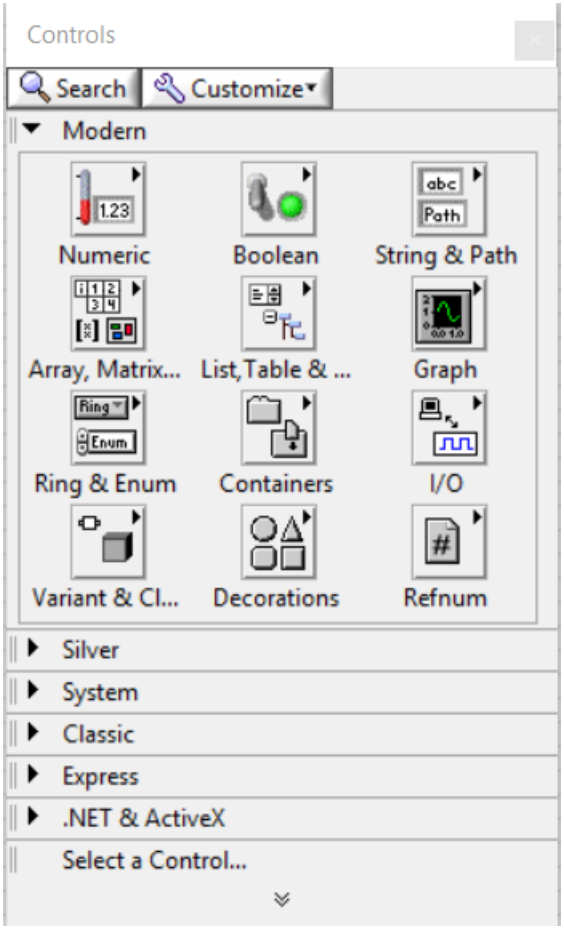
- **Okno čelního panelu aplikace (*Front panel*)**

  Podoba čelního panelu aplikace by měla být maximálně shodná s provedením čelního panelu skutečného měřicího přístroje. Uživatel si zde muže zadávat a měnit parametry pomocí akčních členů tzv. *controls*, kdy dochází k přesunu informace od uživatele do aplikace. Příslušné stavy nebo výsledky zobrazují výstupy tzv. *indicators* např. *waveform chart* nebo *waveform graph*, kdy aplikace (program) předává informace uživateli v podobě grafů.




**Obr. 3 Ukázka front panelu virtuálního přístroje**

V horní liště čelního panelu jsou dále ikony s obdobnými možnostmi jako v blokovém diagramu. Stisknutím pravého tlačítka myši lze zobrazit paletu akčních členů (*controls*). Poté je možné vkládat příslušné výstupy *(indicators*) nebo akční členy (*controls*) různých datových typů tzn. *Numeric*, *Boolean*, *String*, *Eanum*, *Array*… Ve spodní části palety *controls* lze měnit vzhled jednotlivých částí čelního panelu: *Silver*, *System*, *Classic* viz **Obr. 4**.




**Obr. 4 Nabídka Controls**

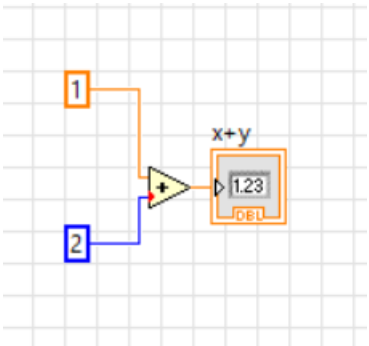
### 1.1.1 Datové typy

Datový typ *Numeric* reprezentuje číselné hodnoty. Lze jej rozdělit na celočíselné hodnoty (*Intiger, fixed-point*) a na čísla s desetinou čárkou (*floating-point*). K dispozici je 12 možností tohoto datového typu s různou přesností (počtem desetinných míst) a tedy s různou velikostí potřebného místa v paměti. Základní datové typy *Numeric*: DBL, I32, U32… jsou zobrazeny na ***Obr. 5***. Z obrázku je zřejmé, že každý datový typ má svoji barvu, kterou nese i logický spoj *(wire*) mezi jednotlivými terminály.[4]

Pro plynulý chod programu je vhodné využívat datové typy s nižší náročností na paměť, ale s dostatečnou přesností pro danou aplikaci. *LabVIEW* si při výpočtu provede samo konverzi na stejný datový typ viz. ***Obr. 6***. Kde je upozorněno na automatickou konverzi datového typu pomocí červené tečky tzv. *coercion dot*. Konverze datového typu v každé iteraci běhu programu způsobuje zpomalení programu, které lze odstranit sjednocením vstupních datových typů.

Datový typ *Boolean* nabývá logických hodnot *true, false*. A na digitálních výstupech karty respektuje hodnoty *high a low* (logická 1 a logická 0). *String* reprezentuje textový řetězec. *Enum* nabývá předem definovaných hodnot a umožňuje práci s textem pomocí přiřazených číselných hodnot. Datový typ *array* je pole, které může obsahovat různé datové typy obdobně jako *Cluster*. V ***Obr. 5*** je vyobrazena ikona *Array* zelené barvy, která obsahuje hodnoty datového typu *Boolean*. Datový typ *matrix* přenáší číselné hodnoty v podobě matice obdobně jako *Array* obsahující číselný datový typ. *Waveform* je datový typ obsahující data v časové oblasti s rozestupy o Δt. Datový typ *waveform* je vhodný pro zobrazení ve *Waveform Chart* nebo *Waveform Graph*. *Waveform Graph* zobrazuje všechna příchozí data najednou a není možné zobrazit historii naměřených nebo vypočtených dat. *Waveform Chart* zobrazí příchozí data včetně předešlých dat do naplnění paměti, poté začíná přepisovat od nejstarších dat. [2]


**Obr. 5 Datové typy**



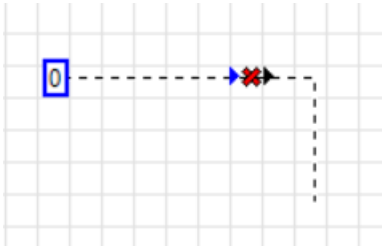

**Obr. 6 Coercion dot**

### 1.1.2 Přenos dat (logického signálu)

Data v rámci VI (soubor s vytvořeným kódem) lze přenášet:

1. Pomocí logických spojů (*wires*)

   Přenos pomocí logických spojů je jednoznačný a tok dat probíhá z levé strany do pravé strany blokového diagramu. Jednotlivé datové typy jsou rozlišeny rozdílnou barvou případně tloušťkou logického spoje např. v závislosti na typu matice 1D, 2D… Pokud dojde k propojení nekompatibilních terminálů, nebo je spoj nedokončený, bude logický spoj přerušovaný a vyhodnocen jako chybný. Program s jakoukoliv chybou nelze spustit.



**Obr. 7 Nedokončený spoj (chybný spoj)**

2. Pomocí lokální proměnné (*Local Variable)*

   Jedná se vlastně o „bezdrátový“ přenos dat v uvnitř jednoho *VI*. Lokální proměnné jsou s výhodou využívány především pro přenos dat mezi více smyčkami. Přenos dat však již není jednoznačný a je nutné postupovat obezřetně. Pomocí lokální proměnné lze měnit hodnotu u výstupu (*indicator*) nebo akčního členu (*control*) zvolením možnosti zápis (*change to write*) nebo čtení (*change to read*). Pokud existuje pro jeden terminál více lokálních proměnných, není jednoznačné pořadí zápisu dat.

3. Pomocí globální proměnné (*Global Variable)*

   Globální proměnné lze využít opět pro „bezdrátový“ přenos mezi více *VI* v rámci projektu. Projekt obsahuje více *VI* nebo i *SubVI*… Do projektu je také nutné implementovat *VI* spravující veškeré globální proměnné.

4. Přes webové rozhraní

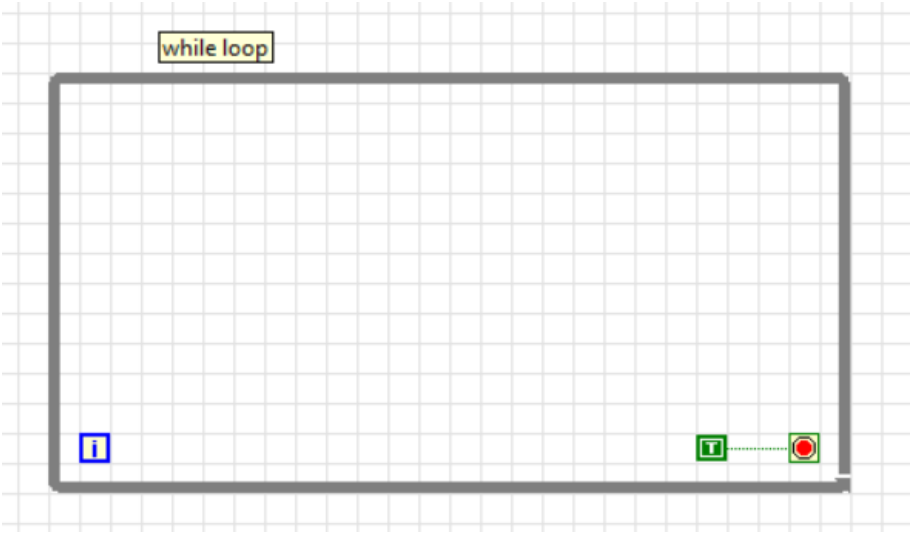
   Přenos dat přes webové rozhraní je výhodné využít pro vzdálené ovládání měřící soustavy nebo pro vzdálené získávání naměřených dat. Pro dané *VI* je nutné ve *web publishing tool* vygenerovat příslušný *html* soubor a vložit jej do projektu. Samozřejmě je nutné nakonfigurovat pro tuto komunikaci počítač nebo jiné zařízení.

### 1.1.3 **Základní struktury** [3]

Některé struktury jsou obdobné jako cykly v textově orientovaných jazycích např. *while* nebo *for* cyklus. V *LabVIEW* mají podobu rámu (*frame*), ve kterém probíhá daná operace.

*While* cyklus *(While Loop)*

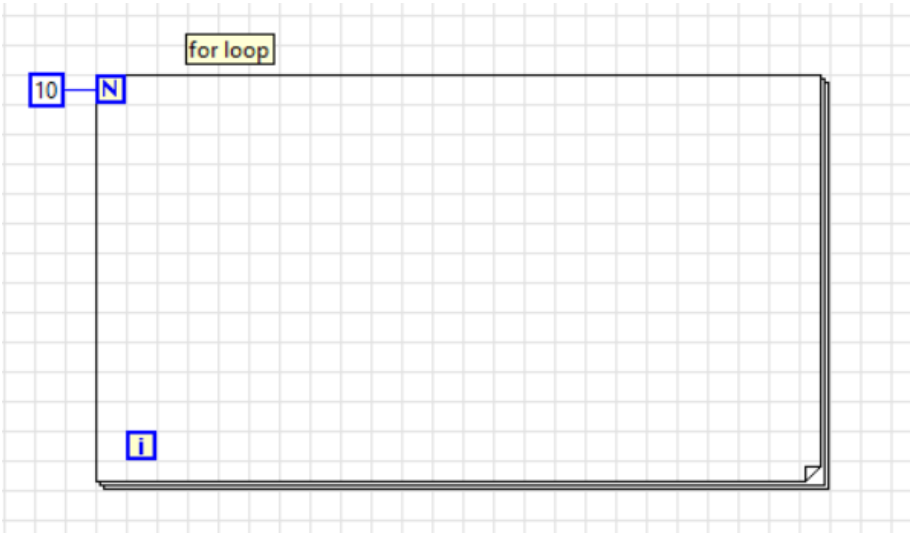
Provádí operace v závislosti na hodnotě, která je na podmínkovém terminálu. Na podmínkový terminál může být připojena *boolean* hodnota *false* nebo *true* v závislosti na nastavení terminálu a požadované akci. Vyhodnocení podmínky probíhá vždy na konci iterace, čili *while loop* proběhne vždy nejméně jednou. Jednotlivé iterace jsou číslovány od nuly.




**Obr. 8 While Loop**

*For* cyklus (*For Loop*)

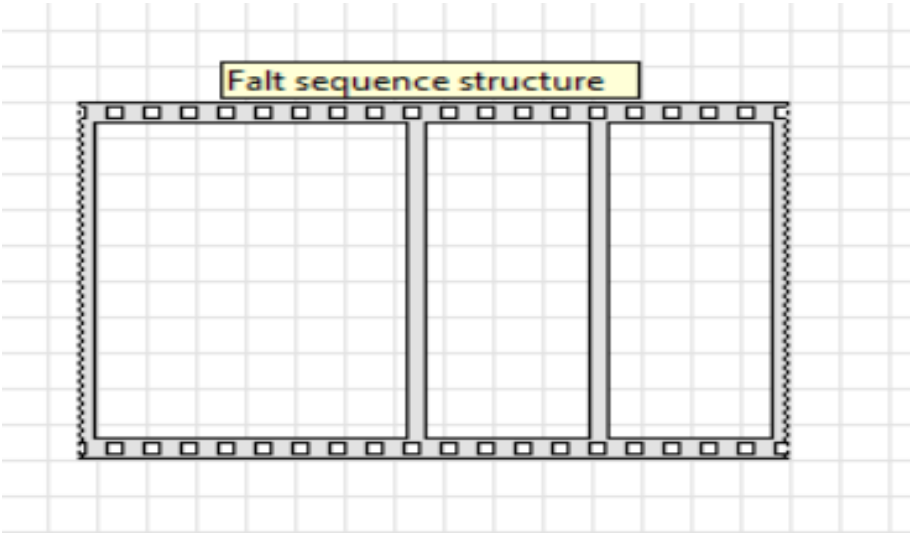
Na rozdíl od *while loop* má cyklus *for* přesně stanovený počet iterací. Počet iterací je dán připojenou hodnotou na terminál N, nebo počtem hodnot pole při nastaveném indexování na vstupním tunelu. Tunel je místo, kde vstupují, nebo vystupují data smyčky. Nastavení tunelu může být: *indexing*, *last value*, *concatenating*. *For loop* se nejvíce využívá pro práci s maticemi.




**Obr. 9 For Loop**

Sekvenční struktura *(Flat Sequence)*

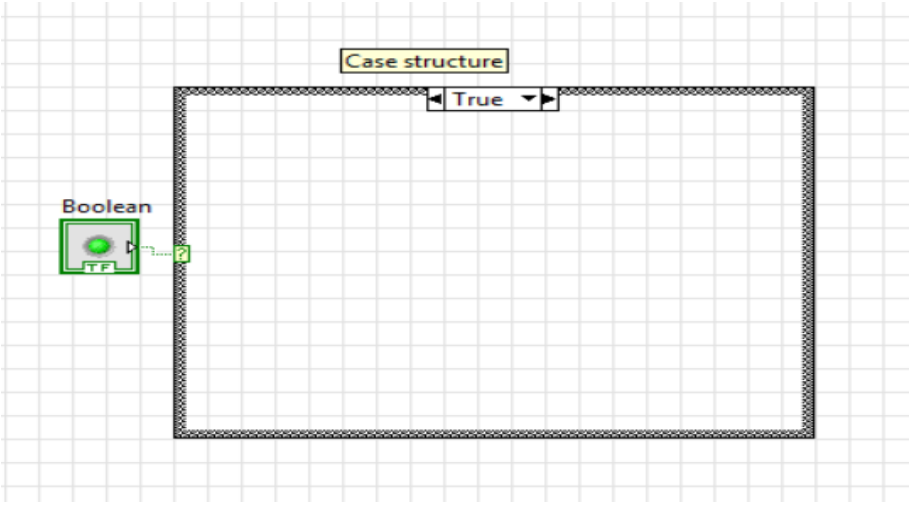
Je určena pro zajištění sekvenčnosti (postupnosti) provádění kódu. Jednotlivé rámce (*frames*) jsou řazeny za sebou tak, jak skutečně budou prováděny. Obdobně pracuje i *Stacked Sequence* s tím rozdílem, že jednotlivé rámce jsou pod sebou nikoliv vedle sebe.




**Obr. 10 Flat sequence**

*Case Structure*

Je obdobou *switche* nebo podmínky *if-else* v jazyce C. Slouží k větvení algoritmu na základě připojené hodnoty na podmínkový terminál. *Case* obsahuje několik rámců (*frames*) a zobrazen, nebo zpracováván je vždy jeden na základě vstupní hodnoty do podmínkového terminálu. Kromě nastavených oken odpovídajících podmínce, je nutné definovat i *default frame*, který se vykoná, pokud je na vstupu např. prázdné pole, nebo jsme mimo rozsah definovaných hodnot.



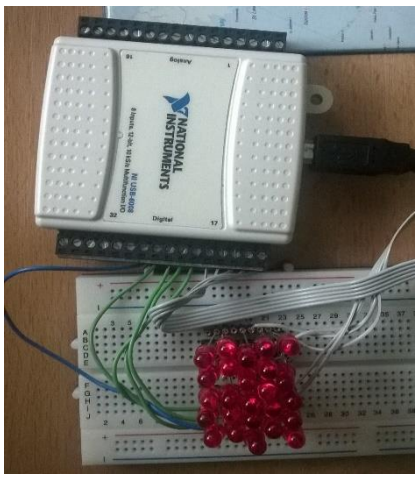

**Obr. 11 Case structure**

## 1.2 Použité karty

USB-6008

První karta, s kterou jsem pracoval, obsahuje 12 digitálních vstupně výstupních kanálů, které postačovaly pro řízení pokusné matice diod 5 x 5 na nepájivém kontaktním poli. Dále disponuje čtyřmi páry analogových výstupů a jedním párem analogovým vstupním. K dispozici jsou také piny pro napájení externích zařízení 5 a 2,5 V DC.

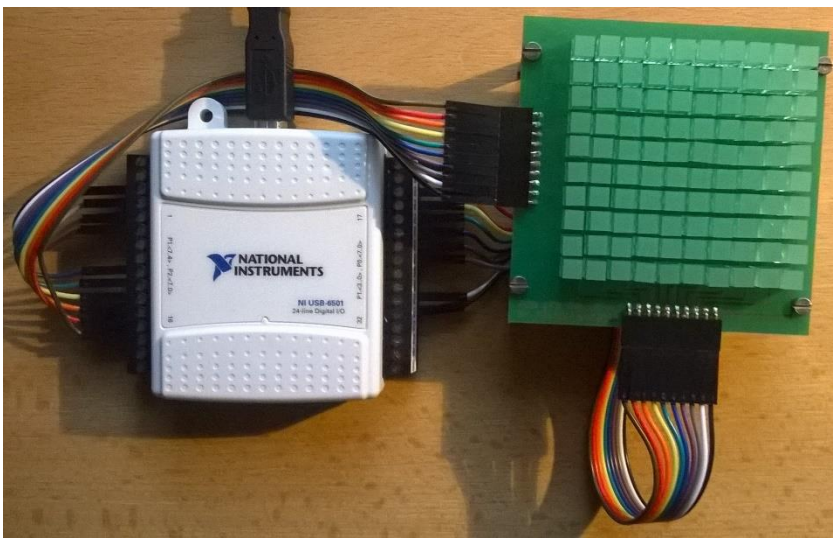
Tato karta se ukázala, jako méně vhodná pro mou aplikaci, jelikož digitální výstupy mohou být pouze ve stavu otevřený kolektor (*open collector)*. A tedy digitální výstup je většinou řízen jedním bipolárním tranzistorem. Kdy ve stavu *low* je tranzistor připojen k zemi a ve stavu *high* je *collector* v plovoucím režimu. Je tedy nutné externí napájení např. přes tzv. *pull-up* (zvyšovací, externí) rezistor. Ve výsledku je na výstupu nedostačující proud pro zřetelné rozsvícení diody bez externího napájení. Dále tato karta nemá dostačující počet digitálních pinů pro ovládání finálního LED pole 10 x 10, kdy je potřeba 20 digitálních kanálů.[7]




**Obr. 12 USB - 6008**

USB-6501

Tato karta obsahuje 24 digitálních kanálů, které lze programově nastavit jako vstupní nebo výstupní. Také disponuje piny pro externí napájení 2 x 5V DC a 6 x GND. Jednotlivé kanály lze nastavit do režimu *open collector* (*open drain*) nebo *active drive*. Zapojení *active drive* je komplementární zapojení dvou bipolárních (unipolárních) tranzistorů, kdy jeden tranzistor např. NPN přivádí *high voltage* (logická 1) na výstup (collector připojen na Vcc + 5V) a druhý tranzistor na výstup přivádí *low voltage* (logická 0, připojen na GND). Na výstupu je poté dostačující proud 10 mA (při konstantním sepnutí) pro zřetelný svit LED diody bez nutnosti externího napájení. Tento proud je také dostačující díky řízení, kdy v jeden časový okamžik svítí pouze jedna dioda a protékající proud se nemusí dělit mezi jednotlivé diody v jednom sloupci. [8]




**Obr. 13 USB-6501**

## 1.3 Principy zobrazovačů

- **Segmentový displej**

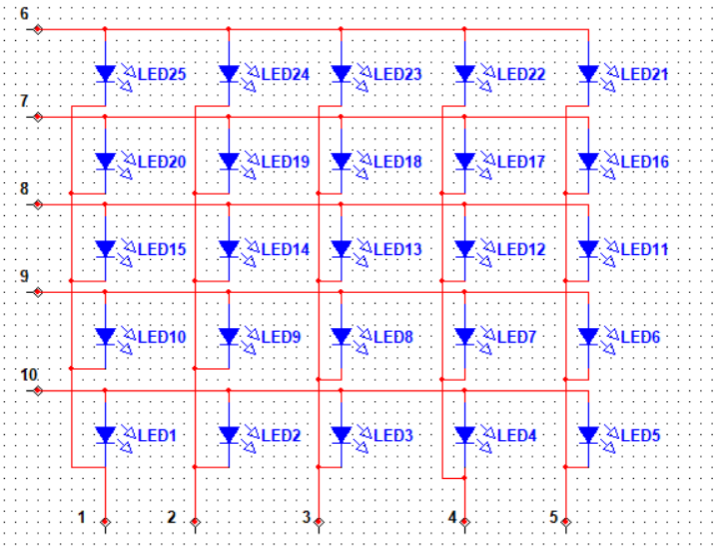
  Je jeden z nejpoužívanějších displejů pro svoji jednoduchost. Využívá se k zobrazování číslic nejčastěji osmisegmentový nebo písmen pomocí šestnáctisegmentového displeje. Jednotlivé segmenty (LED diody) jsou ve vhodné pozici pro výslednou podobu znaku. LED diody lze zapojit se společnou anodou nebo katodou, k zobrazení znaku dochází po přivedení signálu na druhou elektrodu z mikrokontroléru.

- **LCD displej (*Liquid Crystal Display*)**

  Je nejrozšířenější displej s širokým využitím v spotřební elektronice. Využívá se principu elektrostatické polarizace tekutých krystalů. Kdy ve spodní části je zdroj světla (LED diody), následují tekuté krystaly mezi dvěma elektrodami a polarizačními filtry pro jejich polarizaci, tedy aktivaci požadovaných pixelů. Každý pixel se skládá ze tří subpixelů (RGB) pro zobrazení celého spektra barev.

- **Maticový displej**

  Oproti segmentovému displeji má maticový displej větší rozlišení v podobě počtu bodů. Jak plyne z názvu, jedná se o maticové zapojení katod do sloupců a anod do řádků (nebo opačně) viz **Obr. 14**. Pro výsledné zobrazení znaku je zřejmé, že nemůžou být aktivní všechny sloupce nebo řádky v jeden okamžik, jelikož by byli aktivní i nežádoucí diody. Proto je nutné rozsvěcovat postupně příslušné řádky nebo sloupce, případně aktivovat pouze jednotlivé body pole.




**Obr. 14 Ukázka zapojení maticového displeje**

# 2 Popis algoritmu

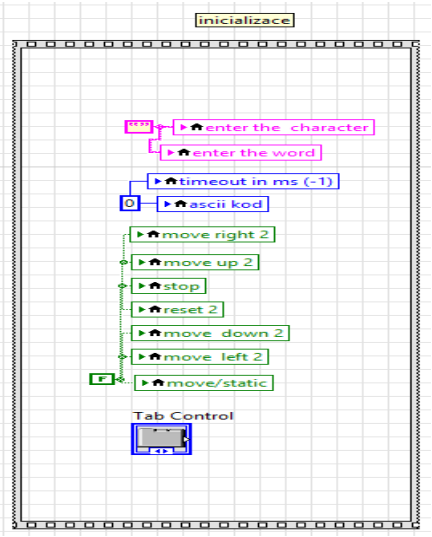
## 2.1 Programové řešení

V počátcích byly načítány binární hodnoty ze souborů a odesílány do karty pro převod logických signálů na požadované hodnoty napětí. Otevírání a načítání hodnot ze souboru velmi zdržovalo celý program a projevovalo se výrazným blikáním LED diod. Binární hodnoty byly poté nahrazeny maticí *boolean* hodnot. Je to naprosto totožné. Toto řešení je více názorné a navíc nepotřebuje konverzi, jelikož vstupní terminál karty vyžaduje právě *boolean* hodnoty.

Dále bylo testováno řízení pomocí dvou karet viz **příloha D**. Kdy jedna karta ovládala anody a druhá katody LED diod. Ukázalo se, že ne vždy pracují synchronně a způsobuje to problémy v podobě nepřesného zobrazení znaků na LED panelu.

Po několika verzích programu se projevil, jako nejvhodnější systém *Producer/Consumer*. Kdy program obsahuje dvě paralelní *while* smyčky. Jejichž synchronizaci zajišťuje tzv. *Queue* neboli fronta. Smyčka *Producer* připravuje data v závislosti na požadavku uživatele. Tedy podle příslušného ascii kódu vybere potřebný znak pro odeslání frontou do smyčky *Consumer*. Smyčka *Consumer* zajišťuje případný posun znaků horizontálně nebo vertikálně v závislosti na požadavku uživatele pomocí podprogramu. Dále tato smyčka zajišťuje pomocí *for* cyklů vybrání dílčích částí matic *boolean* hodnot pro odeslání do karty a jejich následné zobrazení na LED zobrazovači.

## 2.2 Inicializace programu




**Obr. 15 Inicializace programu**

Inicializace programu je nutná. Pokud by uživatel například opětovně spouštěl program, mohlo by dojít k zapsání předchozích hodnot. V blokovém diagramu je obsažen jeden rámec *Flat sequence*. V tomto případě po spuštění běhu *VI* proběhne nejprve právě obsah *Flat sequence*. Kdy dojde k zapsání defaultních hodnot do příslušných lokálních proměnných, respektive do odpovídajících terminálů v různých částech *block diagramu* viz **Obr. 15**. Veškeré části *Front panelu* se taktéž nastaví do defaultního stavu.

## 2.3 Výběr událostí dle požadavku uživatele (*Producer Loop*)

K zpracování požadavku uživatele slouží především událostní struktura (*Event structure)*, která je součástí *while* smyčky: *Producer Loop* (viz **příloha A)**. Kdy v závislosti na stisknutí tlačítka nebo vložení znaku v čelním panelu aplikace dojde k proběhnutí příslušného okna. *Event structure* obsahuje tato okna:

*Enter the charakter* (vložte znak)
Jestliže uživatel vloží znak, dojde k přiřazení ascii kódu prvnímu znaku, který vložil. Toto okno není určeno pro zobrazování slov, a proto dojde k přiřazení pouze prvního znaku. V následující *case* struktuře dojde k výběru příslušné matice *boolean* hodnot dle ascii kódu. Momentálně lze zobrazit velká písmena a číslice, pro zobrazení dalších znaků je nutné pouze předpřipravit příslušné matice. Poté je matice odeslána frontou do *Consumer Loop*. Fronta má dva účely: synchronizaci paralelních smyček a přesun dat metodou FI-FO.

*Enter the word* (vložte slovo)
Na rozdíl od okna *Enter the charakter* toto okno obsahuje navíc *for* cyklus (viz **příloha A)**, kdy je nutné zaznamenat a přiřadit všechny znaky slova. Počet iterací *for* cyklu odpovídá počtu znaků v zadaném slově. Výstupem je zřetězené pole (*concatenated array)*, kdy jednotlivé znaky jsou řazeny za sebou dle zadání uživatele. V tomto případě je nutná transpozice složené matice pro překlopení matice do horizontální polohy, aby bylo možné matici zpracovat. Taktéž je nutné pomocí lokálních proměnných ovládat *SubVI*: Změna posunu znaku, jelikož vzhledem k počtu znaků musí nutně dojít i k jejich posunu pro zobrazení na LED zobrazovači. Pro posun je nutné posun zapnout, zvolit horizontální nebo vertikální posun a definovat velikost výsledné matice.

*Move up, Move down, Move right, Move left*
Tyto okna zajišťují posun znaků i slov dle požadavku uživatele a definují optimální rychlost posunu. Pomocí časovače fronty (*timeout queue)*, která synchronizuje obě smyčky. Například při posunu nahoru dochází k odstranění prvního řádku matice znaku a jeho opětovnému vrácení na konec matice. V případě znaků je nutné mezi znaky vložit prázdný řádek pro snadnou rozlišitelnost znaků. Tyto operace jsou prováděny v *Consumer Loop* v *SubVI*: Změna posunu znaku.

Stop

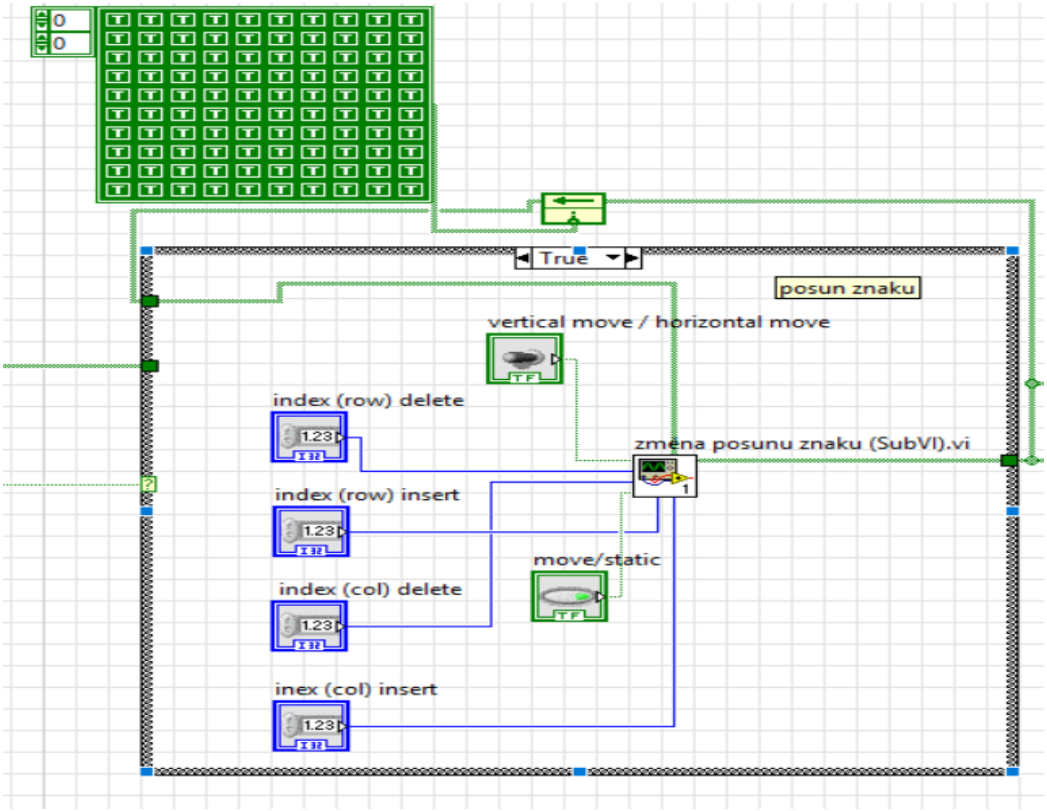
V případě že uživatel stiskne tlačítko stop, dojde k ukončení běhu programu.

Reset

Pokud uživatel stiskne tlačítko reset, odešle se frontou prázdná matice, jelikož nic nebylo zadáno. A tedy nebude zobrazen žádný znak. *Front panel* přejde do defaultního nastavení.

## 2.4 Smyčka pro zpracování dat *(Consumer Loop)*

Stejně jako *Producer Loop* obsahuje tato smyčka hlavní *case* strukturu, která se vykoná, pokud nenastane nějaká chyba např. nepřipojená karta. V případě vzniku chyby dojde k ukončení programu a zobrazení chybové hlášky. *Consumer Loop* viz **Příloha B** obsahuje vnitřní *case* strukturu **Obr. 16**, kdy v jednom případě dojde pouze k proběhnutí dat, jestliže není potřeba znaky posouvat. Pokud uživatel požaduje posouvat znaky, je vnitřní *SubVI* ovládané pomocí hodnot přenesených přes lokální proměnné z *event* struktury. Přenáší se směr posunu a velikost matic. Dále *Consumer Loop* obsahuje dva *for* cykly pro zpracování matic a jejich přípravu pro výsledné zobrazení znaku. Jejichž funkce je podrobně popsána v podkapitole **2.6**.



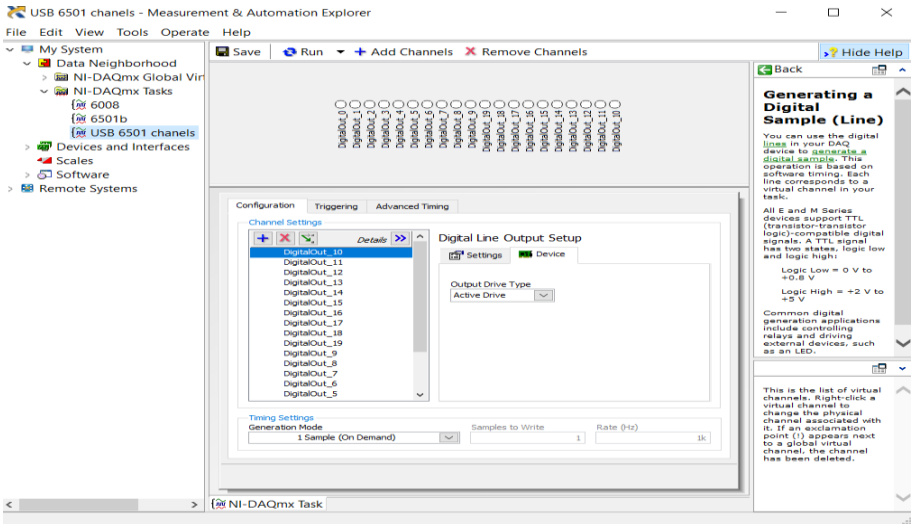

**Obr. 16 Posun znaků (slov)**

## 2.5 Posun znaků nebo slov

Vnitřní *case* struktura smyčky *Consumer Loop* je na obrázku **Obr. 16**. Pokud na podmínkový terminál *case* struktury dorazí hodnota *true* a tedy uživatel požaduje posun znaků, vykoná se rámec *True*. Okno *True* obsahuje *SubVI*: Změna posunu znaku. Princip *SubVI* je, že nejprve dojde k odstranění prvního řádku nebo sloupce matice v závislosti na požadavku vertikálního nebo horizontálního posunu. A následně je odstraněný řádek nebo sloupec vložen na konec matice. Takto změněná matice se v dané iteraci odešle pro zobrazení a zároveň se pomocí zpětné vazby (*Feedback Node)* vrátí zpět na vstup *case struktury* pro další posun následujícího řádku nebo sloupce. Pokud uživatel nepožaduje posun znaků, je na podmínkovém terminálu *case* struktury hodnota *false* a dojde pouze k přesunu matice přes *case structure* bez úpravy.

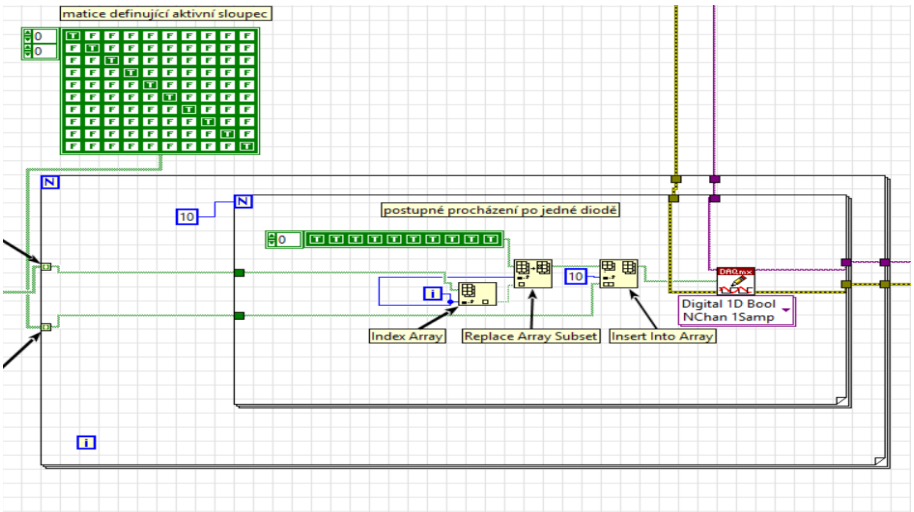
## 2.6 Adresace kanálů karty USB-6501

Vzhledem k nižšímu výstupnímu proudu z digitálního výstupu karty USB-6501, jsem zvolil princip řízení, kdy v jeden časový okamžik svítí pouze jedna LED dioda. Výstupní proud ovlivňuje svítivost diody a při velmi nízkém proudu procházejícím diodou nemusí být znak zřetelný. Toto řešení klade nároky na rychlost karty, která využívá přenosovou rychlost USB 2.0 kabelu 12 Mb/s a také na jednoduchost tzn. rychlost algoritmu případně výpočetní rychlost počítače. Nejprve je nutné vybrat 20 digitálních kanálů z 24 možných. Definovat jejich pořadí, typ vstupní nebo výstupní a režim tzn. *open colector / active drive*. Nejednoduší možnost je použít *DAQ Assistant* tzv. *express VI*. Použití *high level* funkce *DAQ Assistant* přináší nevýhodu v podobě zbytečného načítání karty a uzavírání v každé iteraci programu. Toto lze odstranit použitím *low level* funkcí pro *DAQ* karty. Kdy načtení karty a jejích kanálů se provede pouze při spuštění programu a uzavření po ukončení programu. Proto je vhodné mít předdefinované kanály a nastavení uložit v počítači. Provádí se to v NI-MAX neboli *Measuremnet and Automation Exploler* **Obr. 17**, kde je možné spravovat veškerý hardware ať už připojený nebo nasimulovaný.

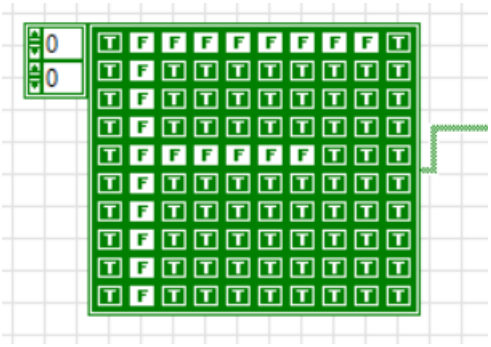



**Obr. 17 NI-MAX**

Pro daný způsob řízení je nutné pracovat s dvěma maticemi. Jedna matice definuje výběr sloupce diodového pole (matice mimo *for* cykly viz **Obr. 18)**. Tato matice má na diagonále hodnoty *true* neboli logická 1, jelikož přivádíme signál na anody diod. Druhá matice obsahuje vybraný znak v podobě *false* hodnot neboli logická 0 pro odeslání na katody LED diod viz **Obr. 19**. Vnější struktura *for* vybírá příslušné řádky z matic, který mají být aktivní v danou iteraci. Vnitřní *for* cyklus o 10 iteracích projde příslušný řádek matice znaku odpovídající řádkům diodového pole (tzn. skutečnému zapojení na plošném spoji). Aby bylo možné aktivovat pouze jednu diodu, je vždy vybrán pouze jeden bod řádku matice pomocí *Index Array*. Vybraný bod je vložen do řádku *true* hodnot pomocí *Replace Array Subset* tzn. danou iteraci je aktivní pouze daný bod, pokud nese hodnotu *false* viz **Obr. 18**, protože odesíláme signál na katody diod. Tedy do *DAQmx write* je odeslána složená 1D matice obsahující 20 hodnot pro příslušných 20 kanálů. Prvních 10 hodnot je určeno pro ovládání řádků diodového displeje, druhá polovina je pro řízení sloupců diodového zobrazovače.



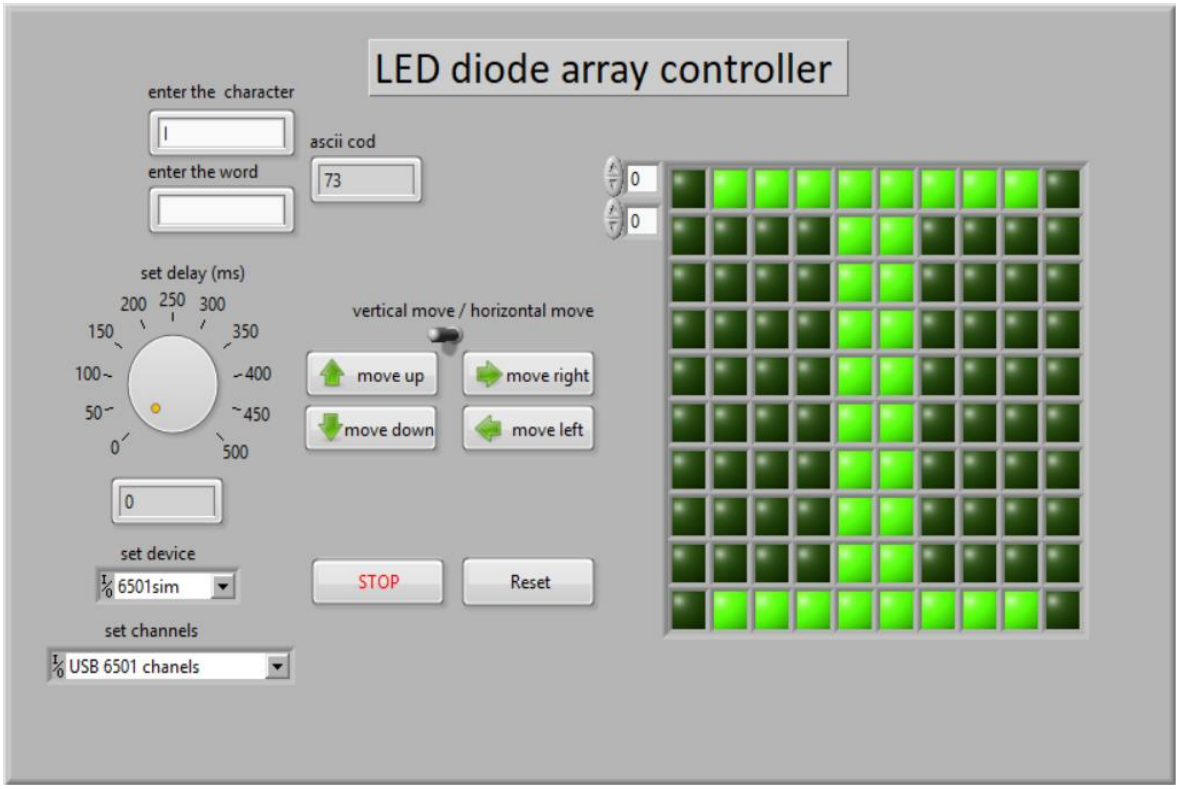

**Obr. 18 Příprava dat pro zobrazení**



**Obr. 19 Matice znaku**

## 2.7 Popis čelního panelu aplikace *(Front panel)*

*Front panel* je uživatelské rozhraní a pro uživatele simuluje vzhled skutečného přístroje. Na **Obr. 20** vidíme rozmístění jednotlivých akčních členů (*controls*) např. okna pro vložení požadovaného znaku nebo slova. Dále je zde i výstup (*indicator*) pro zobrazení ascii kódu. Výstup je také matice virtuálních, čtvercových LED diod pro zobrazení aktuálního dění na skutečném LED zobrazovači. Před spuštěním programu je nutné zadat požitou kartu a nastavení výstupních kanálů karty. Tyto parametry uživatel nastavuje u lišt *set device* a *set channels*. Do okna *enter the character* uživatel vkládá požadovaný znak. Tento znak se zobrazí včetně příslušného ascii kódu. Poté může uživatel zvolit směr pohybu znaku. V případě, že uživatel zadá slovo do okna *enter the word*, nedojde k zobrazení ascii kódu a je nutné vybrat směr pohybu, aby mohlo být celé slovo zobrazeno. Nejprve pákovým tlačítkem uživatel vybere horizontální nebo vertikální směr pohybu a poté specifikuje pohyb nahoru, dolů, doprava nebo doleva. V případě posunu znaku nebo slova může uživatel nastavit rychlost posunu pomocí otočného knoflíku *set delay*, kdy ve skutečnosti nastavuje zpoždění v podobě časovače fronty. Zpoždění je nutné nastavit s rozvahou, protože při velkém zpoždění může dojít k nespojitosti znaku. Vzhledem k řízení diodové matice by se při nastavení příliš velkého zpoždění zobrazoval znak po jednotlivých bodech, případně by se nezobrazil. Pokud uživatel nebude měnit zpoždění, program nastaví optimální hodnotu. Dále uživatel může zastavit program (VI) pomocí tlačítka Stop, nebo resetovat nastavení do defaultního stavu pomocí tlačítka Reset.




**Obr. 20 Front panel**

# 3 Výběr LED diod a zapojení diodového pole

## 3.1 Výběr LED diod

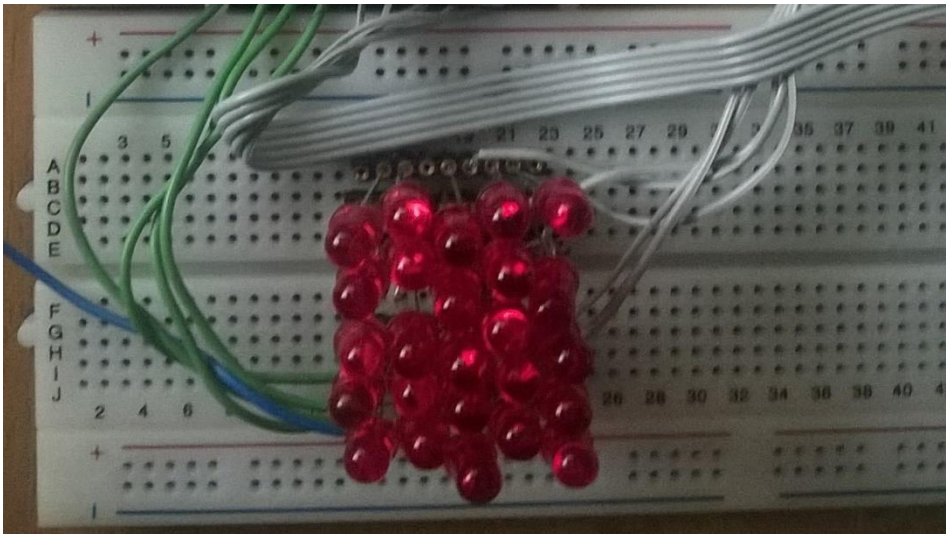
LED dioda

Polovodič obsahující jeden PN přechod. Při přiloženém napětí v propustném směru větším než 0,7 V pro (Si) přechází dioda do vodivostního stavu, dochází k tzv. generaci, kdy valenční elektrony přechází do vodivostního pásu. Velikost potřebné energie dodané do PN přechodu je úměrná použitým příměsím pro vytvoření nevlastního polovodiče a tedy šířce zakázaného pásu. Šířka zakázaného pásu (dáno materiálem příměsi) udává výslednou vyzařovanou vlnovou délku (barvu LED diody) při rekombinaci.

Při prvních testech jsem pracoval s vývodovými LED diodami s oválným pouzdrem. Při testech se ukázalo, že větší vyzařovací úhly LED diod mohou mít vliv na prosvěcování okolních LED diod tedy nečitelnost znaku. V případě oválných pouzder je poměrně velký výběr vyzařovacích úhlů a svítivostí LED diod. Při pokusech bylo nutné zvolit přiměřenou svítivost LED diod, aby nedocházelo k oslnění pozorovatele.

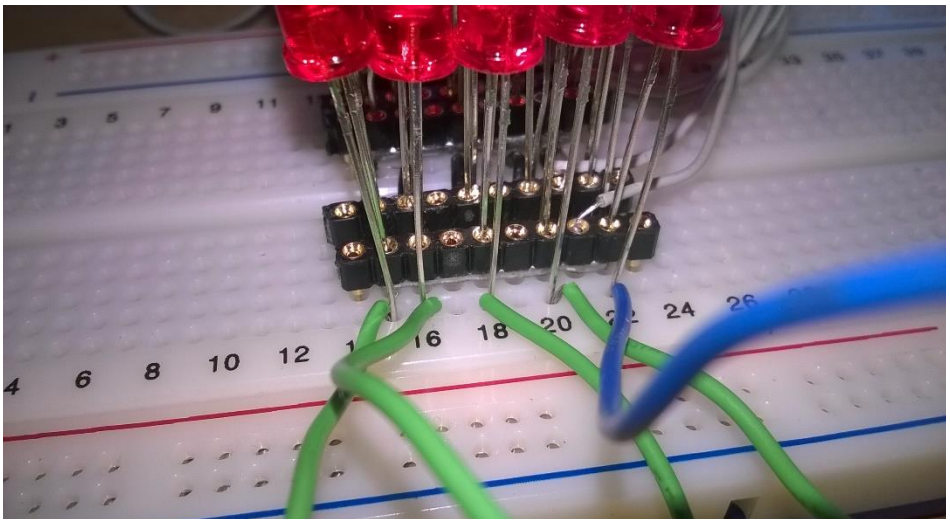
Pro návrh LED pole jsem zvolil vývodové LED diody s čtvercovým pouzdrem 5 x 5 mm, protože jejich rozmístění a případné stínění je jednoduší než u oválných pouzder. Vybral jsem LED diody: L-1553GDT (zelená) a L-1553IDT (červená). V případě čtvercových pouzder není velká nabídka vyzařovacích úhlů. Vzhledem k rozmístění diod, kdy jsou mezi nimi mezery případně stínění. Nedochází k ovlivnění (prosvícení) vedlejších LED diod. Obě LED diody mají téměř totožné parametry: $If_{max}$ = 25 – 30 mA, $Uf_{max}$ 2,5 V, $I_R$ = 10 µA, $U_R$ = 5 V, vyzařovací úhel α = 110°. S rozdílnou svítivostí I = 8 mcd (červená), I = 5 mcd (zelená).

## 3.2 Zkušební zapojení LED pole



**Obr. 21 Zkušební LED pole**

Nejprve bylo nutné navrhnout zapojení, na kterém by bylo možné otestovat příslušný algoritmus. K dispozici jsou tři možnosti. A to: 1) spojení všech anod a matici ovládat pomocí jednotlivých katod, 2) spojení všech katod a ovládání pomocí anod, 3) spojení katod do řádků a anod do sloupců (nebo obráceně). Poslední zapojení výrazně redukuje počet nutných vodičů respektive kanálů karty na 20 při ovládání diodové matice o počtu 100 LED diod. Příslušná LED dioda se rozsvítí po přivedení kladné hodnoty neboli *high voltage* na její anodu současné s přivedením *low volatge* v našem případě 0 V na její katodu. Hodnoty *high* a *low voltage* jsou reprezentovány v *block diagramu* hodnotami *true* a *false*. Pro prvotní testování jsem zvolil nepájivé kontaktní pole, které je vhodné pro vývojové úpravy. Pro realizaci maticového zapojení je však nedostačující, a proto bylo nutné sloupcové propojení katod realizovat externě single in line paticemi s propájenými vývody viz **Obr. 22**. Propojení anod do řádků bylo realizováno pomocí kontaktů nepájivého pole.




**Obr. 22 Detail zapojení matice diod na nepájivém poli**

## 3.3 Návrh plošného spoje

Po provedení testů a odladění algoritmu bylo potřeba navrhnout plošný spoj, protože nepájivé pole při manipulaci není stabilní a dochází k značnému namáhání vývodů diod. Zejména propojení vývodů nemusí být vždy plnohodnotné. Plošný spoj jsem navrhl oboustranný s prokovenými otvory pro lepší přiletování vývodů diod. Kdy na jedné straně plošného spoje je propojena vždy jedna skupina vývodů anod nebo katod, abych nemusel realizovat přemostění křížených cest. Plošné spoje jsem navrhl v studentské verzi programu EAGLE. V **příloze G** je vidět zapojení LED diod s vyvedením na zásuvkovou lištu. Dále zde vidíme rozmístění LED diod na plošném spoji, kdy jednotlivé čtvercové diody 5 x 5 mm jsou od sebe vzdáleny 1,25 mm pro možnost vložení stínění. Navržená deska plošného spoje neobsahuje ochranné rezistory, jelikož u použité karty není velký výstupní proud a tudíž nemůže dojít k zničení LED diod. Po dokončení návrhu jsem vygeneroval soubory nutné pro výrobu. Výrobu jsem zadal firmě, jelikož nemám k dispozici potřebnou technologii. Příslušné LED diody včetně zásuvkových lišt jsem poté přiletoval na plošný spoj.

## 3.4 3D tisk [5]

3D tisk je stále vyvíjející se technologie, která je v poslední době cenově dostupná nadšencům do 3D tisku i domácnostem. Trojdimenzionální tisk umožňuje převedení modelu připraveného v některém z CAD programů do skutečného modelu. V závislosti na použité technologii tisku je možné rychle případně i levně vytisknout a odzkoušet funkční prototypy před zahájením sériové výroby.

### 3.4.1 Technologie 3D tisku

*SLS* – Selektivní laserové slinování *(Selective Laser Sintering)*

Tato technologie využívá k vytvrzení materiálu přesně mířený laserový paprsek. Pro stavbu modelů se používá jemný práškový plast. Po osvícení Laserem je práškový materiál spečen. SLS je vhodné pro rozměrné modely. Pořizovací náklady na technologii jsou velmi vysoké v řádech milionů korun. Tento způsobu přípravy modelů se taktéž nazývá *Rapid Prototyping*.

*FDM – Fused Deposititon Modeling*

**Obr. 23 3D tiskárna s technologií FDM [6]**

Cenově dostupnější a v dnešní době nejrozšířenější technologií je FDM. K stavbě modelu dochází nanášením nataveného vlákna (reaktoplastu) na podložku. Výsledná barva modelu je dána barvou použitého vlákna. Materiál je postupně nanášen po vrstvách od 0,127 mm do 0,3 mm dle typu tiskárny. Vlákna mohou být dvojího typu stavební a podpůrné. Podpůrný materiál složí k zamezení hroucení stavěného modelu a po dokončení modelu jej lze odstranit. Materiály používané FDM technologií tisku odolávají teplotám až 190 °C a výsledné modely dosahují velké pevnosti. Touto metodou tisku byly vytisknuty i konstrukční části LED pole. Tisk spodního krytu desky plošného spoje trval přibližně 2,5 hod a spotřebovalo se cca 60 g materiálu. V **příloze F** jsou vyobrazeny 3D návrhy jednotlivých konstrukčních částí: spodní kryt plošného spoje a stínící rámeček.

*DLP* – Vytvrzení světelným zářením (*Digital Light Projection)*

Tato technologie využívá jako stavební materiál fotopolymery, které jsou vytvrzovány ultrafialovým zářením. Tloušťka jednotlivých vrstev je cca 0,1 mm. Nevýhodou této metody je nízká teplotní odolnost, kdy už od 50 °C může docházet k změnám struktury modelů.

Inkjetové technologie

Principem je vstřikování stavebního materiálu na bázi vosků na pracovní plochu pomocí trysek. Materiál pro stavbu a podporu je odlišný. Podpůrný materiál má nižší teplotu tání, a proto jej lze odstranit ohřátím výsledného modelu. Tento způsob nanášení materiálu je nejpřesnější s přesností až 0,01 mm. Obdobně je možné nanášet také fotopolymery vytvrzované ultrafialovým zářením.

LOM – *(Laminated Object Manufacturing)*

Výstavba modelu pomocí vrstvení (laminování) desek materiálu

Plošná laminace deskových spojů je založena na principu vyřezávání vrstev z plátu materiálu např. PVC a jejich vrstvení a lepení na sebe. Vyříznutí materiálu je provedeno pomocí vyřezávacího nože nebo laserového paprsku. Tato metoda se nehodí pro tisk členitých částí. Výsledný model je po dokončení křehký a vlivem mechanické separace z podpůrných částí může dojít k jeho poškození.

# 4 Možnosti rozšíření úlohy

Vytvořený program zobrazuje ascii znaky a slova zadaná z klávesnice. Umožňuje jejich editaci v podobě výběru směru a rychlosti posunu. Další možné rozšíření úlohy:

1. Zobrazení grafiky

   Do budoucna by mohlo být možné rozšířit program o převod obrázků na potřebné logické signály pro jejich zobrazení na LED poli. Případně provádět animace těchto obrázků.

2. Vytvoření trojrozměrného zobrazovače

   Další nadstavbou by mohlo být zobrazení a řízení 3D grafiky (objektů). S řízením pomocí prostředí *LabVIEW* a hardwaru od *National Instruments*. Toto rozšíření by vyžadovalo vytvořit nový zobrazovač v podobě 3D LED krychle a jiné. Samozřejmě i vytvoření odlišné struktury řídícího algoritmu.

# Závěr

Podařilo se navrhnout a odzkoušet funkční algoritmus v prostředí *LabVIEW* pro zobrazování znaků a slov s využitím dostupného hardwaru. Navržený systém umožňuje zvolit směr a rychlost posunu znaků zadaných z klávesnice počítače v závislosti na požadavku uživatele.

Během práce jsem se potýkal s několika problémy, které jsem postupem času vyřešil. Nepájivé kontaktní pole, které bylo při manipulaci nestabilní, jsem nahradil navrženým plošným spojem. Vytvořil jsem dva prototypy, kdy jeden je včetně stínícího rámečku, pro eliminaci prosvícení sousedních LED diod a tedy zamezení nečitelnosti znaků. V závislosti na použitém hardwaru je možné odzkoušet obě varianty. Odzkoušel jsem různé varianty algoritmů a v závislosti na rychlosti běhu kódu jsem vybral nejvhodnější.

Během této práce jsem poprvé pracoval s 3D tiskárnou, kdy stínící rámeček a spodní kryty plošných spojů jsou vytištěny na 3D tiskárně.

# Seznam literatury a informačních zdrojů

[1] VLACH, Jaroslav, HAVLÍČEK, Josef a VLACH, Martin. *Začínáme s LabVIEW Vyd.* 1. Praha: BEN - technická literatura, 2008. 247 s. ISBN 978-80-7300-245-9.

[2] PECHOUŠEK, Jiří. Základy programování v prostředí LabVIEW. Vyd. 1. Olomouc: Univerzita Palackého v Olomouci, 2004. 84 s. ISBN 80-244-0800-7.

[3] ŽÍDEK, Jan. Grafické programování ve vývojovém prostředí LabVIEW [pdf]. Ostrava: Katedra elektrických měření, Fakulta elektrotechniky a informatiky, VŠB-TU Ostrava, 2002. 215 s. Dostupné z: http://autnt.fme.vutbr.cz/lab/FAQ/labview/VI_Skripta.pdf

[4] National Instruments. LabVIEW. [online]. [Cit 22.2.2018] Dostupné z: http://czech.ni.com/labview

[5] PKmodel. [online]. [Cit 19.4.2018] Dostupné z: http://www.pkmodel.cz/images/geobusiness-2012-01-3Dtisk.pdf

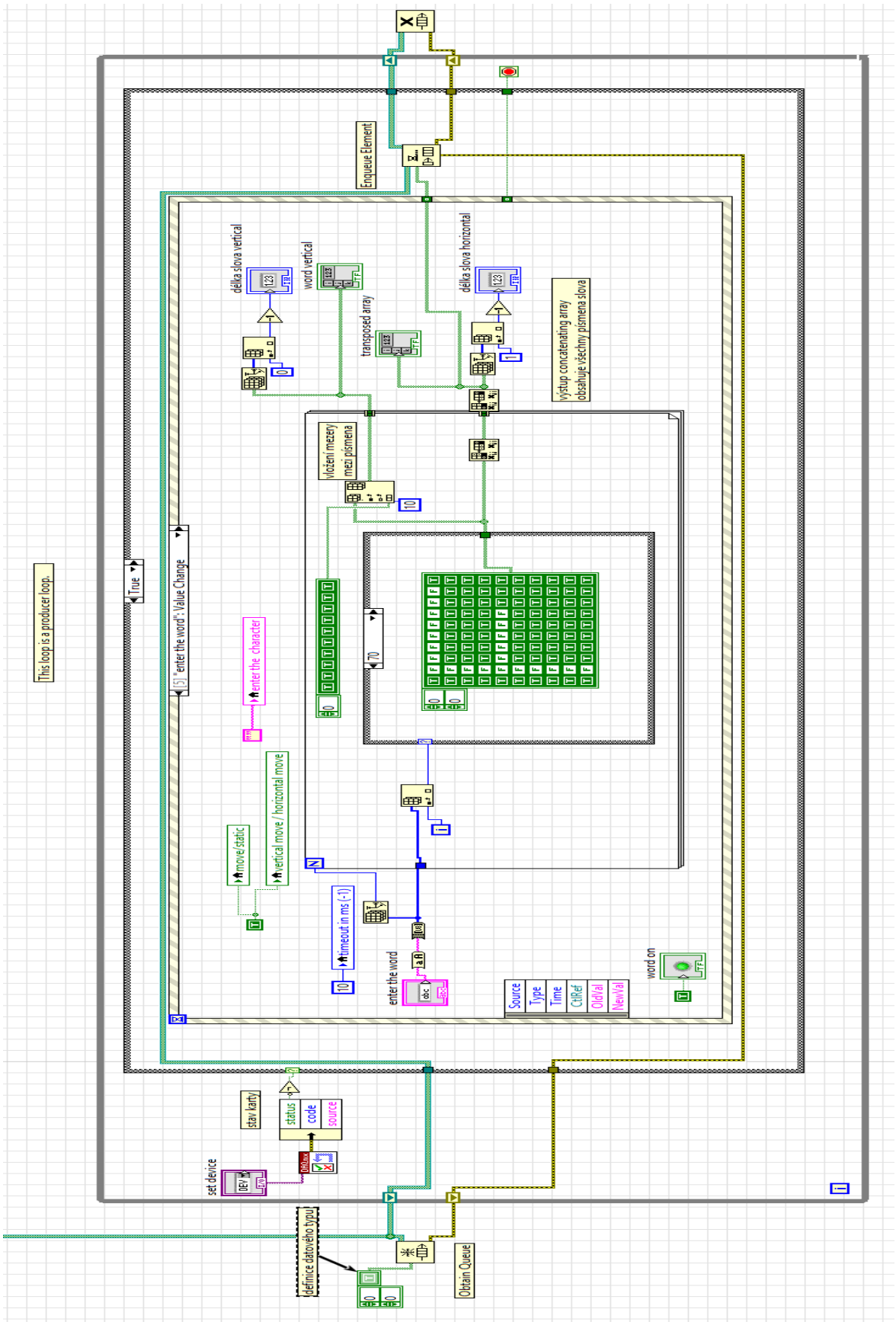
[6] Prusa3d. [online]. [Cit 19.4.2018] Dostupné z: https://www.prusa3d.cz/josef-prusa-announced-release-new-3d-printer-original-prusa-i3-mk2/frontpage

[7] National Instruments. LabVIEW. [online]. [Cit 26.4.2018] Dostupné z: https://forums.ni.com/t5/LabVIEW/What-s-the-difference-between-active-drive-and-open-collector/td-p/3262193

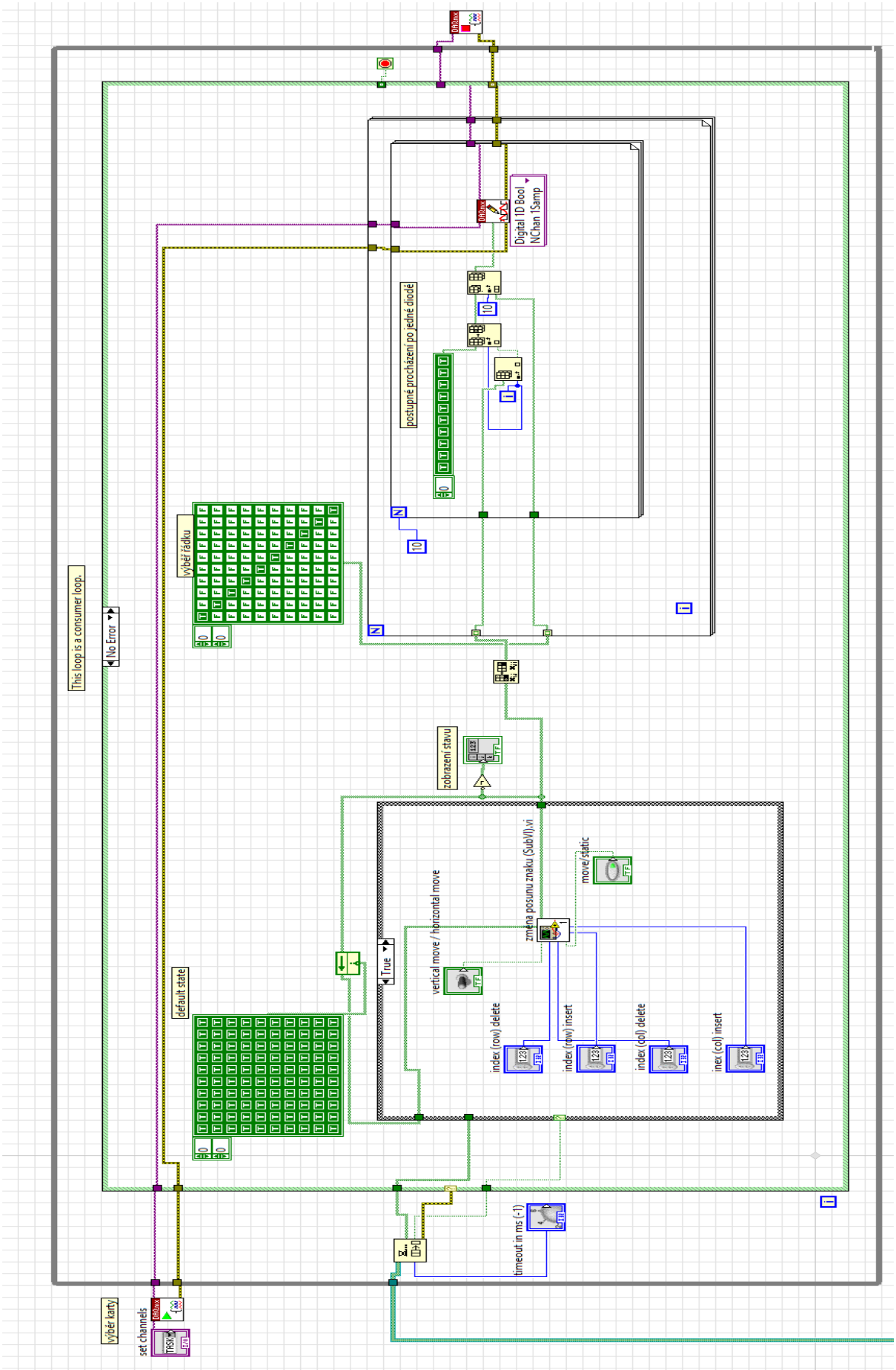
[8] National Instruments.USB-6501 specifications. [online]. [Cit. 26.4.2018] Dostupné z: http://www.ni.com/documentation/en/digital-io-device/latest/specs-usb-6501/specs/

# Přílohy

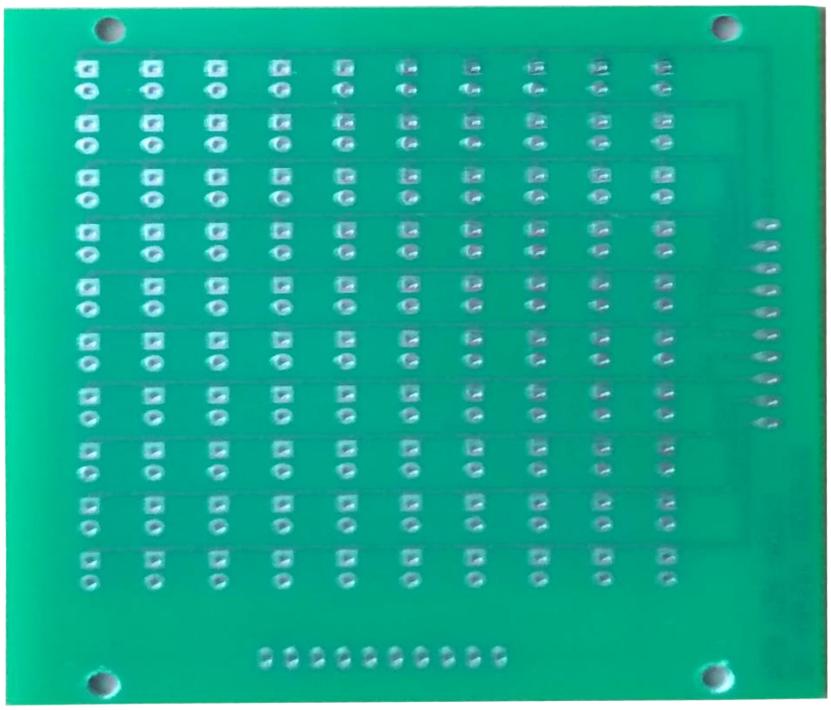
## Příloha A – *Producer Loop*



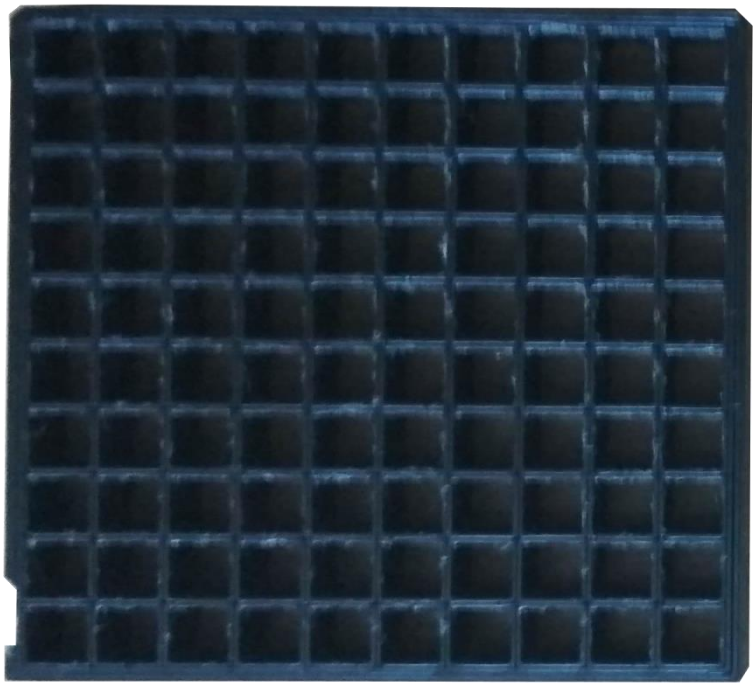
**Příloha B – *Consumer Loop***



## Příloha C – Části LED zobrazovače



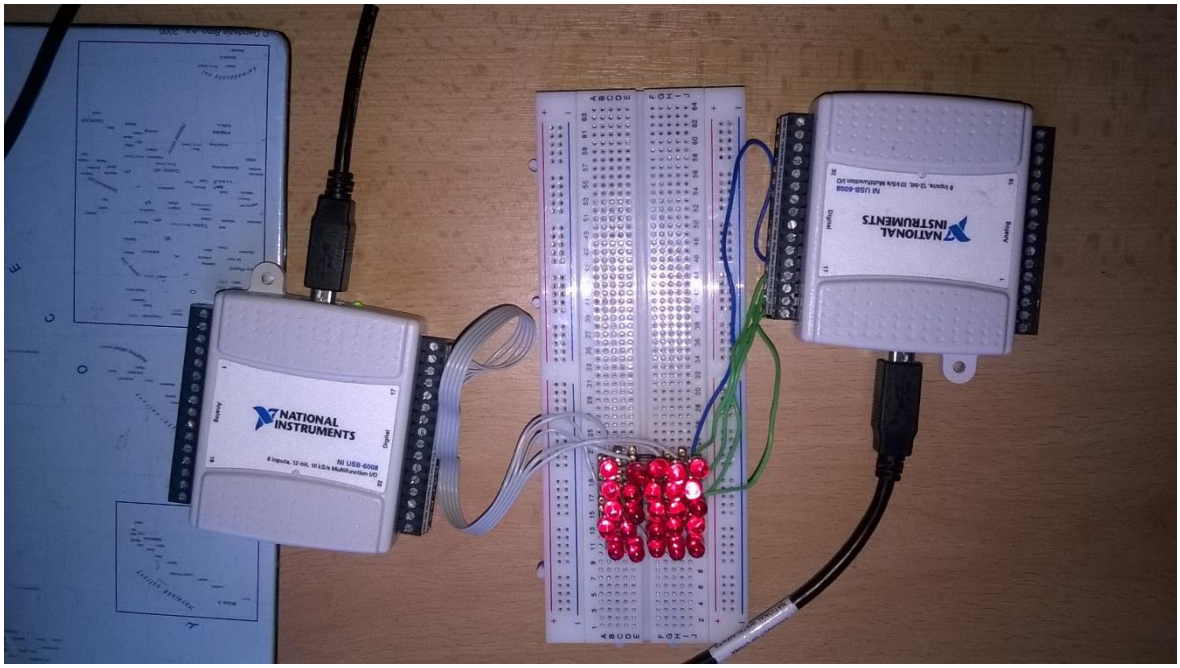
**Obr. 24 Navržená deska plošného spoje**



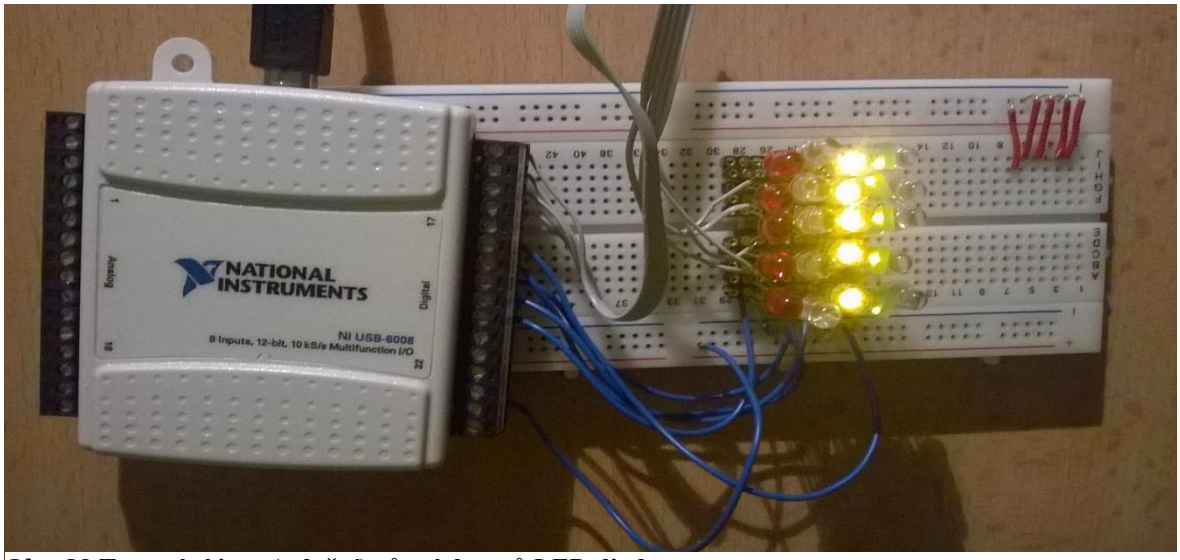
**Obr. 25 Stínící rámeček**

**Obr. 26 Spodní kryt desky plošného spoje**

## Příloha D: Testovací verze LED pole

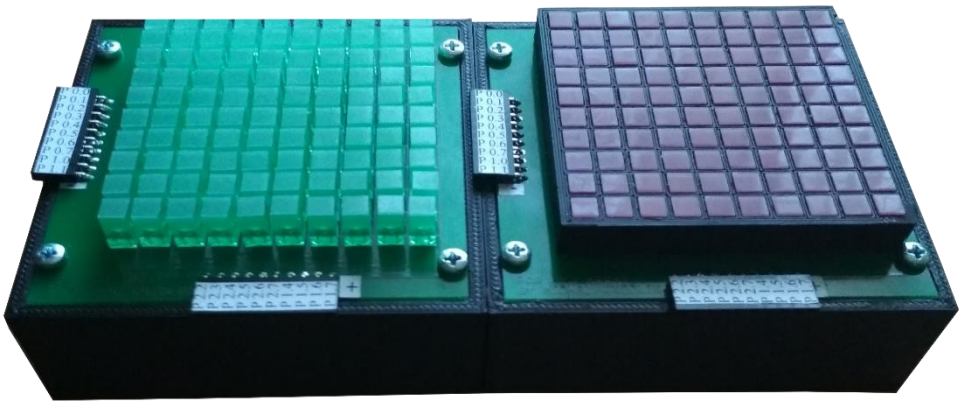


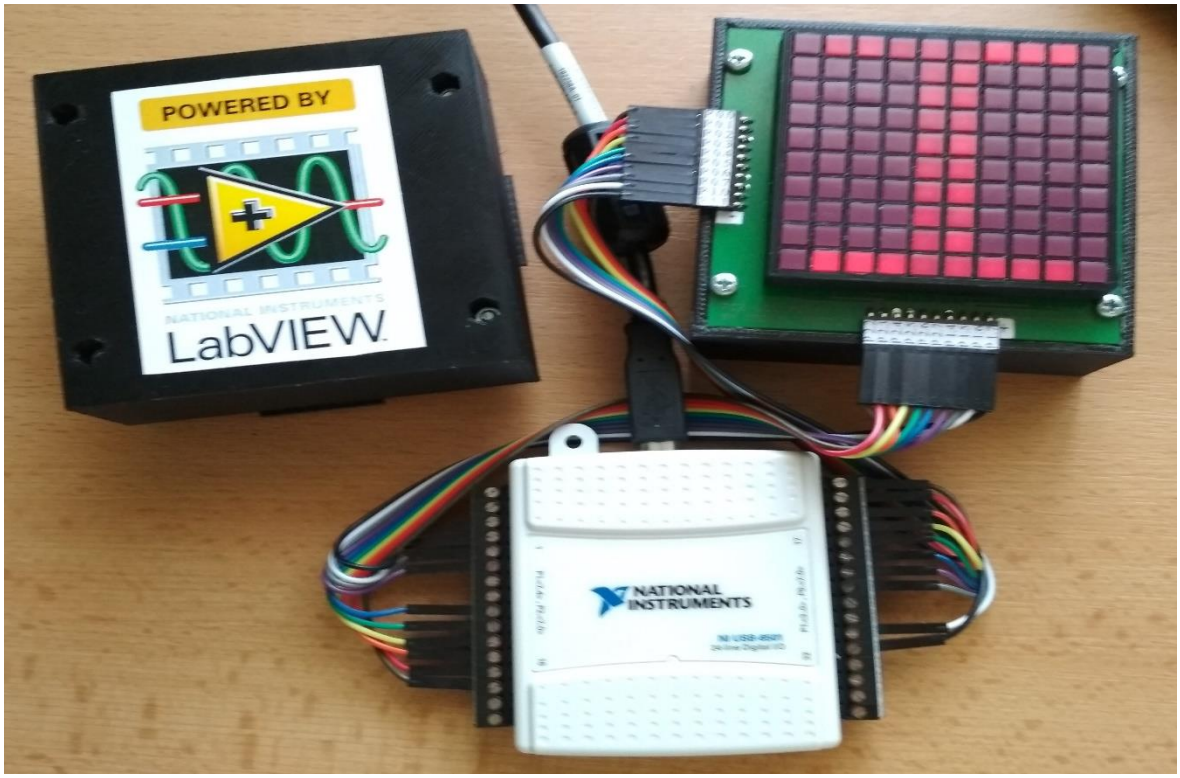

**Obr. 27: Testovací zapojení s řízením pomocí dvou karet**



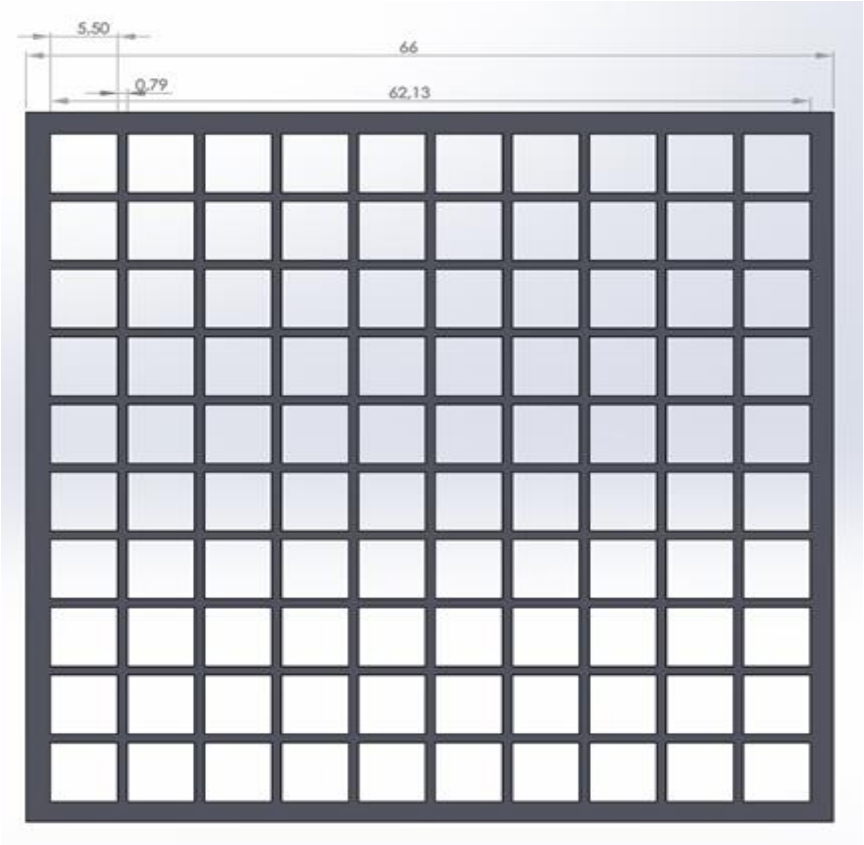

**Obr. 28 Testování jasu (oslnění) různých typů LED diod**

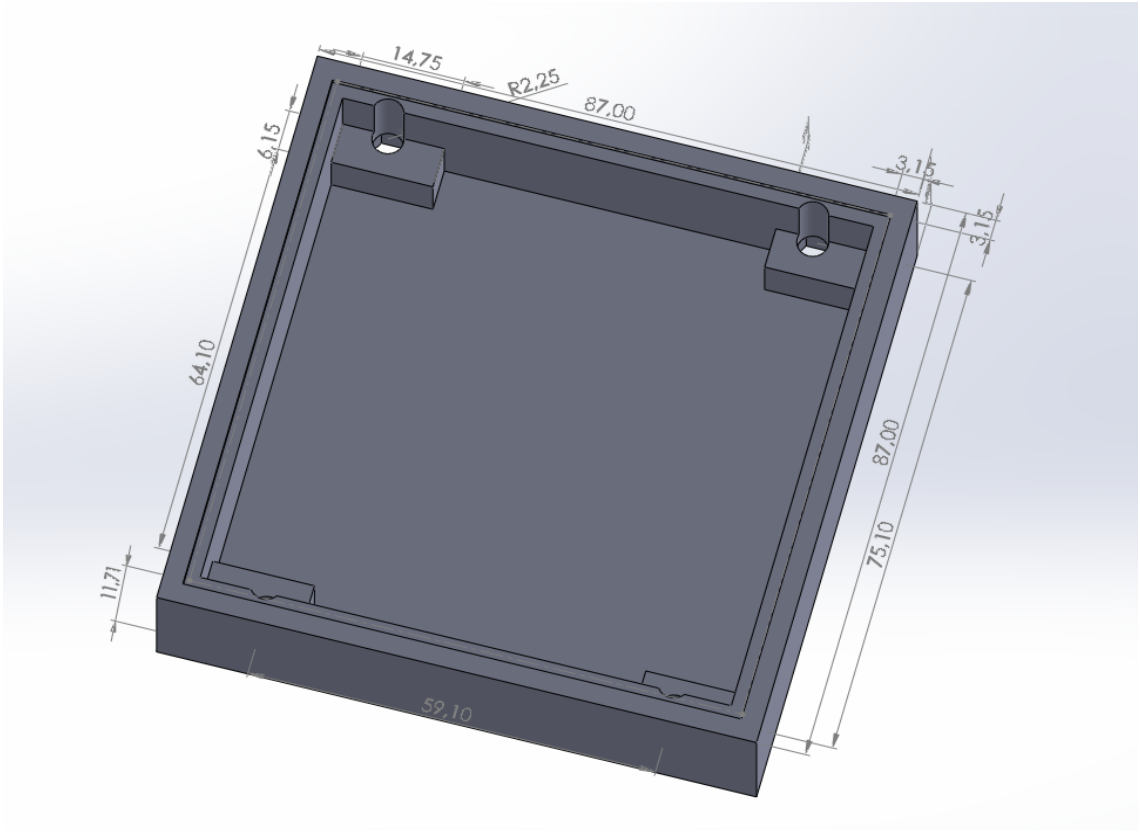
## Příloha E – finální podoba LED zobrazovače

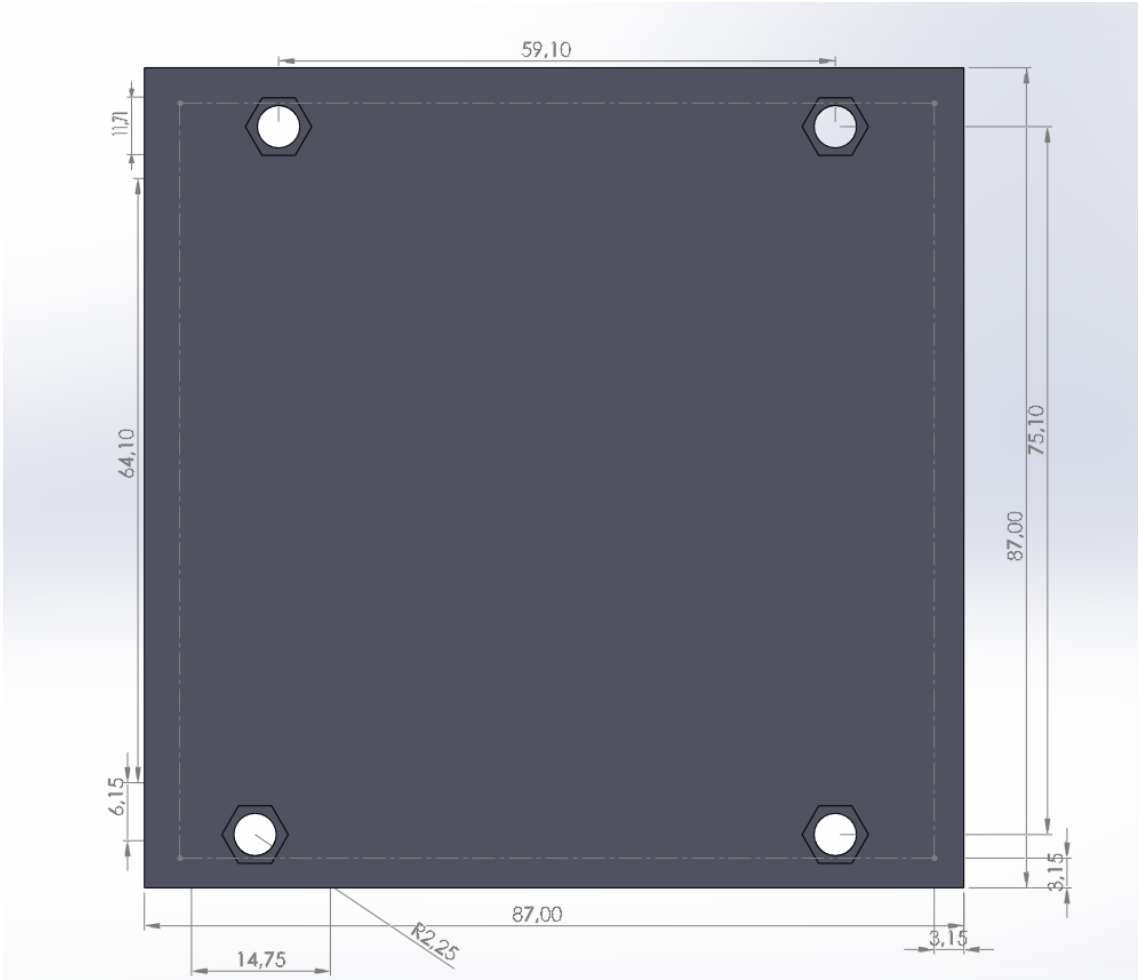




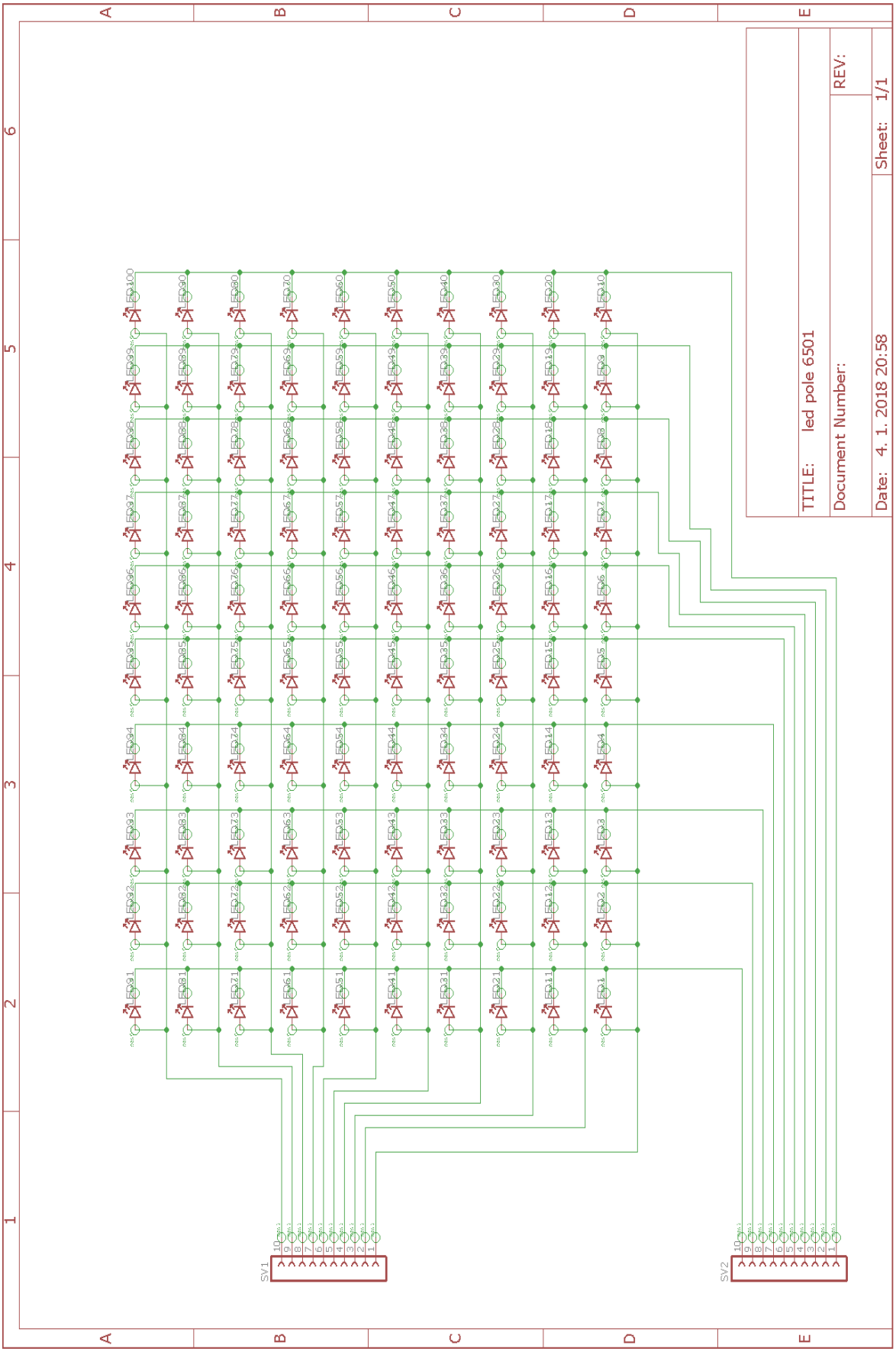

**Příloha F – Návrh stínícího rámečku + návrh spodního krytu pro 3D tisk**



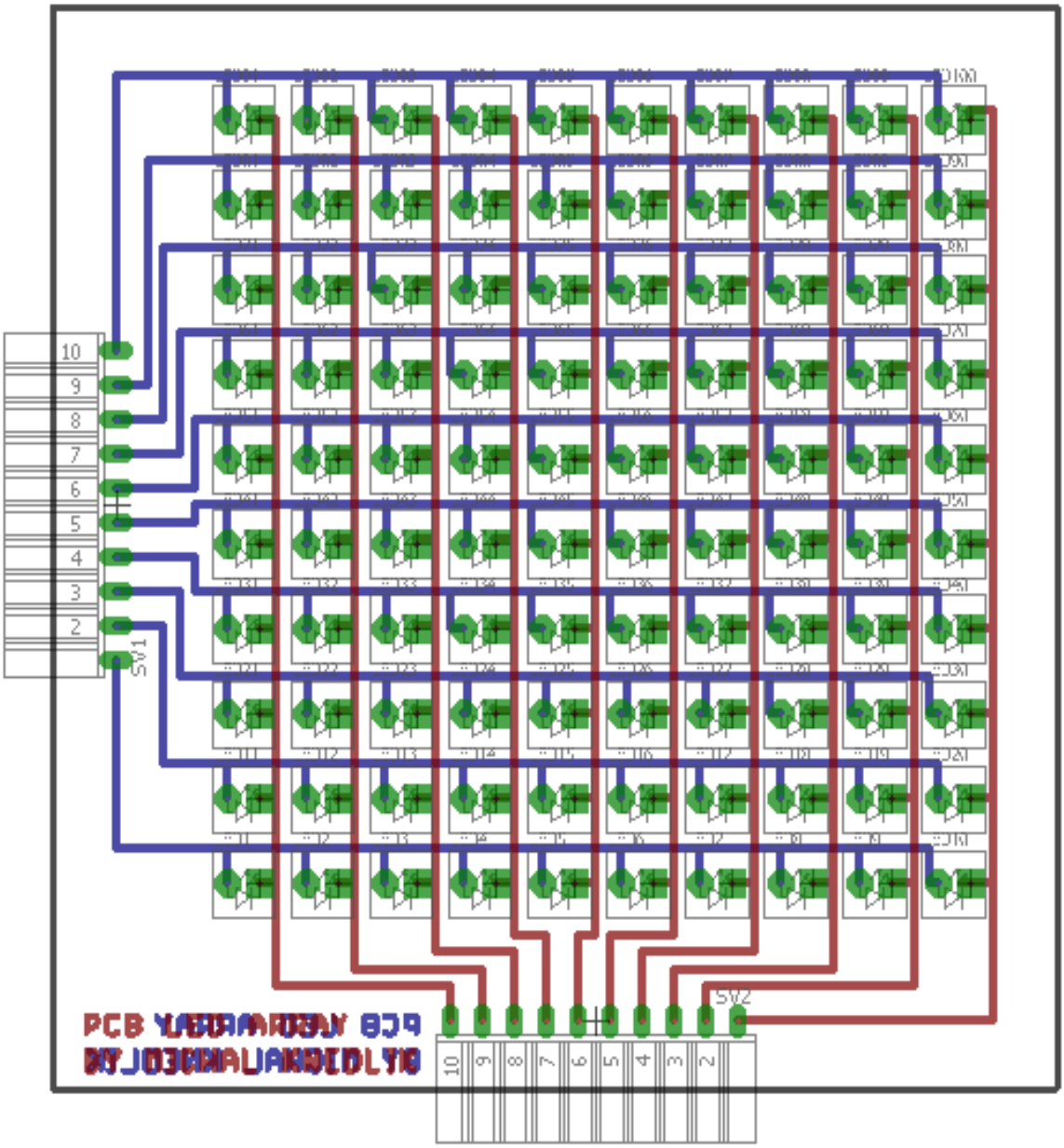

14,75
R2,25
87,00
6,15
3,15
3,15
64,10
87,00
75,10
11,71
59,10


59,10
11,71
64,10
75,10
87,00
6,15
3,15
87,00
3,15
R2,25
14,75

**Příloha G – schéma zapojení + rozmístění součástek na desce plošného spoje**




**Obr. 29 Schéma zapojení v systému EAGLE**




**Obr. 30 Rozmístění součástek na plošném spoji**