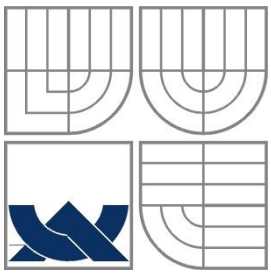
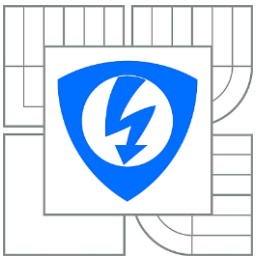


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘÍCÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

BAREVNÝ 3D LED ZOBRAZOVAČ

3D LED COLOR DISPLAY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN RAK
AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ JÍLEK
SUPERVISOR

BRNO 2015

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: Jan Rak
Ročník: 3

ID: 147645
Akademický rok: 2014/15

NÁZEV TÉMATU:

Barevný 3D LED zobrazovač

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte barevný 3D maticový LED zobrazovač ve tvaru krychle pro robot FEKTBot a modul pro jeho napájení z rozvodu 24 V. Samotný zobrazovač bude sestaven z RGB LED. Řídicí modul bude umožňovat komunikaci s nadřazeným systémem, ze kterého bude možné ovládat základní funkce zobrazovače a provést základní diagnostiku jeho stavu. Po zapnutí bude zobrazovač fungovat v automatickém režimu, ve kterém zobrazuje animace, které jsou součástí firmwaru. Po přepnutí do uživatelského režimu umožní zobrazovat externě generovanou animaci. Po oživení celého zobrazovače včetně napájecího modulu bude provedena jeho instalace na robot FEKTBot.

1. Vytvořte rešerši na téma trojrozměrných maticových LED zobrazovačů.
2. Navrhněte a realizujte řídicí modul.
3. Implementujte software pro řídicí modul.
4. Navrhněte a realizujte napájecí modul.
5. Celý zobrazovač zkompletujte do samostatně funkčního zařízení.
6. Ověřte funkčnost zobrazovače včetně napájecího modulu.
7. Osaďte kompletní zobrazovač a napájecí modul do robotu FEKTBot.

DOPORUČENÁ LITERATURA:

Váňa, V. ARM pro začátečníky. BEN - technická literatura, 2009. ISBN 978-80-7300-246-6.

Termín zadání: 9. 2. 2015 ***Termín odevzdání:*** 25.5.2015

Vedoucí práce: Ing. Tomáš Jílek
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cílem této práce bylo navrhnout a zkonstruovat LED kostku pro zobrazování vizualizací a upoutání pozornosti. S ohledem na zadání bylo stanoveno rozlišení 8x8x8. Pro celkové řízení a komunikaci s PC byla zvolena vývojová deska STM32F4 Discovery, která díky svým parametrům plně postačuje pro tuto aplikaci. Kostka tvořená z diod a napájecích vodičů bude usazena na podstavci. Plochými vodiči bude připojena k desce s LED budiči, která bude obsahovat veškeré řídicí a komunikační členy. Implementovaný firmware bude mít v základu animace, taktéž jej bude možné řídit nadřazeným systémem. Podstavec bude potáhnut tkaninou pro lepší vzhled.

Klíčová slova

LED kostka, STM32F4 Discovery, Step-down regulátor, STP16DP05, USART, SPI

Abstract

The goal of this project is to design and construct a LED cube for visualization and imaging attract attention. With respect to the assignment was determined resolution 8x8x8. For the overall management and communication with the PC was chosen development board STM32F4 Discovery, which, thanks to it parameters, is sufficient for this application. Cube consisting of diodes and power wires will sit on the governing board, which will include all parts of management and communication. Implemented firmware will have a base animations, it is also possible to control by superior system.The board will be covered by colored fabric for a better look.

Keywords

LED CUBE, STM32F4 Discovery, Step-down regulator, STP16DP05, USART, SPI

Bibliografická citace

RAK, J. *Barevný 3D LED zobrazovač*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2015. 47s. Vedoucí bakalářské práce Ing. Tomáš Jílek.

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma: Barevný 3D LED zobrazovač, jsem vypracoval samostatně pod odborným vedením a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom 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 autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Tomáši Jílkovy za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne 25. Května 2015 .....
podpis autora

Obsah

Seznam obrázků

1 Úvod

Cílem bakalářské práce je navrhnout a vytvořit mechanickou a elektronickou konstrukci 3D LED kostky, která po uvedení do chodu spustí prezentační režim s několika animacemi. Tyto animace budou navrženy tak, aby zaujaly pokud možno, co nejvíce lidí a splnily, tak hlavní cíl a smysl bakalářské práce, lákat pozornost kolemjdoucích.

Při realizaci je kladen důraz především na velikost a estetičnost. Tyto dva faktory shledávám jako rozhodující, pro získání pozornosti kolemjdoucích. Samozřejmě spolu s poutavými animacemi. Velikost samotné kostky, z hlediska rozlišení byla stanovena na 8x8x8, tato volba byla učiněna s ohledem na poměr celkové ceny a výkonu.

3D LED kostka bude umístěna ve vrchní části robotu FEKTBOT, jehož se po realizaci, stane součástí. Kostka bude umístěna přibližně v úrovni očí, tak aby bylo možné lehké zpozorování kolemjdoucími.

Při návrhu LED kostky je nutné zaměřit se, na její konstrukční řešení a následné řízení. Diody byly vybrány jako RGB, díky této volbě bude možně rozsvěcovat každou diodu jinou barvou, a tím bude docíleno zajímavějších a efektivnějších animací, tak aby přitahovaly pozornost procházejících. Z hlediska funkčnosti, bude kostka komunikovat s počítačem přes sériovou linku a bude jí ovládána. Dále je nutné vytvořit řídicí desku a implementovat jednotlivé řídicí členy, pro ovládání vizualizace kostky. Její ovládání by mělo být snadné i pro laika.

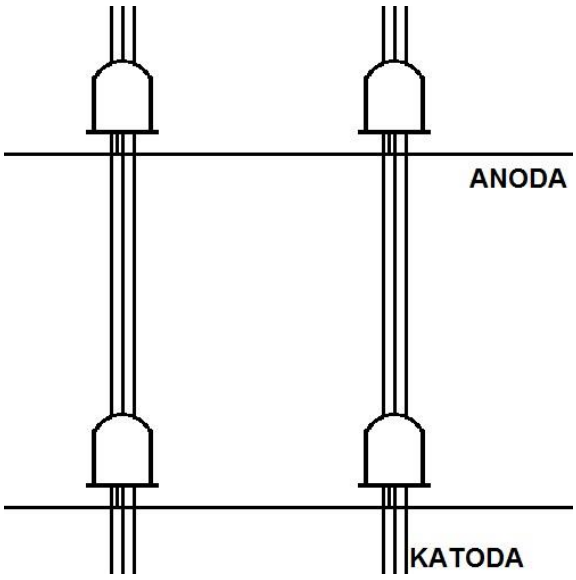
2 Teoretické poznatky

2.1 Rešerše LED kostky

V rámci řízení kostky se musíme zaměřit na její konstrukci a zapojení jednotlivých diod. Kdybychom chtěli jednotlivě řídit každou diodu, musel by být ke každé diodě připojen vodič na uzemnění - v případě jednobarevného provedení. Rám by mohl být tvořen vodič pro napájení. Vícebarevné provedení by toto zapojení diod ještě více komplikovalo. Uvnitř konstrukce, by pak vedlo velké množství vodičů. Což by mělo za následek snížení viditelnosti vnitřních diod a z estetického hlediska by kostka vypadala velmi chaoticky. Z tohoto důvodu se volí koncepce zapojení pro multiplexaci. Tato volba, sebou ovšem nese jistá omezení velikosti kostky.

Při velikosti kostky o délce hrany sto, by jedno patro bylo rozsvíceno setinu zobrazovací periody. Tomu by se musel přizpůsobit proud diodou, aby bylo docíleno stejné svítivosti. Impulzní proud by tak mohl dosahovat řádů ampér, a na to LED diody nejsou stavěny. Multiplexaci zavedeme tak, že propojíme všechny anody na patře kostky. Patro je tedy síť vodičů spojující všechny napájecí vývody diod, v jedné výšce. Každé patro je třeba zapínat jednotlivě. Katody jednotlivých diod jsou spojeny do sloupců kostky. Uzemněním sloupce tedy rozsvěcíme jednotlivé diody, máme-li sepnuté právě jedno patro. Je tedy nutné řídit každý sloupec jednotlivě a individuálně k právě sepnutému patru. Tímto postupem docílíme požadované hloubky obrazce. Kdyby byla sepnuta dvě patra současně, svítil by stejný obrazec na obou z nich. Jedná se o negativní jev, kterého je třeba se vyvarovat.

Na Obr. 1 je schematicky znázorněno zapojení jednotlivých diod v jedné řadě, pro variantu tříbarevné kostky.



Obr. 1 Schematické zapojení diod

Obnovovací frekvenci kostky musíme volit tak aby lidské oko nepostřehlo přepínání jednotlivých pater, tudíž minimálně 25 Hz. Avšak z praxe se osvědčili frekvence minimálně dvojnásobné. Čas sepnutí jednotlivého patra se dá vyjádřit vzorečkem.

$$t_{patra} = \frac{1}{f_{obnovovaci} * \text{počet pater}} \quad (1)$$

Základním faktorem, který ovlivňuje způsob řízení led kostky, je její velikost a použité diody. Diody mohou být jednobarevné nebo vícebarevné. Podle toho nám stoupají nároky na řízení. Totéž platí o velikosti. S každým rozměrem roste počet diod s třetí mocninou. K řízení led kostky je tedy možno přistupovat několika způsoby. Pro uzemnění sloupců se jako nejjednodušší možnost jeví použití tranzistorů, které mohou bít řízeny výstupem mikrokontroléru. V tomto případě by bylo potřeba nastavit proud diodou, proudovým zdrojem. Tento způsob rozhodně není vhodný pro kostku o rozměrech větších než je 5x5x5. Už pro tento rozměr by bylo zapotřebí 25 tranzistorů, a mikrokontrolér se stejným počtem výstupů. Řídicí obvod, by díky takto velkému počtu součástek, nabýval na velikosti a ceně. Toto řešení, tedy není příliš vhodné, ale přesto u malých, jednobarevných kostek někdy používané.

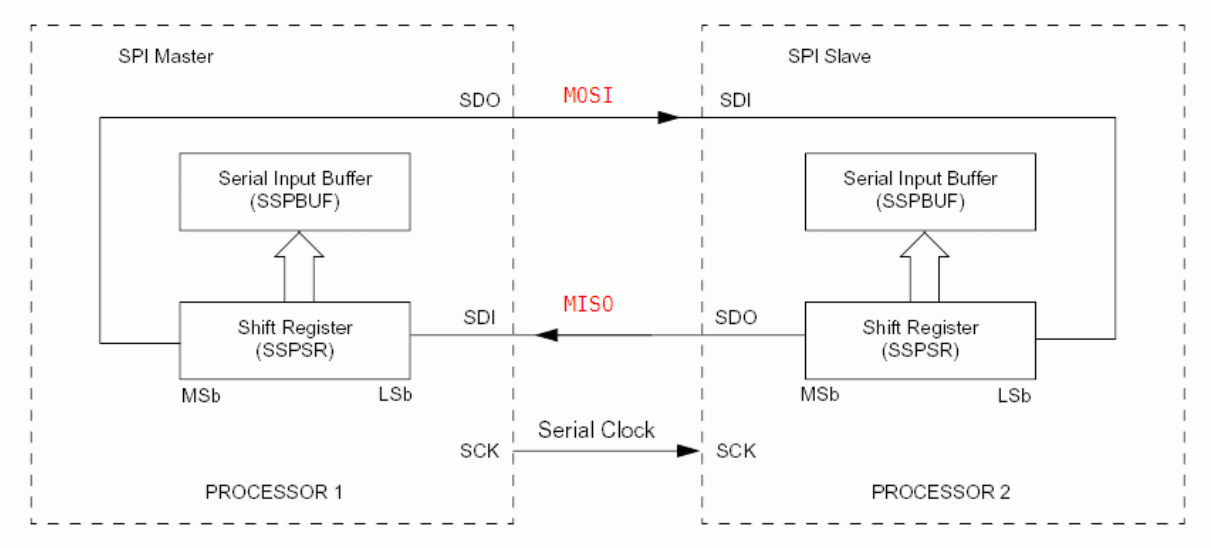
Další možností, jak řídit jednotlivé sloupce, by byla multiplexace jednotlivých řad. V podstatě bychom rozdělili všechny řady stejně jako patra a procházeli bychom v jednom sepnutém patře, řádek po řádku. Při použití tranzistorů, tak jako v předešlém případě, by se tím jejich počet značně snížil, taktéž i počet řídicích pinů mikrokontroléru. [11]

Jednou z nejpoužívanějších možností pro řízení sloupců je použití LED budičů. Jedná se o integrovaný obvod, jehož základem je posuvný registr se sériovým vstupem a výstupem. Tento posuvný registr je rozšířen o proudový zdroj, termální ochranu. Zpravidla se hodnota proudu nastavuje jedním externím odporem. Takovýchto budičů se na dnešním trhu vyskytuje velmi hojné množství, každý větší výrobce má vlastní sérii obvodů, avšak funkce zůstává většinou stejná nebo velice podobná. Jako příklad, bych uvedl STP16DP05 tento obvod je v podstatě stejný jako CAT4016 a spoustu dalších. [2] Výhodou těchto budičů je hlavně jejich pořizovací cena, integrita a potřeba pouze jedné externí součástky. Dalo by se o nich také říci, že pro účel řízení se používají jako převodníky *serial in-paralel out*. Výstupní piny se tedy propojí s jednotlivými sloupci. V takovémto zapojení, tedy odpovídá jeden bit v registru, jednomu sloupci v kostce. Toto řešení je vhodné nejen pro všechny velikosti kostky, ale i pro vícebarevné varianty. Jejich rozšířené verze již obsahují nastavitelné proudové zdroje. Každý výstup lze tedy řídit individuálně. To je jejich hlavní výhodou, protože jas diody lze regulovat bez složitějších algoritmů. Nejsou tak běžné, jelikož jejich pořizovací cena je značně vyšší, než u nižších verzí.

Stavba samotné kostky se dá rozdělit na dva principiálně odlišné postupy. První je stavba po patrech, spočívá ve vytvoření šablony s mřížkou, která odpovídá délce nožiček diody. Do mřížky se dále vyvrtají díry o průměru diod, které jsou použity. Diody dále osadíme do předem připravených dírek a anody diod vhodně ohýbáme tak, aby byly propojeny v celém patře. Takto vzniklá patra se dále skládají na sebe. Onen postup však nedovoluje volit rozteč mezi diodami a je nevhodný pro vícebarevné kostky. Druhou možností je stavba po řadách. Tento postup je podrobně rozepsán v kapitole 3.8.

2.2 SPI sběrnice

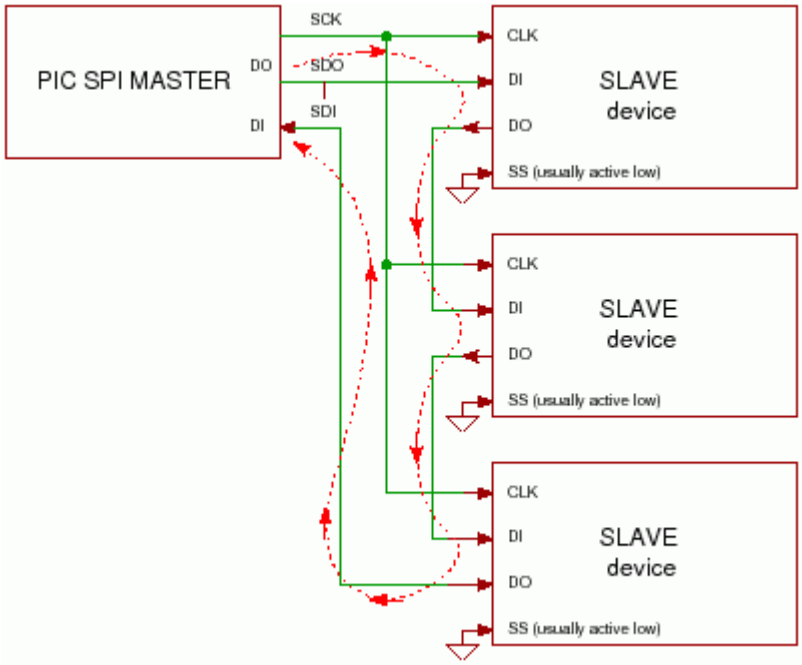
Externí sériová sběrnice, slouží pro vzájemné propojení dvou či více uzlů, z čehož jeden uzel pracuje jako řadič (Master) a ostatní uzly pracují v režimu Slave. Uzel typu Master, vysílá synchronizační signál (SCK), ten je rozveden do ostatních uzlů, a tím zaručuje synchronní přenos dat. Tento signál dosahuje frekvence až 70 MHz, běžnější je však okolo 10 MHz. Dále sběrnice obsahuje dva vodiče značené jako MOSI (Master Out, Slave In) a MISO (Master In, Slave Out). Tyto vodiče přenášejí obousměrně data, takové zapojení se nazývá full duplex. Dalším vodičem je SSEL (Slave Select), ten slouží k výběru uzlu, s kterým bude Master komunikovat.



Obr. 2 Princip komunikace dvou zařízení [7]

Na Obr. 2 je vidět způsob propojení dvou uzlů. Každý uzel obsahuje to nejjednodušší, dva registry, datový záchytný registr (SSPBUF) a posuvný registr (SSPSR). Do registru SSPSR je zapsán bajt, který byl přijat a ještě nebyl zpracován, tento registr slouží jako jednoprvková fronta, zároveň slouží i k vysílání dat. Každý posun tohoto registru znamená, že vysunutý bit je odeslán na výstup SDO a obráceně. Logická hodnota zapsána ze vstupu SDI, je na nejnižší pozici posuvného registru. Uzel v režimu Master, generuje SCK, a tím i časy, kdy se jednotlivé registry (SSPSR) posouvají. U většiny zařízení bývá možnost nástupné či sestupné hrany SCK.

Existují dva způsoby zapojení této sběrnice. První je naznačen na Obr. 3, kde se může představit i více zařízení typu Slave. Dále by bylo rozvedeno několik vodičů typu SSEL, každý do jednoho Slave uzlu. Logickou jedničkou, bychom vybrali konkrétní uzel, s kterým se má komunikovat. Druhý způsob je řetězení Slave uzlů tak, jak je naznačeno na Obr. 3 čárkovanou čarou. [7]



Obr. 3 Slave zařízení propojená do řetězu [7]

Hodinový signál, je do všech uzlů přiveden paralelně. Datové vodiče zde tvoří kruh. Uvážíme-li, že každé zařízení obsahuje posuvný registr, tak tímto zapojením vznikl jeden dlouhý posuvný registr. Toto zapojení je použito pro tento projekt.

Výhodou této sběrnice je snadná implementace, lze jí díky tomu najít v mnoha různých zařízeních, např.: v komunikaci s některými druhy pamětí (EEPROM), v LCD panelech, v hodinách reálného času (RTC). Podpora pro SPI je zabudována do řady mikro kontrolérů STM32, AVR, PIC16xx. Pro jednoduché řízeni okolních zařízení je možno použít jednosměrnou komunikaci, při které se vysílají nebo čtou data z jednoho uzlu. Výhodné je takto zapojit několik posuvných registrů.

Mezi nevýhody této sběrnice určitě patří existence pouze jednoho zařízení typu Master. Je také možné použít zapojení Multiple Master, ale to vyžaduje použití složitějšího komunikačního protokolu. Přenos sběrnice je omezen na kratší vzdálenost, kvůli synchronizaci hodinového signálu s přenášenými daty. [7]

2.3 USB sběrnice

USB (Universal seriál bus) je sériová sběrnice, která je v dnešní době nejpoužívanější u osobních PC ke komunikaci s externím zařízením. Vznikla za spolupráce několika velkých firem (např.: HP, Intel, Microsoft), jejichž cílem bylo sjednotit rozhraní pro komunikaci se zařízeními s nižším datovým tokem. Tato sběrnice, téměř nahradila dříve používané sériové rozhraní RS-232, paralelní port a PS/2.

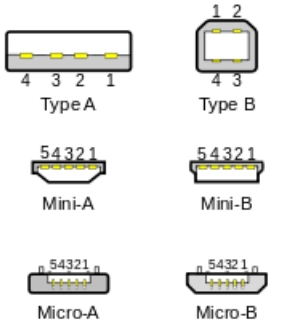
Nespornou výhodou této sběrnice, je funkce Plug and Play, což v překladu znamená, připoj a hraj. Tato funkce umožnuje připojení zařízení, aniž by bylo nutné restartovat počítač. Zařízení se po chvíli načte. V případě prvního připojení se nainstalují ovladače pro dané zařízení. Tato funkce, je již dlouho integrována do většiny operačních systémů.

Zařízení s USB, lze rozdělit do dvou skupin. První skupinu tvoří zařízení s vlastním napájením, nezávislým na napájení USB (velké externí disky, tiskárny). Druhou skupinu tvoří zařízení přímo napájené z USB (Flash disk, klávesnice, myš). Sběrnici tvoří čtyři vodiče. Dva napájecí VCC a GND, a dva datové vodiče D+, D-. Velikost napájecího napětí je 5 V. Proud povolený pro běžné zařízení je 100 mA, avšak na požádání zařízení, tento proud může být maximálně až 500 mA. Těchto výhod využívají některé druhy externích disků, ke kterým výrobci dodávají rozdvojený kabel se dvěma zástrčkami, pro vyšší proudový odběr.

USB rozhraní využívá několik typů konektorů. Nejpoužívanějším je plochý konektor Typ A a v poslední době i Micro-B. Těmito konektory se často setkáme u hudebních přehrávačů, telefonů a spousty dalších zařízení. Typy základních konektorů a jejich piny, jsou zobrazeny na Obr. 4.

Základní parametry:

- Rychlost komunikace: 1,5 Mbit/s až 5 Gbit/s
- Maximální délka sběrnice bez budičů: 5 m
- Maximální počet zařízení na jednom konektoru: 127
- Sběrnice obsahu 5 V napájení

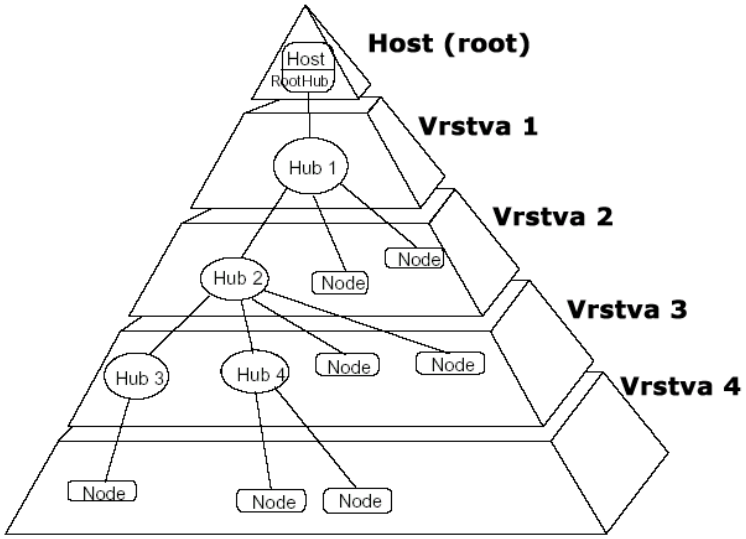


Pin	Signál
1	VCC
2	D-
3	D+
4	GND

Obr. 4 Základní typy USB konektorů

Přenos dat mezi počítačem a zařízením začíná tím, že počítač vyšle takzvaný token paket. Tento paket obsahuje typ a směr výměny dat, USB adresu a číslo koncové jednotky (endpoint). Dále zařízení odešle paket s daty, nebo se nachází ve stavu, kdy žádná data nejsou k dispozici. Po tomto přenosu zařízení, odešle handshake paket s informací o tom, zda přenos proběhl správně.

Zařízení vybavené rozhraním USB je buď rozdělovač, nebo funkční jednotka. Propojení je řešeno pomocí víceúrovňové hvězdicovité struktury. Viz Obr. 5.

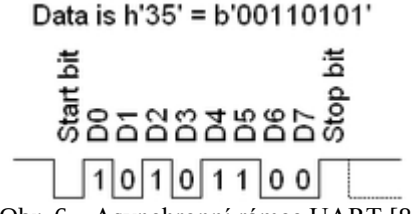


Obr. 5 Struktura USB [6]

2.4 USART

Zkratka USART (Universal Synchronous and Asynchronous Receiver Transmitter), označuje univerzální synchronní a asynchronní sériový přijímač a vysílač, jedná se o integrovaný hardware, který zprostředkovává komunikaci. Jde o nejpoužívanější sériové rozhraní mezi mikrokontroléry, které je schopné obousměrné komunikace s rychlostí až desítek kBaud. Lze jej nastavit na synchronní nebo asynchronní režim.

Asynchronní režim (full duplex) je nepoužívanější formou komunikace mezi mikrokontroléry. V tomto režimu není vysílán hodinový signál, ale může probíhat výměna dat, ve stejný čas, oběma směry. Vysílání dat se uskutečňuje na pinu označovaném jako TX (transmit) a příjem dat na pinu RX (receive). Klidová úroveň signálu je logické 1, neprobíhá výměna dat. Překlopením do logické 0, začne přenos dat, značený také jako start bit. Dále následuje rámec 5 - 9 datových bitů, seřazených od nejméně významného (LSB) až po nejvíce významný (MSB). Datový rámec je ukončen stop bitem, po kterém následuje klidový stav, nebo se znovu začnou vysílat data. Graficky znázorněný průběh je zobrazen na Obr. 6. Do datového rámce také spadají paritní bity, parita může být sudá, lichá, mezerová nebo značená. Většina zařízení nabízí volbu jednoho nebo dvou stop bitů. Tento režim je určen pro rozhraní RS232 a RS485. Pokud hovoříme o asynchronní části USART, je uváděna jen jako UART. Je nutno brát v úvahu, že každé rozhraní má odlišné napěťové úrovně. USART má logickou jedničku v mezích 0 - 0,8 V a logickou nulu 2 – 5 V. RS232 má rozdílné hodnoty, odvozené od použitého zařízení, hodnoty jsou ± 5 V, ± 10 V, ±12 V, ±15 V. Kladná hodnota napětí se vždy bere za logickou nulu a záporná hodnota napětí za logickou jedničku. Pro konverzi logických úrovní se osvědčil převodník MAX232. [9]



Obr. 6 Asynchronní rámec UART [8]

2.5 Vytvořené projekty

Na internetu se několik poslední let, objevuje čím dál více projektů, řešící toto téma. Projekty se často liší, jak konstrukčním řešením, tak i elektronickým řízením. Rozeberme si tedy pár projektů a pokusme se najít klady a zápory jednotlivých řešení.

2.5.1 5^3 LED Cube for PIC16F688

Tento projekt se zabývá kostkou o velikosti 5x5x5, diody jsou jednobarvé, 3 mm v průměru. Tyto parametry byly zvoleny čistě pro jednoduchost a cenu celkového výroku. Autor v textu, také nabádá k experimentování s velikostí LED kostek. Hlavním důvodem k těmto pokusům je poměr cena, velikost. Zamyslíme-li nad kostkou o rozměrech 5x5x5, tvoří ji 125 diod, kostka 8x8x8 je tvořena 512 diodami. Při rozměrech 10x10x10, jde o 1000 diod. Tvůrce je zastáncem názoru, že rozměr 5x5x5 dává nejlepší poměr cena/výkon, k celkové době potřebné na stavbu kostky.

Jak už je patrné z názvu, hlavním ovládacím prvkem je MC PIC16F688. Onen procesor pracuje s frekvencí 8 MHz, tento výkon je pro danou kostku zcela dostačující. Spínání pater zde zajišťuje bipolární tranzistor v zapojení SC.

Stavba vlastní kostky probíhala po jednotlivých patrech. Toho bylo dosáhnuto vytvořením šablony s otvory pro diody, umístěné v mřížovém vzoru. Vzdálenost jednotlivých diod, byla určena délkou nožičky diody. Propojení spočívá ve vhodném ohýbání nožiček, tak aby tvořili mříž. Toto řešení má nevýhodu v pevně dané rozteči mezi jednotlivými diodami.

DPS zde tvoří podstavu pro kostku. Tato forma upevnění má hned několik výhod. První z nich je propojení kostky přímo s řídicí deskou bez žádných propojek, tím se vyvarujeme chyb při stavbě. Otvory pro uchycení kostky jsou už od návrhu přesně dané, a pokud desku nevyrábíme sami, nemusíme se starat o jejich přesnost. [2]

Pro řízení diod je zde použit LED budič STP16DP05. Tento integrovaný odvod se u těchto zařízení používá velmi často a je mu věnována kapitola 3.3.

2.5.2 How not to engineer

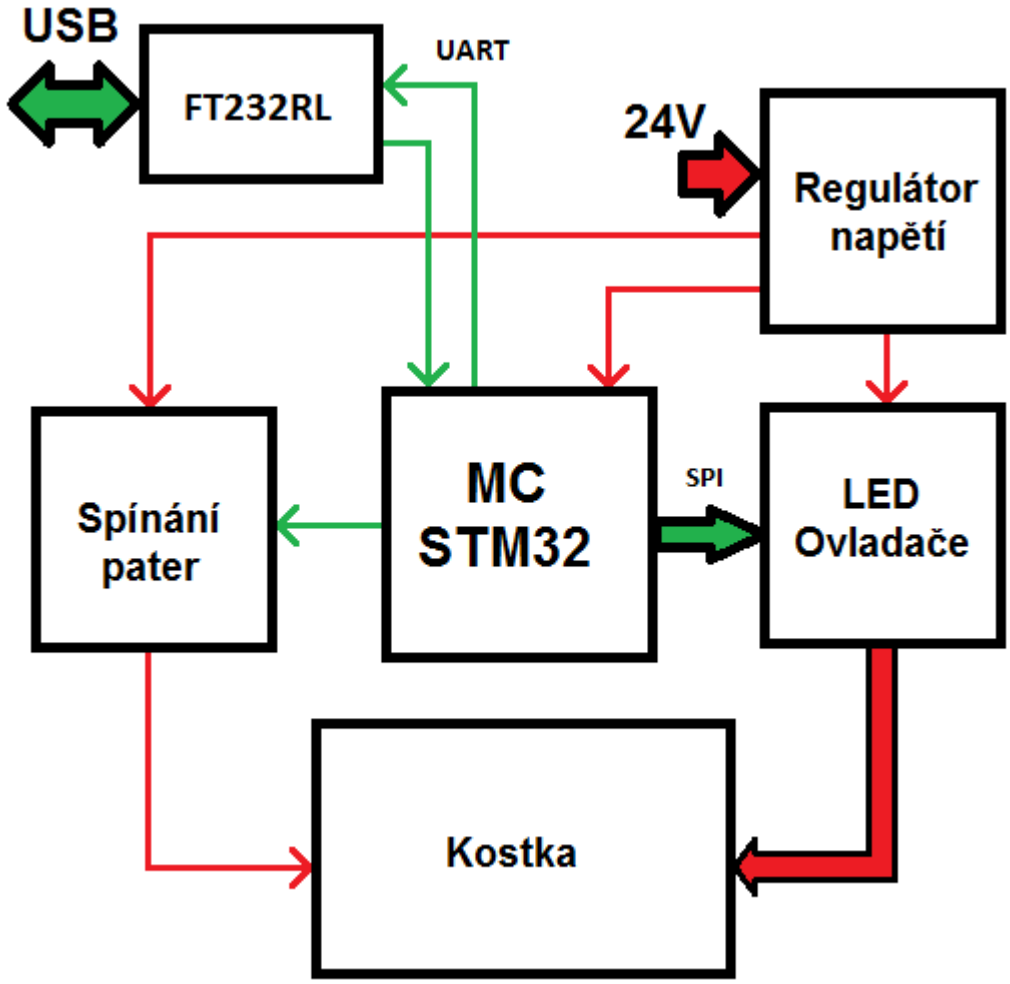
Tyto stránky popisují dva projekty, jejichž autorem je Nick Schulze. Prvním projektem je jednobarevná kostka o rozměrech 8x8x8. Diody jsou zde v difuzním pouzdře, s průměrem 5 mm. Jako řídicí člen, je zde uveden kit Arduino. Stavba samotné LED kostky byla provedena po patrech a není zde příliš rozebírána. Kostka je následně usazena v dřevěném podstavci. Propojení řídicí desky s kostkou, je provedeno pomocí konektorů a propojovacích kabelů. Nechybí zde ani schéma zapojení a zdrojové kódy, z kterých se dá čerpat. Kuriozitou toho projektu, je použití LED budičů jako sérioparalelního převodníku dat pro buzení tranzistoru pro ovládání pater. Tranzistory pro spínání pater jsou tedy typu PNP. Pro řízení LED, je zde použitý LED budič STP16CP05. Autor dále popisuje tvorbu programu a animací.

Druhým projektem je kostka o stejné velikosti, avšak již v provedení RGB. Diody jsou opět v difúzním pouzdru, tentokrát o průměru 10mm. Autor tedy potřeboval zvolit jiný rozměr, než je velikost nožičky diody. V projektu je podrobně probrána stavba kostky po řadách, s přiloženou fotodokumentací. Kostka samotná, je znovu usazena do dřevěného podstavce a s řídicí deskou propojena pomocí kabelů, avšak tentokrát bez konektorů, kabely jsou přiletovány přímo k desce. Ovládacím prvkem je zde chipKIT UNO, s 32 bitovým procesorem. Stejně jako v předešlém projektu, je podrobně probrána programová část, s vysvětlením multiplexoru pater, a také Bit Angle Modulation, této modulaci je věnována kapitola 4.1.2. [3]

3 Návrh kostky

3.1 Blokové schéma

Blokové schéma znázorňuje zapojení jednotlivých komponentů a jejich vzájemné propojení. Hlavním řídicím členem je vývojová deska STM32F4 Discovery (kapitola 3.2). Ta je připojena k řídicímu obvodu pro spínání pater (kapitola 3.5) a LED budičům (kapitola 3.3). Propojení s PC a komunikaci po USB, zprostředkovává integrovaný obvod FT232RL, ten dále komunikuje s hlavním MC pomocí UART sběrnice. Regulátor napětí má svojí vlastní desku plošných spojů.



Obr. 7 Blokové schéma

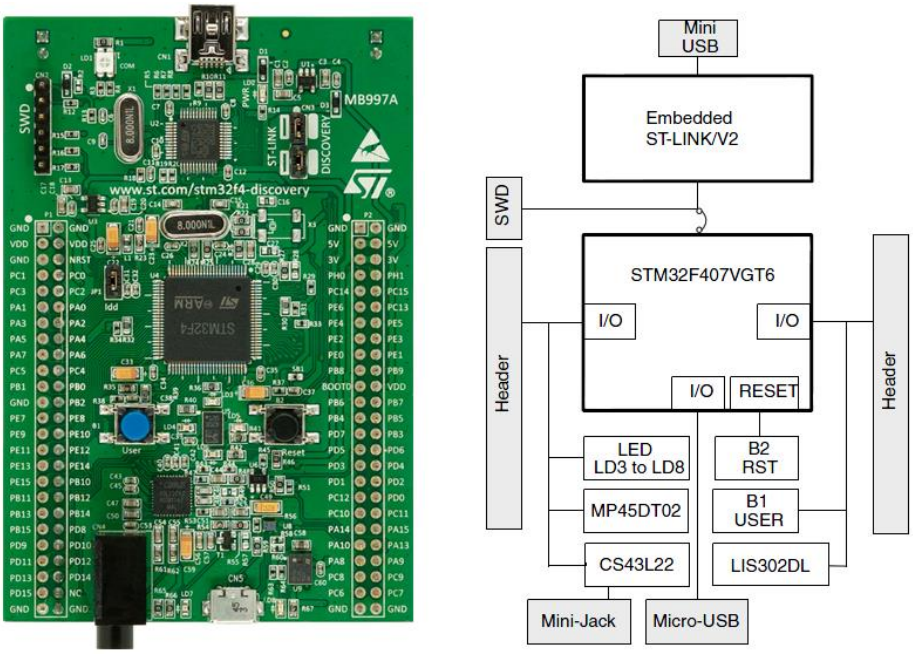
3.2 STM32F4 Discovery

Pro výběr řídicího procesoru je důležité zaměřit se na nároky, které na něj budou kladeny. Řídicí procesor bude muset obsahovat modul, jak pro SPI tak pro USART, s co možná největší komunikační rychlostí, spolu s tím, se budou zlepšovat vykreslovací možnosti kostky. Vzhledem k těmto požadavkům bude muset být frekvence jádra mnohonásobně vyšší, aby se mohli v čas vypočítat potřebné data pro plnění LED budičů. Za tímto účelem je třeba hledat procesor s kmitočtem v řádu stovek MHz.

Pro vlastní realizaci projektu, byl vybrán vývojový kit STM32F4 Discovery. Výhodou této volby jsou rozmanité periferní příslušenství a integrovaná Flash paměť. Desku lze připojit, pomocí USB, k počítači se systémem Windows XP a nebo vyšším. USB rozhraní dále zajišťuje napájení desky. Pro běžné užití, tedy stačí pouze PC s odpovídajícím softwarovým vybavením. Deska také poskytuje napětí 5 V a 3 V s maximálním proudovým odběrem 100mA pro externí aplikace. Připojení externího 5 V zdroje, je možné přes vyvedené piny. Připojení k PC se provádí kabelem USB A – miniB.

Podstatnou součástí desky je programovací a ladící rozhraní ST-LINK/V2, jehož činnost, řídí druhý mikrokontrolér STM32F103. Zajišťuje taktéž komunikaci s PC. Toto rozhraní slouží k programování a ladění mikrokontrolérů řady STM32 a STM8. Je možné s ním naprogramovat, jak hlavni MC, tak i jiné aplikace, pomocí zvláště vyvedeného konektoru SWD a dvou propojek (jumper), kterými se volí požadovaná funkce.

Základem této desky je mikrokontrolér STM32F407VGT6 s 32 bitový RISC jádrem ARM Cortex-M4F, 1 MB Flash a 192 KB RAM pamětí. Mikrokontrolér pracuje na kmitočtu 168 MHz [1]. Deska je vybavena rozmanitým příslušenstvím např. 3 osy akcelerometr, digitálním mikrofonem, programovacími obvody. Na desce je vyvedeno 5, 16 bitových registrů PA – PE, některé z registrů se používají pro periferní příslušenství [1]. Pro účely toho projektu, byla využita možnost komunikace po sériové sběrnici SPI, a volně vyvedené výstupy pro spínání pater, taktéž sběrnice USART. SPI sběrnice je vyvedena na pinech PA5 (SCK), PA7 (MOSI). Spínání pater probíhá na pinech PE8 – PE15. Pin PE0 je použit pro spínání výstupů registrů, vstup LH\DM1 (kapitola 3.3).



Obr. 8 Kit STM32F4 Discovery a jeho blokové schéma

3.3 Budiče LED

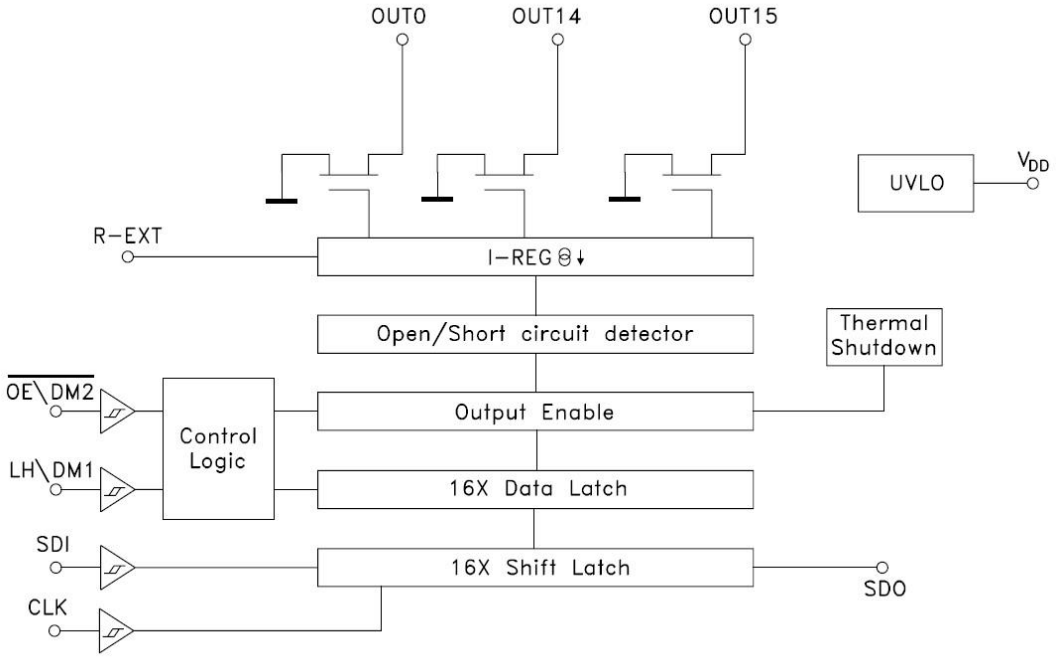
Pro spínání jednotlivých sloupců byl zvolen integrovaný obvod STP16DP05. Budič obsahuje 16 bitový posuvný registr, pro komunikaci po SPI sběrnici. Maximální kmitočet komunikace je 30 MHz. Vstupní proud se nastavuje externím odporem. Hodnota odporu byla zvolena podle katalogového listu [4] jako 1 kΩ, touto volbou dosáhneme proudu 20 mA pro jednu diodu. Detailní zapojení ovladače je uvedeno v příloze 4.

Na Obr. 9 je znázorněna pravdivostní tabulka obvodu.

CLOCK	LE\DM1	$\overline{\text{OE}}$\DM2	SERIAL-IN	OUT0 ..... OUT7 ..... OUT15	SDO
⌐¬	H	L	Dn	Dn ..... Dn - 7 ..... Dn -15	Dn - 15
⌐¬	L	L	Dn + 1	No change	Dn - 14
⌐¬	H	L	Dn + 2	Dn + 2 ..... Dn - 5 ..... Dn -13	Dn - 13
¬⌐	X	L	Dn + 3	Dn + 2 ..... Dn - 5 ..... Dn -13	Dn - 13
¬⌐	X	H	Dn + 3	OFF	Dn - 13

Obr. 9 Pravdivostní tabulka obvodu STP16DP05 [6]

Z tabulky si můžeme všimnout, že kdykoliv po přivedení nástupné hrany na vstup LH\DM1, dojde k překlopení posuvného registru do I-REG, tedy na vstupy budiče. Z tabulky je taky patrné, že k vypnutí výstupů stačí aktivovat vstup OE/DM2. Pro potřeby kostky postačuje zapojení vstupu LH\DM1, ten je připojen k pinu PE0. Vstup OE\DM2 není v této práci využit, avšak v rámci budoucího rozšíření řídicího programu, byl připojen k pinu PE1. Tímto zapojením je možné kontrolovat přijatá data. Tento budič byl zvolen vzhledem poměru cena, výkon a také hlavně kvůli jednoduché ovladatelnosti. Jeho cena se pohybuje kolem 60 Kč (květen 2015). Na Obr. 10 je uvedeno vnitřní blokové schéma zapojení budiče.



Obr. 10 STP16DP05 – vnitřní blokové schéma budiče [4]

3.4 Diody

3.4.1 Výčet podstatných parametrů

Při volbě diody je nutno se zaměřit na několik parametrů. Přiblížíme si jejich základní vlastnosti:

Pouzdra diod jsou vyráběna v rozmanitých tvarech a barvách, spolu s různými materiálovými provedeními. Nejvhodnějším typem pouzdra je klasické s drátovými vývodový. Avšak nejdůležitějším parametrem pouzdra je typ provedení čiré nebo difúzní. Čirá pouzdra nejsou vhodnou volbou pro led kostky, jelikož jejich vyzařovací úhel je značně omezen. Pro led kostku je velmi významné aby svit diody bylo možné pozorovat z co nejvíce úhlů. Z tohoto důvodu je vhodné volit typ pouzdra, který se nazývá difúzní. Jde o technologii výroby, kde dojde k zmatnění (difundování) plastu. Zbarvení těchto pouzder často odpovídá barevnému svitu diody. Diody RGB jsou vyráběny v bezbarvém provedení, ukázka na Obr. 14. Tento typ pouzdra rozvede světlo do celého objemu těla diody. Tím docílíme lepší pozorovatelnosti z velkých úhlů. Difúzním pouzdrem také předejdeme nepříjemným jevům, které mohou nastat jako např. *ghost* efekt. To znamená, že se dioda může zdát rozsvícena, ale při tom je pouze nasvícena diodou ze spodu. Dalším nežádoucím faktorem čirého pouzdra je špatná viditelnost při použití za denního světla.

Svítivost diody vyjadřuje hustotu světelného toku. Základní jednotkou je kandela[cd], u LED diod se nejčastěji uvádí v řádu mcd. Tento parametr je rozhodující pro viditelnost za denního světla. Jako osvědčená hodnota se bere minimálně 700 mcd při buzením proudem 20 mA. Tato hodnota svitu je za normálních světelných podmínek v místnosti zcela dostačující, ale pro použití venku nebo na přímém slunci už ne. Nedodržením tohoto parametru riskujeme nevýraznost nebo dokonce nerozeznatelnost svitu diody.

Vyzařovací úhel je volen pokud možno co největší, jas diody je tak více patrný ze všech stran. V praxi se nedoporučuje úhel menší než 20°. Pokud by byl pozorovací úhel menší, vzniká riziko *ghost* efektu.

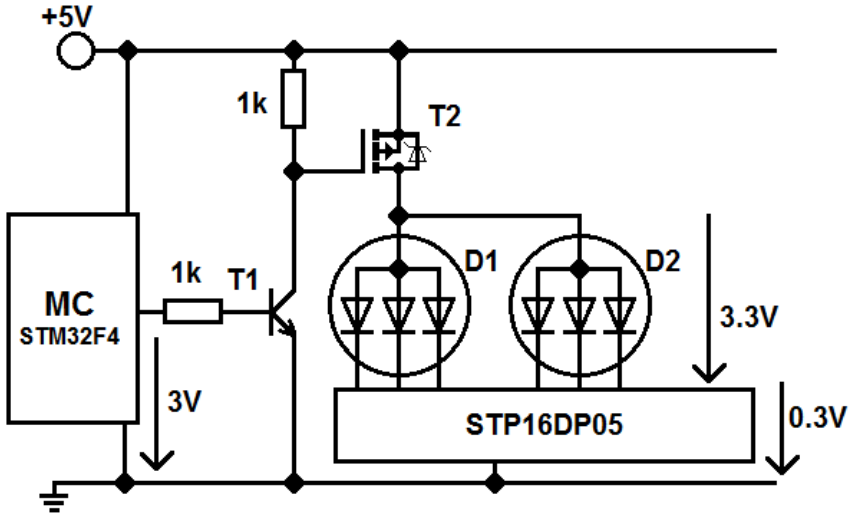
3.4.2 Použité diody

Na základě předešlé kapitoly můžeme shrnout, že pro realizace LED kostky jsou nejvhodnější volbou diody difuzní s drátovými vývody, s co největším vyzařovacím uhlem a minimální svítivostí 700 mcd. Dalšími zvolenými parametry bylo provedení RGB a velikost pouzdra 5 mm. Tato velikost pouzdra dává nejvhodnější poměr cena/výkon. Při pohledu na český trh zjistíme, že cena takovýchto diod se pohybuje kolem 20 Kč/ks (květen 2015), cena 512 diod tedy celkem vychází přes deset tisíc korun, což je příliš neúměrná částka. Podíváme-li se po světových aukčních portálech, můžeme tyto diody nalézt přímo od výrobců, přičemž se cena pohybuje okolo 1,5 Kč/ks (květen 2015). Pro účely této práce tedy byla zvolena druhá možnost, i když její velkou nevýhodu je 30 denní dodací lhůta. Vyzařovací úhel použitých diod je udáván jako 20°, a svítivost diody je pro každou barvu jiná pro červenou 3000 mcd, zelenou 6000 mcd, modrou 4000 mcd.

3.5 Spínání pater

Obvod pro spínání pater se skládá ze dvou tranzistorů a dvou odporů. Toto zapojení bylo nutné vytvořit kvůli nízkému napětí na výstupu vývojové desky, v sepnutém stavu. Ze zjednodušeného schématu (Obr. 11) je patrné, že potenciální napětí pro rozsvícení diod, je větší, než napětí na výstupu MC.

Spínací tranzistor T2 potřebuje pro správnou funkci tranzistor T1, který zajistí jeho buzení. Budící tranzistor T1 není výkonové zatěžován a jeho ztrátový výkon je zanedbatelný. Tranzistor T2 musí být volen s ohledem na maximální zatěžovací proud. Tranzistory budou pracovat v nízkofrekvenční oblasti kmitočtu tudíž kmitočtové parametry tranzistorů T1 a T2 nejsou podstatné. Tranzistor T2 byl zvolen jako unipolární v rámci minimalizace proudového zatížení řídicí obvodů. Jeho druhým podstatným parametrem je odpor při zapnutí, ten činní pouhých 0,06 Ω, z toho vyplývá že tranzistor bude téměř bezztrátový.



Obr. 11 Zjednodušené schéma spínání patra

3.6 Napájecí modul

3.6.1 Žádané vlastnosti

FEKTBOT je napájen dvěma autobateriemi v sérii, tudíž 24 V. Proto je nutné regulovat napájecí napětí pro LED kostku. Je žádoucí si předem stanovit, jaké proudové zatížení bude muset zdroj snést, a také výstupní napětí, které bude potřeba k napájení LED diod. Proud diodou byl zvolen 20 mA, optimální proudové zatížení diody. Při maximálním zatížení, které nastane při rozsvícení všech diod na patře, tzn. 192 diod, vyjde proud 3,84 A. Napětí, kterým budou diody buzeny, je třeba volit s ohledem na účinnost. Nejnižším napětím, kterým by diody mohly být buzeny, je 3,5 V. Při tomto napětí, se otevírá modrá dioda. Nicméně je třeba brát v úvahu i požadavky ostatních řídicích členů, jako je mikrokontrolér. Ten je třeba napájet 5 V. Vzhledem k tomu, že při volbě napětí nižšího než je 5 V, by bylo zapotřebí dvou zdrojů napájení, a že při volbě napětí 5V, nebude docházet k zbytečné ztrátě výkonu, je logické že výstupní napětí by mělo být 5 V.

Pro napájení kostky je tedy zapotřebí zdroje, který bude mít na výstupu 5 V a proudovou zatížitelnost 4 A. Při volbě způsobu regulace napětí, byla zvolená spínací topologie zdroje z důvodu požadavku malých ztrát (velké účinnosti) a výdrže baterie FEKTBOTu. Pro napájecí modul byla vytvořena samostatná deska plošných spojů. Výhoda toho řešení spočívá v tom, že pokud by došlo k poškození regulátoru nebo desky s budiči, může obvody kontrolovat separovaně a tím snadněji odhalit chybu.

3.6.2 Realizace modulu

Pro vlastní realizaci regulátoru napětí byl vybrán obvod TPS54560, mezi jeho přednosti patří účinnost, která je v oblasti spojitých proudů 80% a více. Jedná se o typ step-down regulátoru (buck), s integrovaným horním spínačem N-MOSFET. Integrovaný obvod má v sobě oscilátor, který lze nastavit jedním odporem, připojeným k pinu RT/CLK, od hodnoty 100 kHz až po 2.5 MHz. Takto široce nastavitelná frekvence, umožnuje nastavit účinnost nebo určitou toleranci při výběru výstupního filtru. Implementována je taktéž kontrola výstupního proudu, která snižuje výstupní kapacitu a zjednodušuje externí kompenzaci frekvence. Pro možnost řízení externím hodinovým signálem, je integrovaný obvod vybaven interní smyčkou fázového závěsu, připojenou na RT/CLK, která synchronizuje frekvenci externího signálu na sestupné hraně. Interní N-MOSFET má odpor pouhých 92 mΩ, tento parametr podporuje vysokou účinnost a dovoluje dodávat spojitý proud do zátěže. Do maxima 5 A. [12]

Většina externích součástek byla zvolena podle doporučených hodnot v katalogovém listu. Avšak pro vlastní realizaci bylo zapotřebí některé součástky nahradit ekvivalentními náhradou, kvůli dostupnosti součástek na trhu. Pro integrovaný obvod a jeho keramické kondenzátory je doporučeno dielektrikum X7R a nebo vyšší. Kondenzátory s tímto dielektrikem mají nízké ESR (ekvivalentní sériový odpor) a tím se zlepšují vlastnosti obvodu. Nízké ESR je taktéž nezbytným parametrem výstupního filtru regulátoru.

Důležitým parametrem, který bylo třeba ověřit je indukčnost cívky a také proudové zvlnění na výstupu.

Výpočet indukčnosti

$$L_0 = \frac{V_{IN(max)} - V_{OUT}}{I_{OUT} * K_{IND}} * \frac{V_{OUT}}{V_{IN(max)} * f_{SW}} = \frac{30-5}{5*0{,}3} * \frac{5}{30*4*10^5} = 6{,}9\ \mu H \quad (2)$$

Zvlnění proudu

$$I_{RIPPLE} = \frac{V_{OUT} * (V_{IN(max)} - V_{OUT})}{V_{IN(max)} * L_0 * f_{SW}} = \frac{5*(30-5)}{30*6{,}9*10^{-6}*4*10^5} = 1{,}5\ A \quad (3)$$

Z čehož si lze odvodit špičkový proud cívkou

$$I_{L(peak)} = I_{OUT} + \frac{I_{RIPPLE}}{2} = 5 + \frac{1{,}5}{2} = 5{,}75\ A \quad (4)$$

Podle těchto parametrů byla vybrána potřebná cívka, která odpovídá nejbližší katalogové hodnotě, její saturační proud činní 6 A, jmenovitý proud 8 A a indukčnost 7,2 μH.

Vlastní deska plošný spojů byla vytvořena v dvouvrstvém provedení na základě doporučení v katalogovém listu. Tento krok byl nezbytný kvůli odvodu tepla z integrovaného obvodu. Na spodní straně integrovaného obvodu je umístěn power-PAD který vyžaduje termální prokovy na DPS kvůli odvodu tepla z integrovaného obdu. Masku DPS spolu s osazovacím plánem můžeme nalézt v příloze 3.

Celkový obvod při testování v laboratoři dosahoval 80% účinnosti v oblasti spojitých proudů, při skokových změnách odebíraného proudu se odchylka výstupního napětí měnila v řádech desítek mV. Do oblasti spojitých proudů se obvod dostal při výstupním proudu 0,6 A.

3.7 Obvod FT232RL

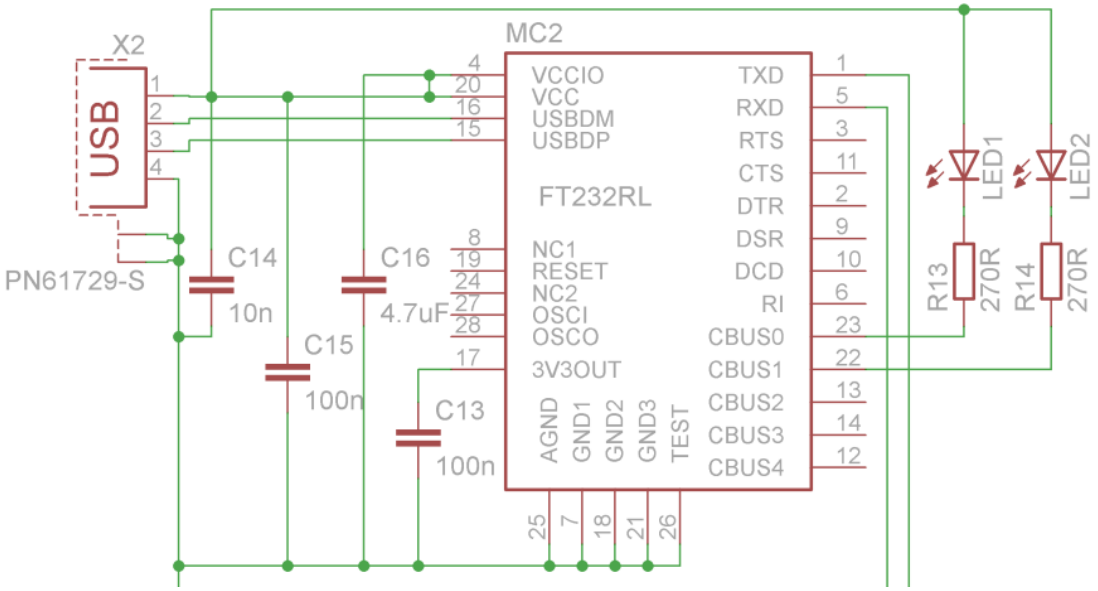
Pro převod dat z PC do kostky, byl zvolen obvod FT232RL od firmy FTDI (Future Technology Devices International Ltd.). Tato firma je zaměřena na převodníky mezi klasickými počítačovými periferiemi (např. RS485, RS232, UART) a rozhraním USB. Jedná se tedy o jednočipový převodník, který nevyžaduje mnoho externích součástek. Obvod je sám o sobě schopný pracovat jako USB zařízení. Po propojení s PC a nainstalovaní ovladače VCP (Virtual COM Port), se v počítači vytvoří virtuální sériový port, jenž lze ovládat jako standartní port COM. Druhou variantou komunikace jsou přímé ovladače D2XX. Tyto ovladače umožnují aplikačnímu programu komunikovat s obvodem FT232RL pomocí kódu, který je uložen v dynamické knihovně DLL. Tyto knihovny jsou využívány v mnoha vývojových prostředích. [7]

Základní parametry

- implementovaný celý USB protokol
- integrovaný oscilátor 12 MHz, externí není třeba
- UART podpora 7-8 bitového přenosu, 1-2 stop bity, parita sudá, lichá, značená, mezerová a bez parity
- FTDIChip-ID, jedinečné identifikační číslo
- široko nastavitelná komunikační rychlost
- velikost přijímacího bufferu 128 B, velikost vysílacího bufferu 256 B
- interní 1024 bit EEPROM paměť pro popis zařízení a konfigurace CBUS I/O

Napájení obvodu FT232RL je připojeno na USB sběrnici, to šetří baterii robota FEKTBOT. Připojení obvodu na sběrnici USB je realizováno pomocí konektoru typ B, ten je znázorněn na Obr. 4.

Pro signalizaci příjmu a odesílání dat jsou použity LED diody LED1 a LED2. Ty jsou připojeny na piny CBUS0 a CBUS1, tyto piny jsou už v základě nastaveny pro tuto funkci. Obvod má na napájecím vedení připojen několik blokovacích kondenzátoru, tantalový o kapacitě 4,7 µF a keramické o hodnotě 10 nF a 100 nF. Převodník dále obsahuje 3,3 V LDO regulátor, ten je třeba připojit externím 100 nF kondenzátorem na zem. Toto zapojení vychází z doporučeného zapojení v katalogovém listu. [7]

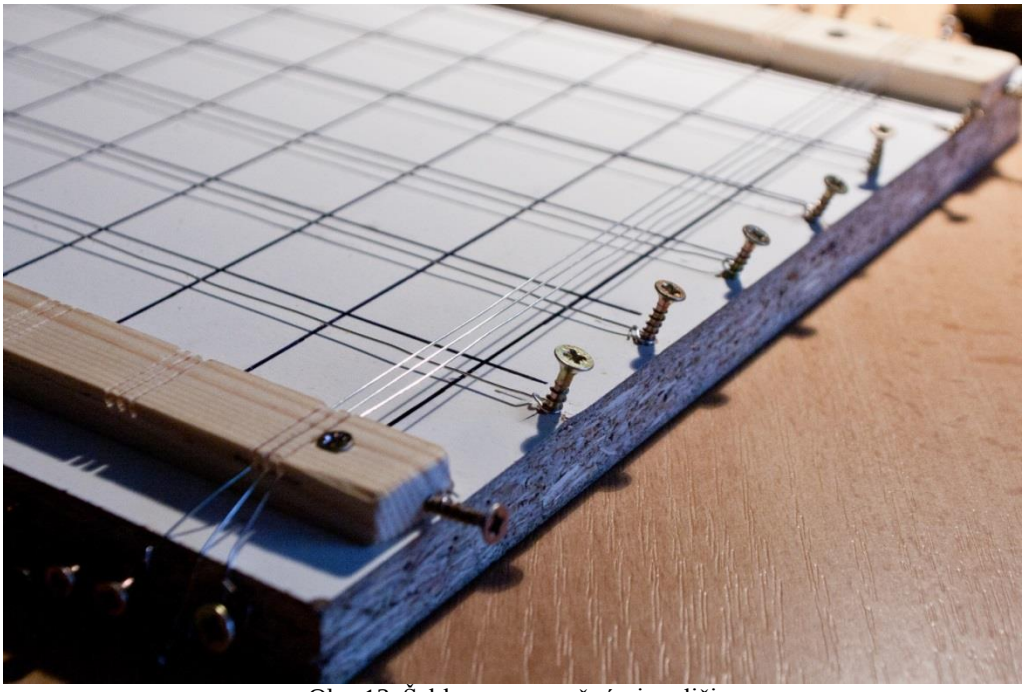


Obr. 12 Zapojení obvodu FT232RL

3.8 Konstrukce kostky

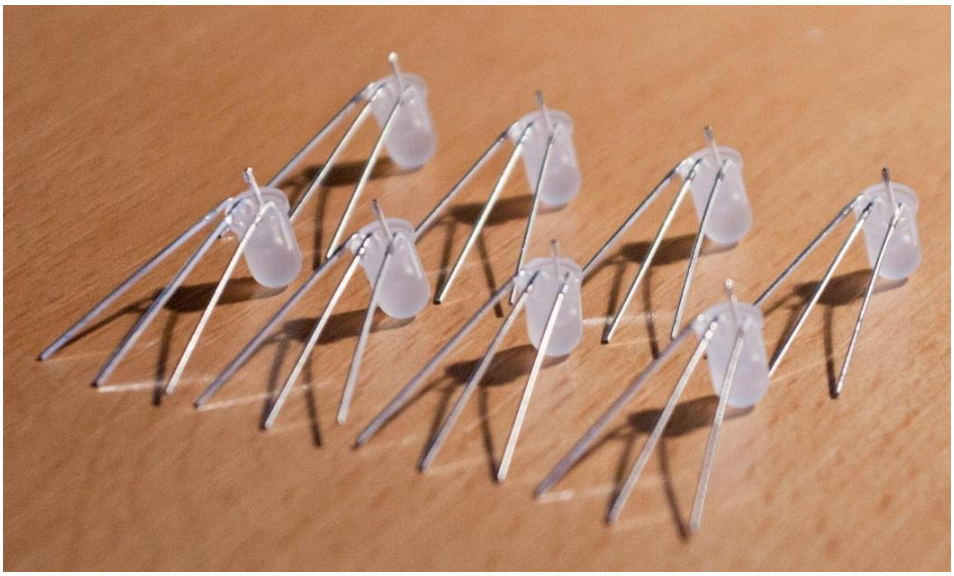
3.8.1 Konstrukce řady

Spočívá v sestrojení čtvercového vzoru z jednotlivých vodičů. Tento vzor je vhodné umístit na pevný podklad, ke kterému budeme moci připevňovat jednotlivé vodiče. Je důležité, aby šablona, podle které budeme tvořit, byla vyrobena s přesností na milimetry. Rozteč mezi středy diod, byla zvolena jako 35 mm, vzdálenost mezi sloupci katod byla zvolena jako 3 mm. Tato hodnota byla zvolena podle tloušťky diody. Celková šíře jednoho sloupce je 6 mm, průměr diody je 5 mm. Volbou takto malé mezery vzniká optický efekt sloupu, což přidává na pravidelnosti kostky. Na Obr. 13 je zobrazené řešení tohoto projektu.



Obr. 13 Šablona s upevněnými vodiči

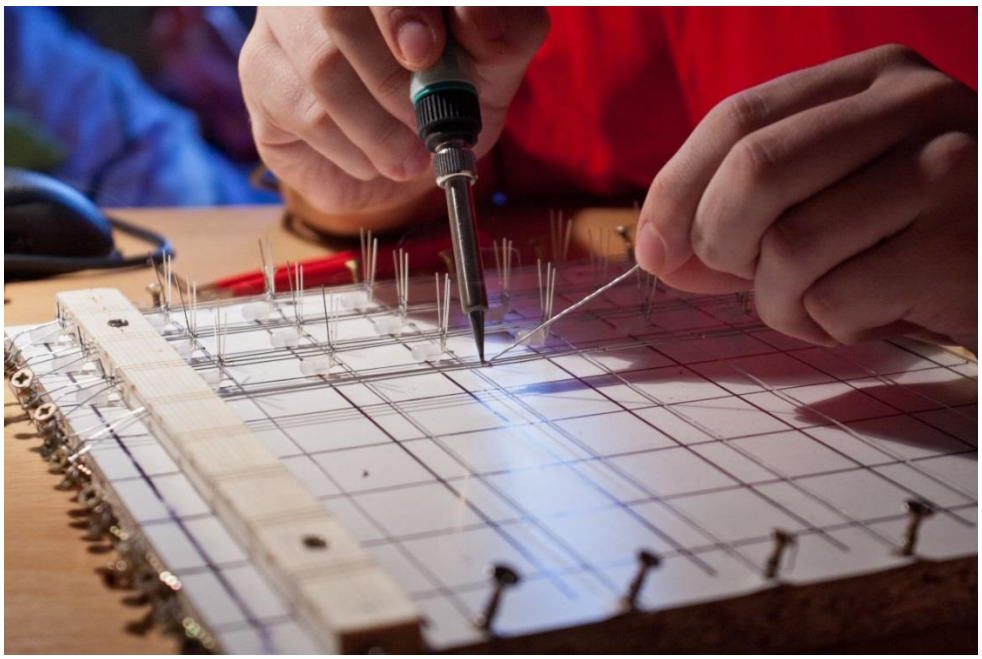
K takto vytvořené šabloně, je nutné přizpůsobit vývody diod. Na Obr. 14 vidíme řešení.



Obr. 14 Upravené diody

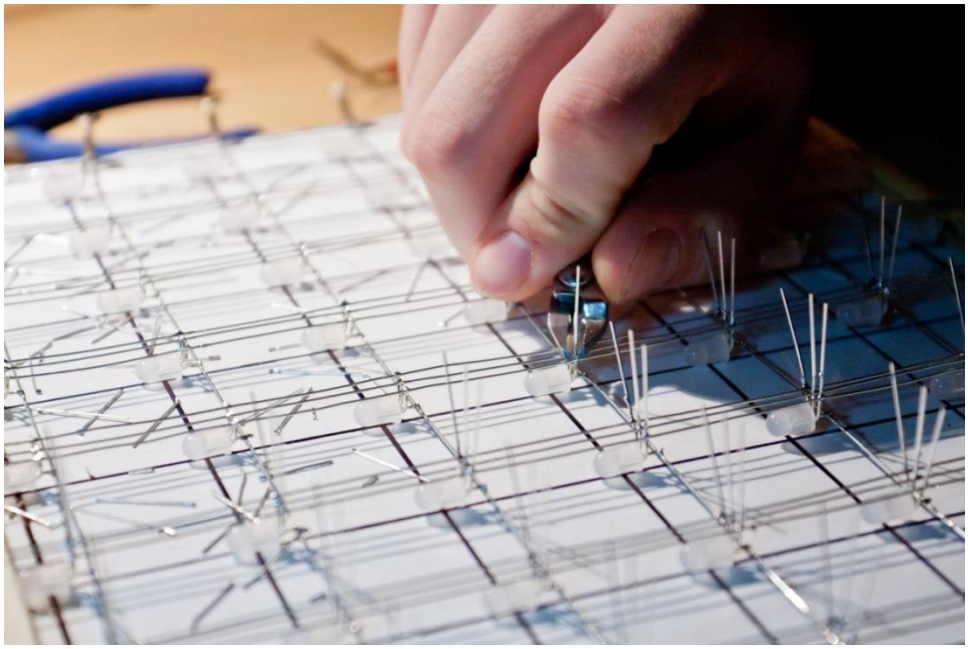
K tvorbě takto upravených diod, je vhodné si vytvořit šablonou. Osvědčil se kousek latě, s otvorem o poloměru 5 mm. Ohýbání je jednodušší a vytyčením polohy nožiček, je docílena vždy stejná rozteč.

Po této přípravě je na řadě samotné letování a ukládání jednotlivých diod na pozice. Vhodné je před samotným uchycením nanést na jednotlivé vodiče pájku, pro jednodušší upevnění diod. Jak je naznačeno na Obr. 15.



Obr. 15 Pájení patra

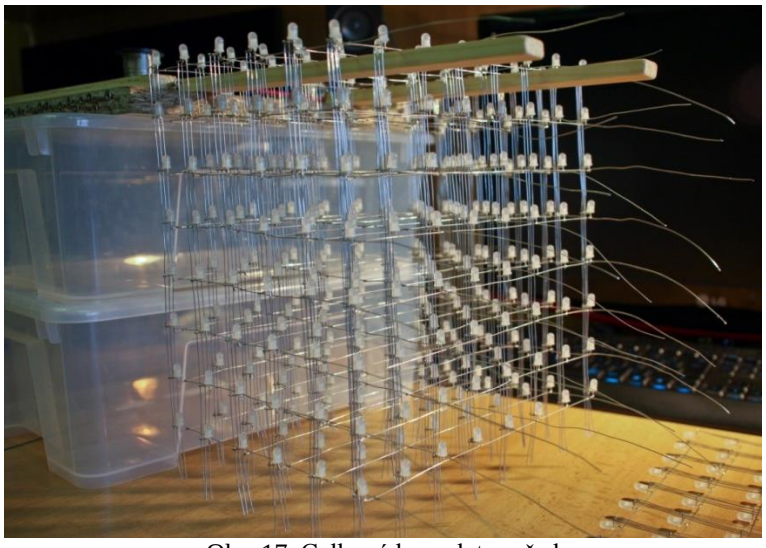
Pro stejné uchycení katod diod, byla připravena šablona z tvrdého papíru s přesnými zářezy, o požadovaném rozestupu 3 mm. Tato šablona lépe poslouží kontrole. Po dokončení letování je potřeba odstranit přebytečné nožičky diod a zbylé nepotřebné vodiče. Dolní vývodové vodiče, které budou upevněny v podstavě, je třeba pro snadnější upevňování zkrátit na stejnou délku.



Obr. 16 Dokončení řady

3.8.2 Celková kompletace

Po dokončení takto vzniklých řad, přistoupíme k propojení jednotlivých pater. To provádíme nejlépe v zavěšeném stavu, kdy s patry lze lehce manipulovat. Každé patro je propojeno třemi vodiči, dva na krajích a jeden uprostřed. Toto rozdělení je nutné, pro celkovou pevnost kostky. Vyšším počtem vodičů bychom docílili silnější pevnosti. Nicméně v porovnání s časovou náročností dané práce a ziskem pevnosti, není třeba vyššího počtu vodičů. Je vhodné vytvořit znovu papírovou šablonu k udržení předem daných vzdáleností. Po kompletaci přidáme napájecí vodiče pro jednotlivá patra, ty umisťujeme do středu kostky, poblíž sloupce, pro lepší optický efekt.

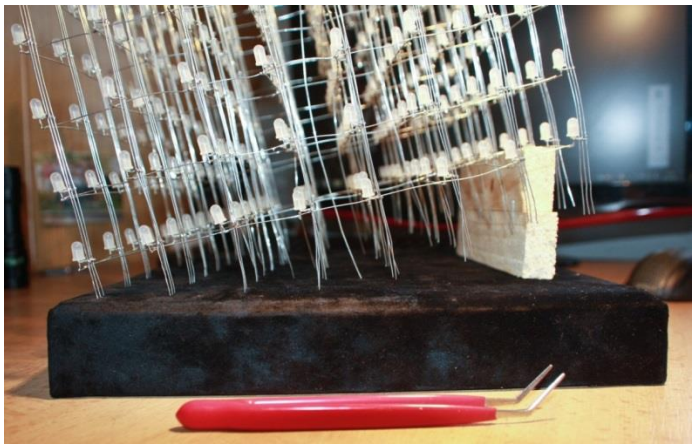


Obr. 17 Celková kompletace řad

3.8.3 Montáž do podstavce

Podstavec z drátů tvoří opěrnou část pro kostku. Podstavu tvoří rám z latí, které jsou sešroubovány. Na rám je upevněná deska s otvory pro uchycení kostky. Jelikož podstavec tvoří podstatnou viditelnou část, je důležité, aby esteticky působil jednoduše a celistvě. Tento problém byl vyřešen překrytím látkou (semišem).

Dalším krokem je upevnění kostky do podstavce. Tento proces provádíme nejlépe v nakloněném stavu, kdy se podstavce dotýká jenom jedna řada. Jak je ukázáno na Obr. 18.



Obr. 18 Upevňování kostky v podstavci

Kostku v podstavci, uchytíme přilepením tavící pistolí a ohnutím vyvedených vodičů. Na tyto vodiče, již můžeme přiletovat propojující vedení, k řídicí desce. Více detailních obrázku je připojených v příloze 1. V rámci této práce byla také vytvořena krychle z plexiskla, která slouží jako kryt před mechanickým poškozením a znečištěním.

3.9 Deska s budiči

Při návrhu desky s budiči, je třeba zaměřit se na několik podstatných parametrů, které není možné opomenout. V první řadě se jedná o šířku vodiče, který bude přivádět proud na jednotlivá patra. Tento proud byl stanoven jako 3,84 A. Dalším faktorem, který bychom se měli pokusit eliminovat, je odraz na vedení, sběrnice SPI. Tato sběrnice může komunikovat na frekvenci až 30 MHz, tudíž je nutné snažit se vytvořit tyto vodiče tak, aby byly co nejkratší.

Tyto dva důvody nám napovídají, že pro tento účel, je třeba vytvořit dvouvrstvou desku. V rámci co největší minimalizace desky, byly všechny nevýkonové součástky, zvoleny v pouzdře SMD. Při použití dvouvrstvé desky, nám odpadá starost se šířkou přívodního vodiče k jednotlivým tranzistorům. Můžeme jednoduše použít rozlévané země a na druhé straně rozlévaného napájení. Tato volba ulehčí celkovému rozvodu vodičů.

Návrh této desky byl vytvořen programem Eagle 6.4, tento program je komplexní návrhové prostředí, sloužící pro návrh DPS a schémat zapojení. Jsou zde implementovány rozsáhlé knihovny součástek s pouzdry. Nicméně pro tuto práci bylo zapotřebí vytvořit několik schématických značek a pouzder, například pro vývojový kit STM32F4, LED budiče, FT232 atd. Tuto knihovnu součástek lze nalézt v elektronické příloze.

Vrchní část DPS je osazena polovinou LED budičů, spolu s lištami pro připojení STM32F4 Discovery kitu, FT232RL a pěti budícími tranzistory. Spodní část DPS obsahuje druhou polovinu LED budičů, výkonové tranzistory, spolu se zbývajícími budícími tranzistory a všemi vývodovými konektory připojenými k samotné LED kostce. Masku a osazovací plán návrhu, lze nalézt v příloze 3. Celkové schéma zapojení je pak přiloženo v příloze 4.

4 Ovládací software

4.1 Řídicí program mikrokontroléru

Podstatnou část práce, tvoří řídicí program mikrokontroléru. Program musí obsahovat jak příjem a obsluhu dat, tak jejich konverzi na nové datové posloupnosti, které se dále budou posílat v přesném pořadí do registrů (LED budičů). Je nezbytné, aby tyto děje probíhali v co možná nejkratším časovém intervalu, aby nedocházelo k nežádoucímu jevu - snížení zobrazovací frekvence, a tím k problikávání jednotlivých diod. Tato kapitola se tedy zabývá implementováním dříve zmíněných částí, příjmem dat po sériovém rozhraní, prezencí přijmutích dat a jejich následné úpravě. Nakonec i plněním posuvných registrů. Inicializace jednotlivých klíčových částí bude taktéž rozebrána. Celý firmware byl vypracován v prostředí, Code Bloks.

4.1.1 Inicializace klíčových částí

Jedna z nejpodstatnějších částí, je správné nastavení přerušení (interrupt). Toto přerušení přesně stanovuje intervaly mezi jednotlivými přepínání pater a odesíláním dat. Jak je vidět na následujícím příkladu kódu, do funkce *SysTick_Config*, vkládáme za kolik hodinových taktů, chceme vyvolat přerušení. Proměnná *SystemCoreClock* je definována jako 168 000 000, to je frekvence jádra na 168 MHz. Vydělíme-li tuto proměnou osmi tisíci, získáme tak osm tisíc přerušení za vteřinu, to odpovídá přerušení každých 125 µs. Volba tohoto časového intervalu bude odůvodněna v kapitole 4.1.2.

```
15  void setSysTick(void){//preruseni kazdych 125us
16      if (SysTick_Config(SystemCoreClock/8000))
17      {
18          // zachyceni chyby
19          while (1);
20      }
```

Další nezbytnou inicializací je nastavení výstupních pinů, přiklad je uveden na inicializaci portu PE, tyto piny slouží k postupnému spínání pater a vyklápění dat z posuvných registrů na výstup (PE0).

```
47  void init_GPIOE(void){  //inicializace pinu na prepinani pater
48      // Zde probiha inicializacepinu PE8 az PE15,PE0,PE1
49      GPIO_InitTypeDef GPIO_InitStr;              // konfiguracni struktura
50      RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOE, ENABLE);// Zapnuti CLK na portech PE
51      GPIO_InitStr.GPIO_Pin = GPIO_Pin_15|GPIO_Pin_14|GPIO_Pin_13|GPIO_Pin_12|
GPIO_Pin_11|GPIO_Pin_10|GPIO_Pin_9|GPIO_Pin_8|GPIO_Pin_1|GPIO_Pin_0;// vyber pinu 0,1,8-15
52      GPIO_InitStr.GPIO_Mode = GPIO_Mode_OUT;     // pin nastavit jako vystupni
53      GPIO_InitStr.GPIO_Speed = GPIO_Speed_50MHz;// nastaveni frekvence CLK
54      GPIO_InitStr.GPIO_OType = GPIO_OType_PP;    // nastavit jako push / pull
55      GPIO_InitStr.GPIO_PuPd = GPIO_PuPd_NOPULL; // nastavit jako  odpor neaktivni
56      GPIO_Init(GPIOE, &GPIO_InitStr);//konfiguracni strukturu nahraj do nastavujcich
registru
57      first_pin_set();
```

Na začátku inicializace je třeba vytvořit inicializační strukturu, odpovídajícího typu pro nastavení pinu (*GPIO_InitTypeDef*), do kterého se budou ukládat námi požadované vlastnosti. V dalším kroku na řádku 50, je provedeno zapnutí hodinového signálu vnitřní komunikační sběrnice AHB1, která zajišťuje obsluhu výstupních portů a dalších bloků. Do struktury se zadávají konkrétní piny na portu, které chceme použít, proměnná *GPIO_Pin*. Dále pracovní mód pinu, vstupní, výstupní, s alternativní funkcí (SPI, USART) nebo jej lze nastavit jako analogový vstup, rychlost frekvence CLK, typ výstupu push/pull nebo open drain. V poslední řade výběr výstupního rezistoru, který může být pull up, pull down nebo nepřipojen ani jeden. Je-li toto všechno nastaveno, vložíme strukturu do funkce GPIO_Init, která podle parametrů struktury, nastaví potřebné registry. Pro tento port byla vytvořena funkce *first_pin_set*, která zaručí nastavení všech výstupů na log. 0, aby nedošlo k náhodnému zapnutí pater dříve, než bude potřeba. Toto nastavení je nutno provést pro všechny použité piny, takže i u pinů sběrnice SPI a USART, jelikož se jedná o obdobné nastavení, nebude dále rozebíráno. Změna je pouze v portech.

Nastavení SPI se provádí velmi obdobně. Znovu se vytvoří inicializační struktura pro GPIO, tentokrát pro port A, to je samozřejmě odvozeno od použitého SPI, a inicializuje se jako v předchozím případě. Tentokrát však potřebujeme propojit dané piny, s alternativní funkcí SPI. To se provede funkcí GPIO_PinAFConfig, jak je znázorněno v příkladu níže. Zapojení všech alternativních funkcí a další nezbytných věcí lze nalézt v katalogovém listu – *production data*.[10]

```
 62      SPI_InitTypeDef SPI_InitStr;// konfiguracni struktura pro SPI komunikaci
 76      GPIO_PinAFConfig(GPIOA, GPIO_PinSource5, GPIO_AF_SPI1);//spojeni SPI1 pinu s
alternatyvni funkci
 77      GPIO_PinAFConfig(GPIOA, GPIO_PinSource7, GPIO_AF_SPI1);// spojeni SPI1 pinu s
alternatyvni funkci
 78      RCC_APB2PeriphClockCmd(RCC_APB2Periph_SPI1, ENABLE);// Zapnuti CLK pro funkci SPI1
 79      SPI_InitStr.SPI_Direction = SPI_Direction_1Line_Tx;// nastaveni vysilani dat, data
se posilaji do registru Tx, jenom vysilani dat
 80      SPI_InitStr.SPI_Mode = SPI_Mode_Master;// nastaveni STM32F4 jako Master pro SPI
komunikaci
 81      SPI_InitStr.SPI_DataSize = SPI_DataSize_16b; // delka vysilanych dat
 82      SPI_InitStr.SPI_CPOL = SPI_CPOL_Low;         // CLK standartne na 0 V
 83      SPI_InitStr.SPI_CPHA = SPI_CPHA_1Edge;       // data vysilat na nabeznou hranu
 84      SPI_InitStr.SPI_NSS = SPI_NSS_Soft | SPI_NSSInternalSoft_Set; // slave select
 85      SPI_InitStr.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_8;// SPI frekvence je
APB2(84MHz)/8
 86      SPI_InitStr.SPI_FirstBit = SPI_FirstBit_MSB; // prvni se posila MSB
 87      SPI_Init(SPI1, &SPI_InitStr);// konfiguracni strukturu nahraj do nastavujcich
registru
 88      SPI_Cmd(SPI1, ENABLE); // zapnuti SPI1
```

Stejně jako při inicializaci pinů, je třeba i pro SPI1 zapnout hodinový signál na sběrnici AHB, avšak z katalogového listu víme že SPI1 je připojeno na sběrnici AHB2.[10] Do struktury pro SPI je nutno nastavit:

- mód Master nebo Slave, pro tuto práci stačí data posílat, tedy Master
- šířku posílaných dat, ta je určena velikostí posuvného registru - 16 bit
- děličku výstupního hodinového signálu
- první bit, který se bude odesílat, tzn. MSB nebo LSB
- vysílání dat na náběžnou nebo sestupnou hranu
- základní hladinu výstupu hodinového signálu při nečinnosti

Poslední dva body jsou dané vlastnostmi LED budičů, proto bylo zvoleno, že data budou posílána na náběžnou hranu a základní hladina CLK bude v log. 0. Odesílání prvního bitu MSB, bylo určeno podle funkce pro zpracování dat, popis této funkce je v kapitole 4.1.3. Komunikační rychlost byla nastavena jako 10,5 MHz. Nakonec je třeba tuto strukturu opět převést do nastavení příslušných registrů a k tomu slouží funkce SPI_Init. Pro správnou funkci je třeba SPI1 povolit funkčnost, to se provede funkcí SPI-Cmd.

Inicializace poslední klíčové části, USARTu je podobná jako pro SPI, proto přeskočíme k podstatným parametrům, které udávají vlastnosti komunikace této sběrnice.

```
98      USART_InitTypeDef USART_InitStr; // konfiguracni struktura pro USART
125     USART_InitStr.USART_BaudRate = baudrate;// komunikacni rychlost
126     USART_InitStr.USART_WordLength = USART_WordLength_8b;// sirka odesilaných dat
127     USART_InitStr.USART_StopBits = USART_StopBits_1; // pocet stop bitu
128     USART_InitStr.USART_Parity = USART_Parity_No;  // paritni bit
129     USART_InitStr.USART_HardwareFlowControl = USART_HardwareFlowControl_None;//HW kon.
130     USART_InitStr.USART_Mode = USART_Mode_Tx | USART_Mode_Rx; // prijem i odesilani
131     USART_Init(USART1, &USART_InitStr);
146     USART_Cmd(USART1, ENABLE);
```

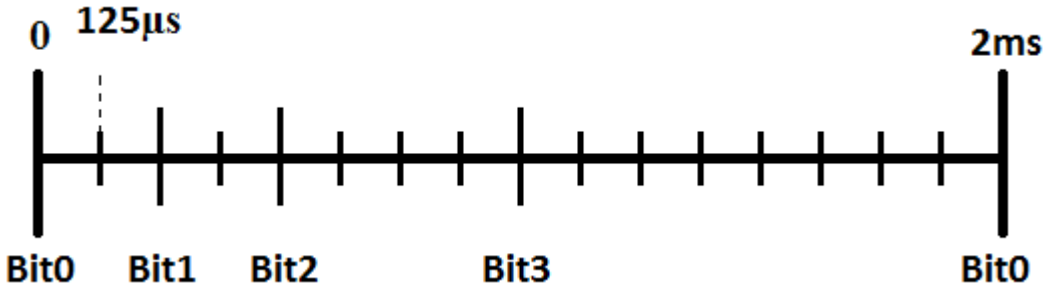
Jak je vidět z ukázky kódu, opět se jedná o ten samý princip inicializační struktury. Rychlost komunikace je zde zastoupena proměnnou, která je vstupem inicializační funkce. Rychlost byla nastavena na nejvyšší možnou, a to je 115200 baud. Šířka posílaných dat byla nastavena na standartních 8 bit. Stop bit v rámci urychlení odesílaných dat, byl nastaven jako 1 bit, ze stejné podmínky byla vypnuta i parita. Hardwarová kontrola taktéž není zapotřebí. V poslední řadě potřebujeme data jak přijímat, tak i odesílat, pro kontrolu a hlášení stavu. Takto nastavenou komunikaci je třeba nastavit i na druhé straně. Funkcí

USAT_Init, nastavíme znovu potřebné registry. Stějně jako u SPI, na konci nesmí chybět povolení funkčnosti USART, volíme tedy funkci USART_Cmd.

4.1.2 Obsluha přerušení SysTick

Jedná se o jednu z nejpodstatnějších funkcí, jednak zde dochází k přepínání pater, ale také k odesílání dat formou Bit Angle Modulation. Všechny tyto děje se řídí na základě hodnoty proměnné *cas*, tato proměnná je inkrementována při každém přerušení tzn. každých 125 µs. Tento časový interval byl zvolen s ohledem na čtyř bitovou Bit Angle Modulation a s požadavkem na celkovou obnovovací frekvenci kostky. Obnovovací frekvence byla zvolena, jako 16 ms (62,5 Hz) z toho vyplívá délka sepnutého patra jako 2 ms (500 Hz). Jak již bylo zmíněno, čtyř bitová Bit Angle Modulation, nabývá hodnot od 0 do 15, tudíž délku jedno patra je nutno vydělit 16, abychom získali potřebný údaj pro přerušení a to je tedy 125 µs.

Bit Angle Modulation (dále jen BAM), také známá pod názvem Binary Code Modulation, je jakousi formou pulzně šířkové modulace (dále jen PWM), ale její nároky na výpočetní výkon jsou v porovnání s PWM, zanedbatelné. Hlavním rozdílem mezi PWM a BAM je, že BAM se řídí váhou jednotlivých bitů svého rozlišení. Kdybychom zde chtěli použít čtyř bitové PWM, bylo by potřeba při každém přerušení, šestnáct krát za periodu patra, odesílat nová data, k právě danému okamžiku. Jelikož při použité frekvenci SPI (10,5 MHz) trvá odesílání dat (192 bit) 18,3 µs, tak by procesor byl poměrně často vytížen a nezbývalo by moc času na výpočet animací. U BAM nám za to stačí tato data poslat pouze čtyři krát. Jelikož jsme schopní si uvědomit váhu jednotlivých bitů, rozdělíme celou periodu do nelineárních intervalů, které odpovídají váze jednotlivých bitů. Grafické znázornění je na Obr. 19. Uveďme příklad na čísle 10, to je v binární podobě zapsáno jako 1010. Tato hodnota nám značí, že interval 2 a 8 - bude dioda svítit, a tím se dohromady složí číslo 10. Tato metoda je nevhodná pro řízení motorů.



Obr. 19 Bit Angle Modulation

K docílení takto rozdělených intervalů slouží *switch*, který je řízen proměnou *cas*, jak je znázorněno na kódu níže. Pro odesílání dat bylo vytvořeno 12 polí, každé pole je typu uint8 o délce 64, tudíž jedno pole obsahuje 512 bit. Každý bit pole představuje jednu diodu v kostce, pro daný časový usek. Každá barva má tedy svá čtyři pole, pro každý časový úsek jedno. Pro odesílání těchto polí byla napsána funkce *SPI_send*, jejímiž vstupy jsou aktuální pole a patro.

```
148  void SysTick_Handler(void){      //preruseni na piny pro patra
149  if(!usart_data_received && !usart_num_anim && !usart_length_anim &&
        !usart_time_anim && !usart_letter){// kontrola 0 pro posilani dat
150      if( cas == 0 ){
151          switch(patro){
152          default: break;
153          case 0:
154              GPIOE->BSRRH = GPIO_Pin_15;
155              GPIOE->BSRRL = GPIO_Pin_8;
158              break;
```

```
159             case 1:
160                 GPIOE->BSRRH = GPIO_Pin_8;
161                 GPIOE->BSRRL = GPIO_Pin_9;
163                 break;
190         switch(cas){
191             default: break;
192             case 0:
193                 pulse_ledm();
195                 SPI_send(patro*0x8,red1,green1,blue1);
197                 break;
198             case 2:
199                 pulse_ledm();
200                 SPI_send(patro*0x8,red2,green2,blue2);
201                 break;
202             case 4:
203                 pulse_ledm();
204                 SPI_send(patro*0x8,red3,green3,blue3);
205                 break;
```

Jak je patrné z příkladu kódu, pro časový interval 0 se odesílají data pro interval 2. Toto řešení bylo nutné, kvůli přesnému zobrazování dat v daný okamžik. Funkce *pulse_ledm*, zajišťuje dostatečně dlouhý impulz pro vyklopení dat z vnitřního registru LED budiče na výstupní piny. Tyto funkce se nevykonávají, pokud probíhá přenos dat po USART, to zajišťuje podmínka na začátku s příznakovými proměnnými.

4.1.3 Data animací

Zásadním faktorem pro tvorbu a zobrazování animací, je způsob uchovávání dat animace v paměti. Za tímto účelem je nutno vytvořit pole, do kterého se data budou ukládat v daném pořadí a v příslušném formátu. Nejlogičtější volbou v našem případě bude pole o rozměrech 8x8x8, výhody jsou patrné na první pohled. Hlavní výhodou toho pole je snadnější představa rozmístění rozsvícených diod v kostce, a zároveň nám jednotlivé buňky pole mohou uchovávat informaci o barvě dané diody. Vytvoříme-li takovéto pole, tak uchováme jeden snímek celé kostky, což pro animaci, která má být vizuálně zajímavá není dostačující. Nabízí se nám tedy dvě volby.

Pole o rozměrech 8x8x8, můžeme v pravidelných intervalech měnit, podle žádaného algoritmu. Tato volba byla zvolena pro animace integrované přímo ve firmwaru. Výhodou toho řešení je zobrazování animací, které potřebují pro svou funkci delší časový interval, s vyšším počtem snímků, nebo také neopakující se sekvence (např. náhodná barva). Nebo jako druhý způsob, můžeme naše trojrozměrné pole rozšířit o čtvrtý rozměr, který bude uchovávat časovou posloupnost snímků. K tomuto poli je vhodné vytvořit ještě časový vektor, který bude udávat délku zobrazování jednotlivých snímků. Doba, po kterou bude svítit jeden snímek, může být daná, ale pouze by se tím omezili možnosti animace. Tato možnost uchovávání animací byla použita pro online řízení z nadřazeného systému. Výhodou druhého způsobu uchování dat, je animace uložená v paměti se všemi svými stavy a hodnotami, tudíž není potřeba žádný výpočet. Omezení toho řešení je pouze paměť zařízení. Pro tuto práci tedy byly vytvořeny dvě pole - kostka3d[8][8][8] a kostka3d_animace[8][8][8][200] a jsou typu uint8. Tento typ proměnné odpovídá formátu BMP uloženém v 256 barevném rastru, tedy osm bitů. Protože informace o barvě je uložena ve formě čísla od 0 po 255, je třeba tabulka barev (angl. Color table), která udává poměr rozsvícení jednotlivé barvy (RGB) vůči obnovovací frekvenci. Tabulka je v programu představována třemi vektory, které byly upraveny pro čtyř bitový BAM.

Jsou-li data animací takto připraveny a uloženy, je třeba jejich informace přenést ve správném formátu do polí pro BAM. Za tímto účelem byly napsány tři funkce *led*, *vypocet* a *vypocet_pc_anim*. Funkce led, slouží přímo pro zápis do polí pro BAM, jejími vstupními parametry jsou souřadnice *x,y,z* (sloupec, řádek, patro), jednotlivé diody, které chceme

nastavit a doba (0-15) rozsvícení jednotlivých barev. Na následujícím přikladu je uvedena část kódu této funkce.

```
419  void led(uint8_t radek, uint8_t sloupec, uint8_t level,uint8_t red, uint8_t
green,uint8_t blue){
420      uint8_t whichbyte = level*8 + radek;
421      uint8_t pomoc=0x01;
422      pomoc=pomoc<<sloupec;
423      if( red & 0x01)
424          red0[whichbyte]|=pomoc;
425      if( red & 0x02)
426          red1[whichbyte]|=pomoc;
427      if( red & 0x04)
428          red2[whichbyte]|=pomoc;
429      if( red & 0x08)
430          red3[whichbyte]|=pomoc;
```

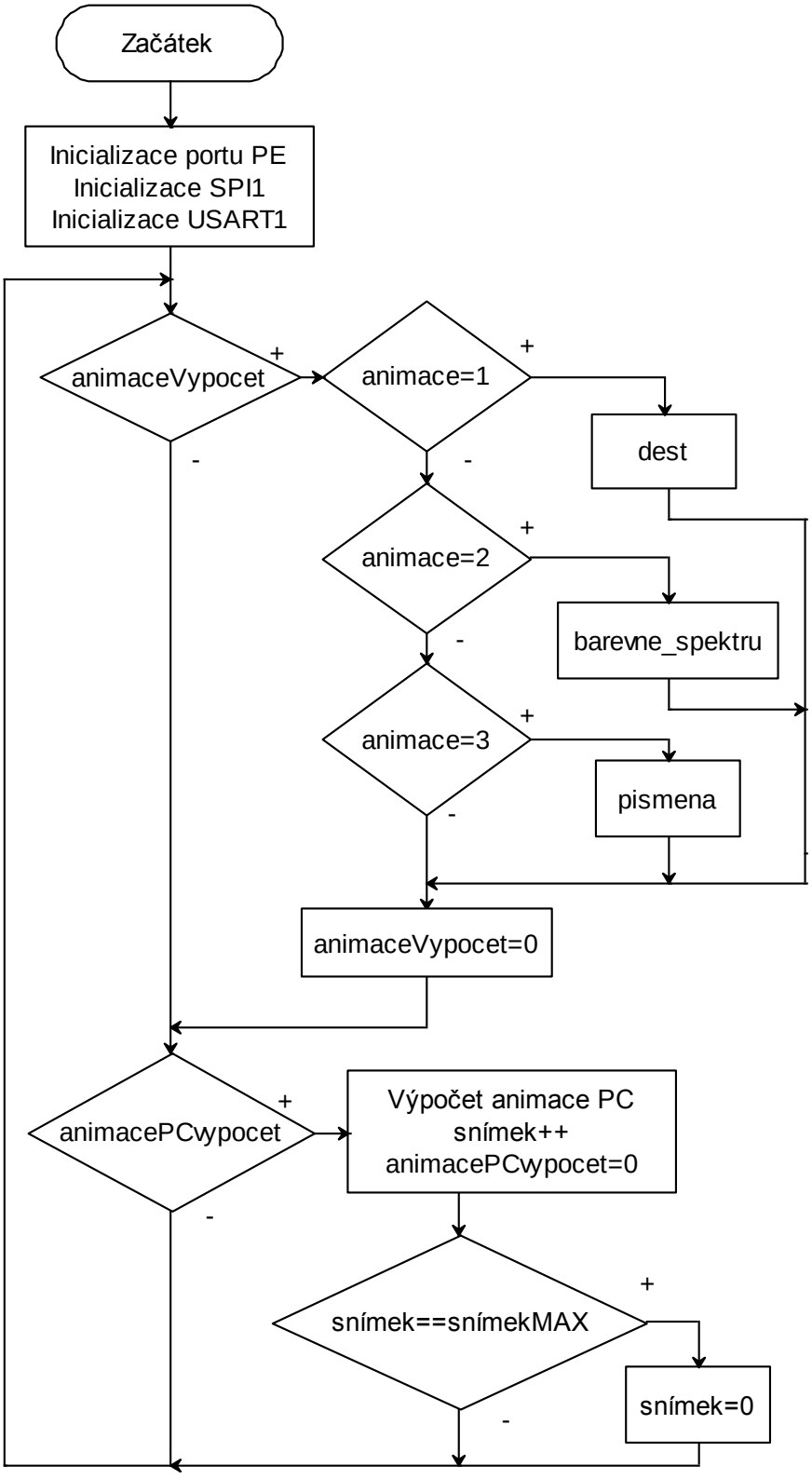
Na kódu lze vidět, že souřadnice pro řádek a patro, jsou použity pro určení správné buňky v poli, a souřadnice pro sloupec určuje konkrétní bit, který chceme nastavit. Dále probíhá test všech barev, bit po bitu a případné plnění požadovaného pole. Jedním z důvodů vytvoření této funkce je přímá tvorba animaci, avšak tento způsob je komplikovanější k zadávání barvy diody a orientace v prostoru.

Funkce *vypocet* používá funkci *led*, tak že prochází pole *kostka3d* po jednotlivých buňkách a v případě, že je hodnota pole nenulová, volá funkci *led*. Pro správnou funkci je nutné nejdříve vynulovat pole pro BAM. Jelikož vstupem funkce *led* jsou čísla od 0 do 16, tak zde probíhá konverze přes tabulku barev. Funkce *vypocet_anim* je založena na stejném principu, je pouze rozšířena o informaci aktuálního snímku. Na následujícím úryvku kódu je zobrazena podstata funkce *vypocet*.

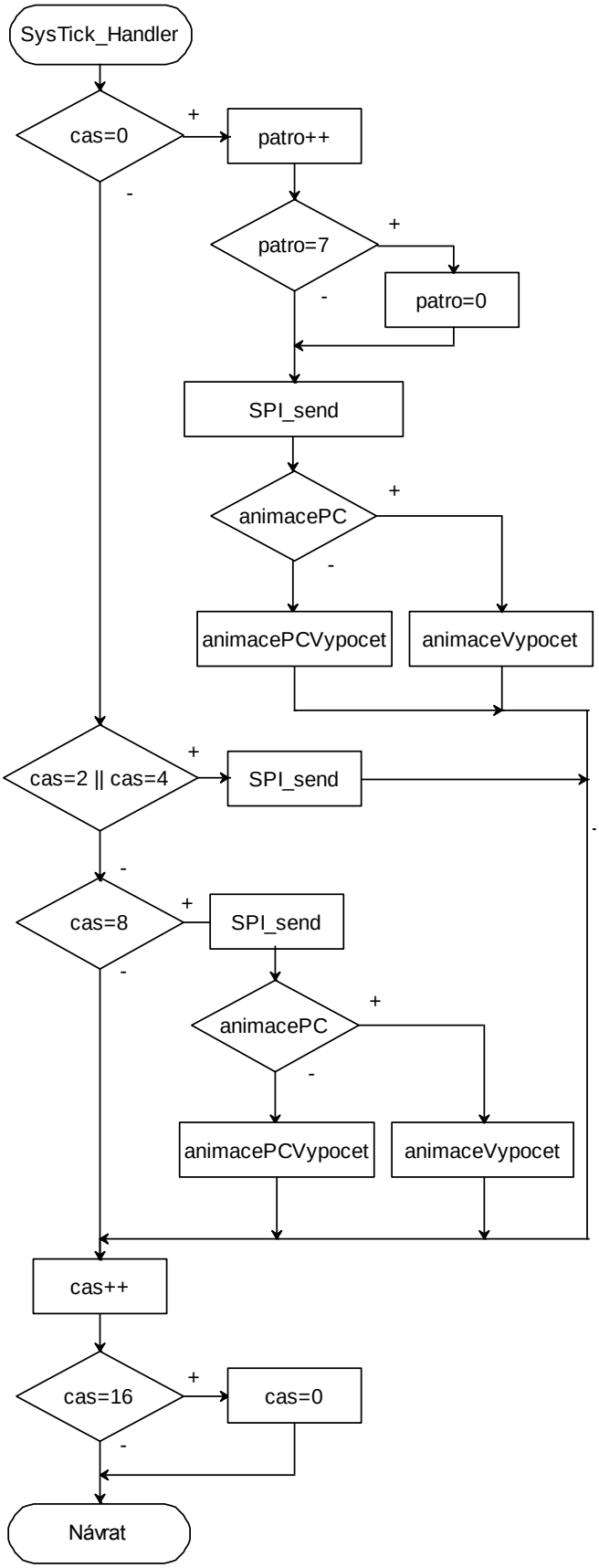
```
467      for(i=0;i<8;i++){
468          for(o=0;o<8;o++){
469              for(p=0;p<8;p++){
470                  if(kostka3d[p][o][i]!=0){
471
led(p,o,i,red_t[kostka3d[p][o][i]],green_t[kostka3d[p][o][i]],blue_t[kostka3d[p][o][i]]);
```

4.1.4 Vývojové diagramy

Vývojový diagram hlavního programu a nekonečné smyčky.



Vývojový diagram obsluhy přerušení.



4.1.5 Komunikační protokol

Celý komunikační protokol je implementován ve funkci *USART1_IRQHandler*. Tato funkce je přerušením při přijmu dat. Funkci na začátku, nesmí chybět kontrola dokončení přenosu dat. Celou funkci následně vyplňují dva druhy podmínek. První jsou podmínky pro řídicí znaky, dále následují podmínky pro následné zpracování. Podmínky pro řídicí znaky spouští sekvenci k přípravě pro příjem dat, vypínají svojí funkci a zapínají proměnné, pro podmínky na následný příjem dat.

Komunikační protokol byl navržen tak, že prvním znakem, který je přijat, se ověřuje stav zařízení, zda je připojeno. Tento znak lze opakovaně odesílat, do doby odeslání řídicího znaku. Řídicí znak vybírá účel a zpracování následujících dat. Při přijímání dat delších znakových nebo číselných řetězců, je implementována průběžná kontrola stavu zařízení. Toho je docíleno pomocí znaků pro synchronizaci přenosu.

V Tab. 1 jsou uvedeny všechny znaky a jejich funkce.

Znak	Typ znaku	Funkce
s	Inicializační	Znakem se ověřuje stav zařízení
a	Řídicí	Změna animace
d	Řídicí	Sekvence pro odeslání animace z PC
l	Řídicí	Změna textu v animaci *pismena*
k	Řídicí	Změna parametrů animace *dest*
r	Synchronizační	Výzva k sekvenci odesílání dat
o	Synchronizační	Hlášení o stavu přijmutích dat ok
e	Synchronizační	Hlášení o ukončení sekvence příjmu dat

Tab. 1 Znaky komunikačního protokolu

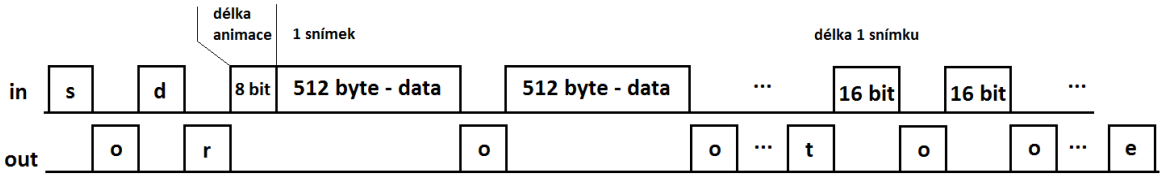
Po přijetí řídicího znaku se spustí sekvence pro nastavení příjmu dat a provede se potřebné nastavení vnitřních proměnných. Dále následuje odeslání znaku *r* jako výzva k odesílání dat nadřazeného systému. Příjem dat probíhá buď s hlášením o stavu zařízení anebo bez něj, to je ovlivněno typem přijímaných dat. Po ukončení příjmu dat se spouští sekvence, která zajišťuje nastavení všech používaných proměnných do výchozího nastavení, jakmile se vše provede, je odeslán synchronizační znak *e* jako potvrzení správného příjmu dat a ukončení tohoto děje.

Po přijetí řídicího znaku *a* se spustí přijímací sekvence a očekává se příjem jednoho čísla, určujícího implementovanou animaci, která se má přehrávat. Řídicí znak *d* zahajuje sekvenci pro animaci vytvořenou na PC, po odeslání znaku *r* se očekává příjem délky animace, prezentovanou osmi bitovou proměnnou. Jakmile je známá délka animace, očekává se příjem 512 byte (1 snímek). Po přijetí těchto dat, se odešle synchronizační znak *o*, jako hlášení o stavu. Po dokončení příjmu všech snímků, se očekává příjem stejně dlouhého vektoru casu, s dobou trvání jedno snímku. Jelikož doba trvání jednoho snímku je zastoupena šestnácti bitovým číslem, bylo nutné vytvořit algoritmus, který příjme dvě 8 - bitová čísla a spojí je v jedno. Nakonec po úspěšném přijetí všech dat se odešle ukončovací znak *e*. Na následujícím úryvku kódu je znázorněno, jak probíhá přenos dat animace.

```
234         if( usart_data_received ){//prijem dat pro animaci
235             kostka3d_animace[cnt_x][cnt_y][cnt_z][cnt_frame]=data;//plneni pole daty
236             cnt_y++;
237             if(cnt_y==8){//kontrola hodnoty radku
238                 cnt_y=0;cnt_x++;//nulovani radku a presun do dalsiho sloupce
239                 if(cnt_x==8){//kontrola hodnoty sloupce
240                     cnt_x=0;cnt_z++;//nulovani radku a presun do dalsiho sloupce
241                     if(cnt_z==8){//kontrola hodnoty patra
242                         cnt_z=0;cnt_frame++;//nulovani patra a presun do dalsiho snimku
243                         USART_puts(USART1,'o');//kontrola stavu
244                         if(cnt_frame==length_anim){//kontrola konce animace
245                             cnt_frame=0;
246                              usart_data_received=0;//ukonceni prenosu dat animace
247                              usart_time_anim=1;//prepnout na prijem casoveho vektoru
```

Proměnné pro určení aktuální pozice pole musí být typu *static,* jelikož se nacházíme ve funkci přerušení. Průběžné hlášení o stavu zařízení dále probíhá při přenosu vektoru času. Hlášení o stavu, tedy znak *o*, se odešle za každým přijatým údajem. Řídicím znakem *l* se program nastaví do stavu, kdy očekává příjem pole znaku pro animaci písmena. Jakmile je přenos zahájen, provádí se kontrola každého znaku, dokud se nepřijme znak @, tento znak ukončí sekvenci příjmu dat.

Na Obr. 20 je graficky znázorněna výměna dat s nadřazeným systémem při příjmu dat animace vygenerované v PC.

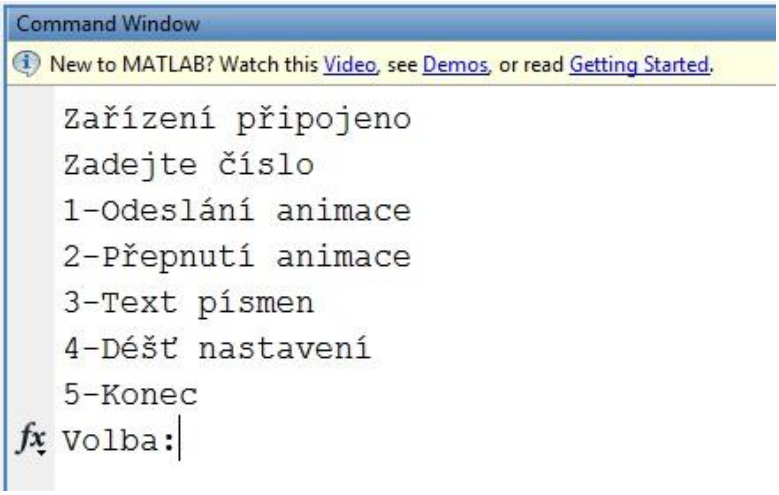


Obr. 20 Výměna dat při přijímání animace

4.2 Nadřazené ovládání z PC

Pro ovládání LED kostky z PC, byl vytvořen skript pro program Matlab. Výhodou této volby je jednoduchost ovládání sériového portu, snadná tvorba více rozměrových matic a jejich následovné zpracování a ukládání, univerzálnost Matlabu je však jeho hlavní výhodou. Program byl vytvořen formou stavového automatu. Na začátku program vyžaduje číslo COM portu, ke kterému je LED kostka připojena. Tato informace se jednoduše zjistí ve správci zařízení, pro operační systém Windows. Následuje vytvoření sériového portu, funkcí *serial,* s požadovanými parametry, které jsou uvedeny v kapitole 4.1.1.

Port je nutno otevřít funkcí *fopen*, aby bylo možné použít příkazy k odesílání a přijímání dat. Dále probíhá připojování zařízení a indikace stavu LED kostky. Následuje výpis možností pro ovládání kostky. Výpis můžeme vidět na Obr. 21.



Obr. 21 Výpis skriptu v Matlabu

Vybráním první možnosti, se do příkazového okna vypíšou potřebné instrukce k vložení animace, kterou chceme odeslat. Ta musí být uložena v samostatném souboru. Ten musí být typu *mat.* Tento soubor musí obsahovat dvě proměnné, první je *animace.* Tato proměnná musí být pole, o rozměrech 8x8x8x200. Poslední rozměr nemusí být nutně 200, tato hodnota je maximální. Druhá proměnná se musí nazývat *time*, musí mít stejný rozměr, jako čtvrtý rozměr matice s animací. Každý snímek tedy bude mít svojí individuální dobu zobrazení. Po zadání toho souboru, program dále kontroluje správnost dat, které chceme odeslat. V případě že data neodpovídají požadovanému formátu nebo nemají správnou hodnotu, tak se do příkazového okna vypíše hlášení o špatném formátu zadávaných dat a čeká se na potvrzení stisknutím *Enter*, po stisknutí se program vrátí zpět do hlavního menu. Proběhne-li kontrola dat v pořádku, zahájí se odesílání dat podle komunikačního protokolu. Během odesílání dat, se do příkazového okna vypisuje stav procesu, a počet odeslaných snímků. Posléze počet odeslaných dat z proměnné *time*. Následně se vypisuje hlášení o úspěchu nebo o neúspěchu přenosu. Hlášení se vyhodnocuje na základě obdržení synchronizačního znaku *e*. Toto hlášení se provádí pro všechny možnosti z hlavního menu.

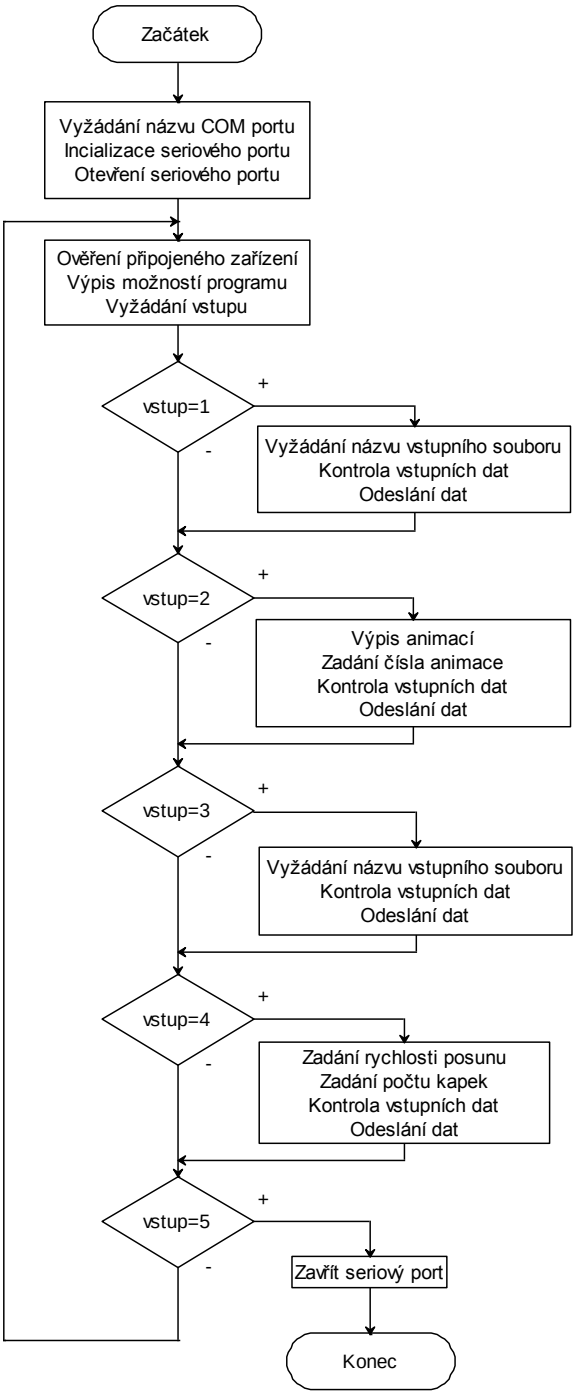
Druhá možnost, přepnutí animace, po zadání vypíše všechny dostupné animace, každé animaci je přiřazena hodnota, podle které se vybírá. Po zadání požadované animace proběhne kontrola správnosti údaje a následně odeslání čísla animace, kterou chceme zobrazit. Třetí možností je změna znaku v animaci písmena. Zadáním této možnosti se vypíše hlášení o požadovaném formátu a typu dat. Opět se jedná o soubor typu *mat*, který má však v sobě uloženou jednu proměnou s názvem *text*. Tato proměnná může obsahovat maximálně 31 znaků, přičemž posledním znakem v řetězci, musí byt znak @, v animaci je tedy možné zobrazovat 30 znaků. Dále se zobrazí požadavek pro zadání času, který určuje dobu přechodu písmene mezi jednotlivými řádky. V průběhu odesílání dat, je opět uveden výpis kolik znaků se již odeslalo a stav zařízení.

Předposlední volbou je nastavení animace *dest*, znovu se zde vypisují požadavky pro zadávána čísla. Tyto hodnoty určují počet kapek na patro a čas mezi posuvy pater. Čísla logicky nemůžou být záporná ani nulová, maximální hodnota pro počet kapek je 64, počet diod na patře. Hodnota pro čas mezi posuvy, je uložena v šestnácti bitové proměnné, může tedy nabývat maximální hodnotu tohoto typu. Poslední možností je ukončení programu. Touto volbou se přeruší nekonečná smyčka a dojde k uzavření sériového portu funkcí *fclose*.

K odesílání dat jsou využity dvě funkce, *fprintf* je použita k odesílání znaků. Druhou funkcí je *fwrite*, ta odesílá číselné hodnoty. Obě dvě funkce vyžadují jako první parametr, název sériového portu. Pro příjem dat je použita funkce *fread*, prvním

parametrem je opět název portu, druhým se určuje počet přijatých byte. Návratovou hodnotou funkce *fread,* jsou přijatá data. Pro účely této práce, stačí přijímat pouze jeden znak. V rámci této práce byl také vytvořen skript na tvorbu matice, v které se vykresluje funkce sinus, touto maticí lze ověřit funkci online řízení z PC.

Dojde-li k náhodnému přerušení komunikace, odpojení kabelu USB, (například) Matlab ukončí vykonávání programu v místě, kde dojde k chybě. V důsledku toho nedojde k zavolání funkce *fclose* a sériový port zůstává otevřený. Pokud bychom chtěli v práci pokračovat, je nezbytné port uzavřít, zadáním potřebné funkce. Neučiníme-li tak a program bychom spustili znovu, dojde k opětovnému volání funkce *fopen.* V řadě, tím dojde v Matlabu k chybě, kterou můžeme odstranit pouze restartováním programu.



5 Závěr

Cílem práce bylo navrhnout barevný 3D zobrazovač jak po stránce konstrukční a elektronické, tak i po stránce softwarové. Konstrukční řešení lze děli do několika skupin. Vlastní konstrukce LED krychle, deska budičů, napěťový regulátor, řídicí deska. Po připojení napájení kostka automaticky začne přehrávat animaci *pismena* s předdefinovaným řetězcem se znaky VUT. Po připojení k nadřazenému systému lze integrované animace měnit nebo nahrávat nové vygenerované v PC.

Pro buzení LED se velmi osvědčili LED budiče STP16DP05, jejich jednoduchá konstrukce a ovladatelnost, dovoluje budit 192 diod na jednom patře, za pomocí třech vodičů. Patra jsou přepínaná postupně v takové obnovovací frekvenci, že to lidský zrak nedokáže rozeznat. Tranzistory použité k spínání pater jsou mírně předimenzované, avšak to funkčnosti nepřekáží ba naopak při spínání plného impulzního proudu 4A se ani mírně nezahřejí. Při výběru LED diod pro realizaci kostky, bylo nutné se řídit několika kritérii – jako je dostatečná svítivost a pozorovací úhel, taktéž měli přednost diody v difúzním pouzdře. Na Českém trhu se tyto diody vyskytují značně předražené oproti zahraničí, proto v rámci úspory byli diody objednáni z Číny. Vývojový kit STM32F4 se osvědčil jako spolehlivý prvek, ošetřený vůči nešikovnosti při zacházení. Během stavby a ladění ukázal značnou odolnost vůči zkratu. Modul napěťového regulátoru se ukázal jako velmi spolehlivý. Při testování v laboratoři dosahoval 80% účinnosti v oblasti spojitých proudů, při skokových změnách odebíraného proudu se odchylka výstupního napětí měnila v řádech desítek mV.

Při stavbě drátové části LED kostky, bylo nezbytné dodržovat přesnost mřížky a umístění diod v řádech milimetrů. Při větší odchylce byl výsledný tvar esteticky nepřijatelný a bylo nutné jej předělat. Postup stavby popsaný v této práci bych označil za velmi efektivní avšak časově náročný.

Řídicí program mikrokontroléru byl napsán a odzkoušen. Největším problémem, s kterým jsem se zde setkal, bylo správné odladění funkce *SysTickHandler* u této funkce je velmi důležité aby probíhala včas a bez zdržení. Občas se objeví náhodné chyby jako zaseknutí animace při častém nahrávání animací z PC. Do budoucna tento program by bylo vhodné rozšířit o možnost programování Flash paměti. Animace generované v PC by se tak mohli nahrát na do paměti a přehrávat jako úvodní animaci po spuštění. Dalším vhodným rozšíření by bylo naprogramování 3D her jako je například hra Had a Tetris. Posledním rozšířením by mohlo být přímé zobrazování vysílaných dat z PC, v nekonečné smyčce.

Skript v Matlabu byl napsán jako stavový automat, kterým vybíráme zobrazovaná data v kostce. Vylepšením toho skriptu by mohlo, být ošetření vůči spatně zadanému nazvu animace, program při nesprávném názvu se nesprávně ukončí (nevolá se fclose). Taktéž by jej bylo třeba v případě integrování her, do řídicího programu, rozšířit o možnost ovládání těchto her.

Kostka je jako celek funkční a kompaktní zařízení její umístění na robota FEKT BOT ještě nebylo provedeno. Avšak je připravena k usazení.

6 Seznam literatury

[1] STM32F4DISCOVERY. ST. [online]. [cit. 2015-01-06]. Dostupné z: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419

[2] 5^3 LED CUBE Controller. [online]. [cit. 2015-01-06]. Dostupné z: http://picprojects.org.uk/projects/lc/index.htm#Component_List

[3] HOW NOT TO ENGINEER: RGB LED Cube. [online]. [cit. 2015-01-06]. Dostupné z: http://www.hownottoengineer.com/projects/rgb-led-cube.html

[4] STP16DP05. ST. [online]. [cit. 2015-01-06]. Dostupné z: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00149163.pdf

[5] TIŠNOVSKÝ, Pavel. Externí sériové sběrnice SPI a I²C: SPI. ROOT.cz [online]. [cit. 2015-01-11]. Dostupné z: http://www.root.cz/clanky/externi-seriove-sbernice-spi-a-i2c/

[6] HW server. ŘEHÁK, Jan. Universal Serial Bus - Popis rozhraní. [online]. [cit. 2015-05-26]. Dostupné z: http://www.hw.cz/navrh-obvodu/rozhrani/usb/usb-universal-serial-bus-popis-rozhrani.html

[7] Datasheet - FT232RL, USB to serial UART interface. [online]. [cit. 2015-04-04]. Dostupné z: http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf

[8] Robot klub Rychnov. LOCKER, Martin. [online]. 2014-01-25 [cit. 2015-04-19]. Dostupné z: http://robotika.vosrk.cz/guide/arduino/lesson07/cs

[9] HW server. OLMR, Vít. [online]. 2005-12-12 [cit. 2015-04-19]. Dostupné z: http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html

[10] STM32F407VG: ST [online]. [cit. 2015-05-13]. Dostupné z: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00037051.pdf

[11] STEJSKAL, Martin a Dominik SCHMIDT. *Projekt do MMIA: RGB LED Cube 5x5x5* [online]. [cit. 2015-05-22]. Dostupné z: http://www.urel.feec.vutbr.cz/MIA/2013/stejskal/index.html

[12] TPS54560. Texas Instruments [online]. [cit. 2015-05-24]. Dostupné z: http://www.ti.com/lit/ds/symlink/tps54560.pdf

7 Seznam příloh

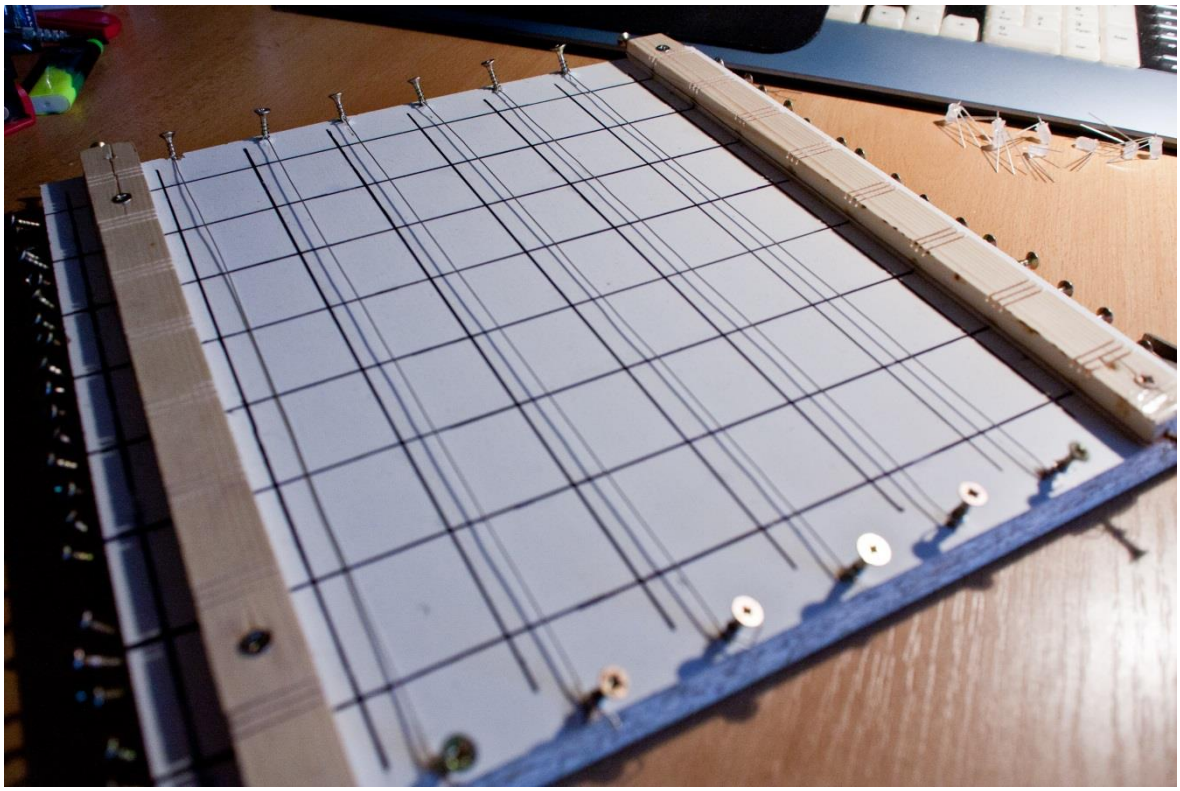
Příloha 1 = Podrobné obrázky stavby kostky

Příloha 2 = Schéma zapojení regulátoru napětí

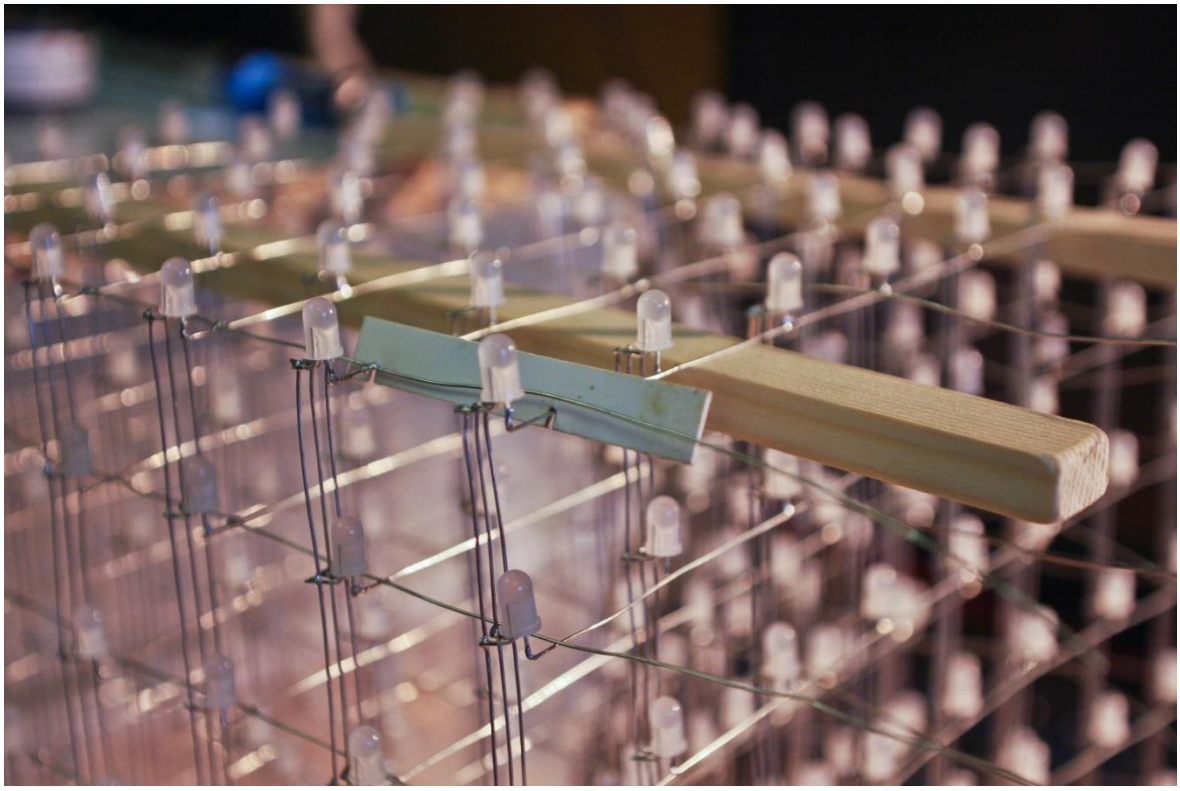
Příloha 3 = Masky DPS a osazovací plány

Příloha 4 = Celkové schéma obvodu, velikost A3

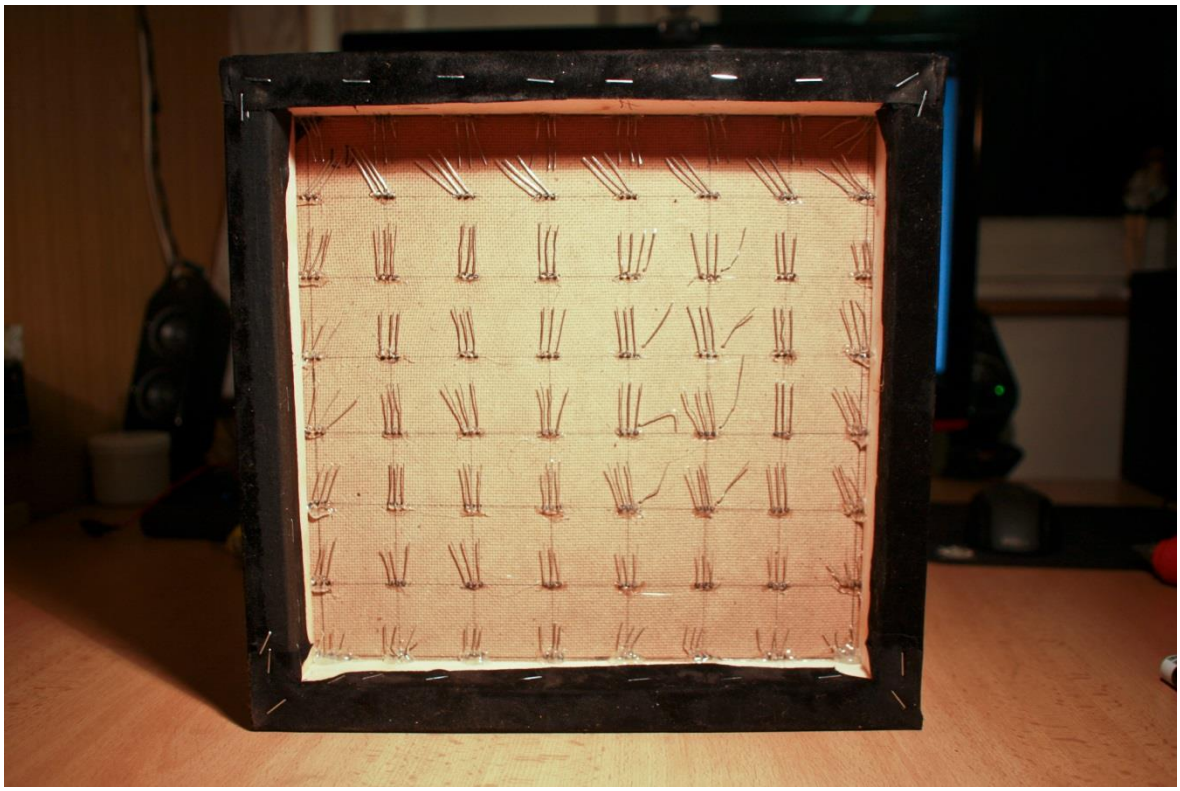
Příloha 1



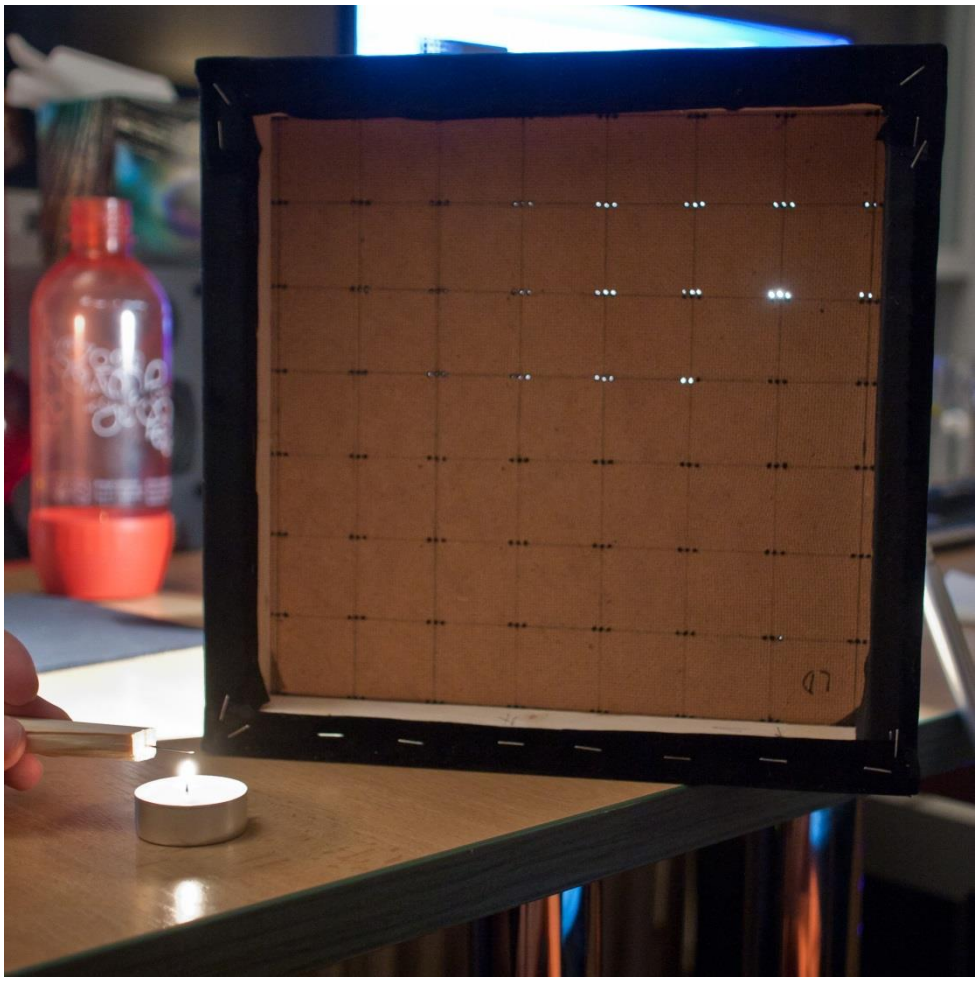
Vypnutí vodičů pro patra



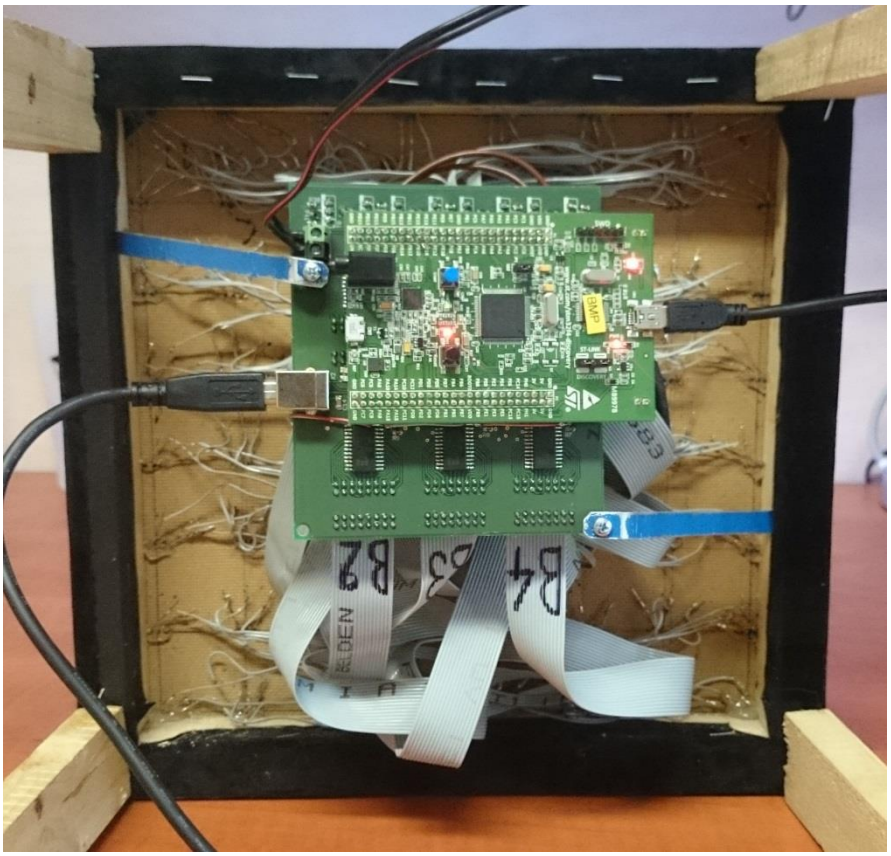
Ukázka použití šablonky při celkové kompletaci



Ukázka upevnění kostky v podstavci



Vytváření děr v tkanině, pro upevnění kostky v podstavci



Konečná podoba LED kostky pohled ze spodu

Hotová LED kostka

Příloha 2

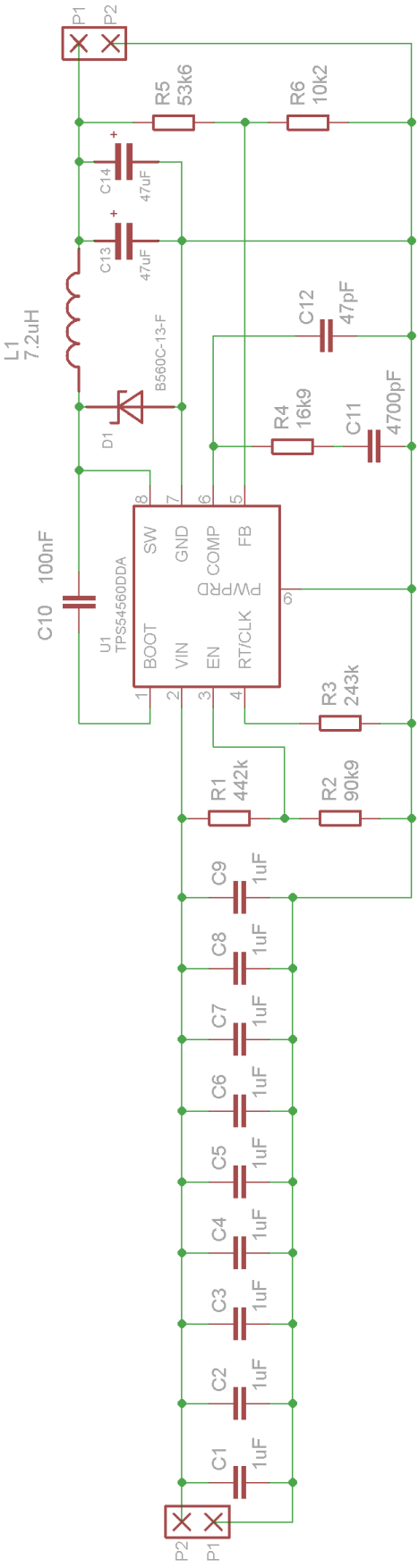
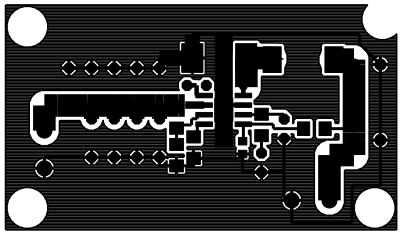
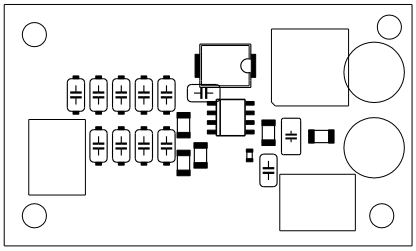


Schéma zapojení regulátoru napětí

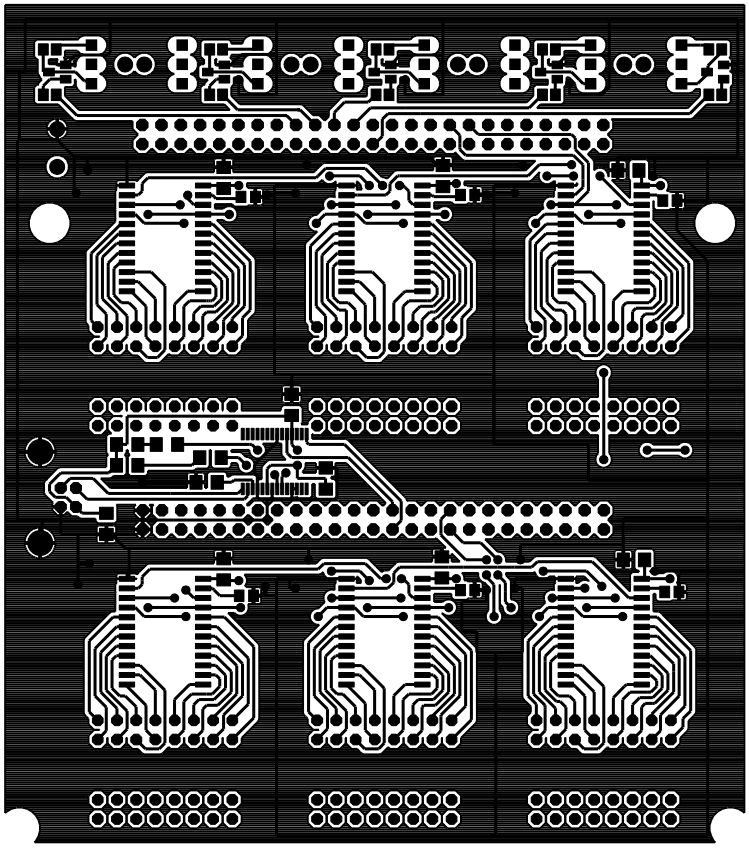
Příloha 3



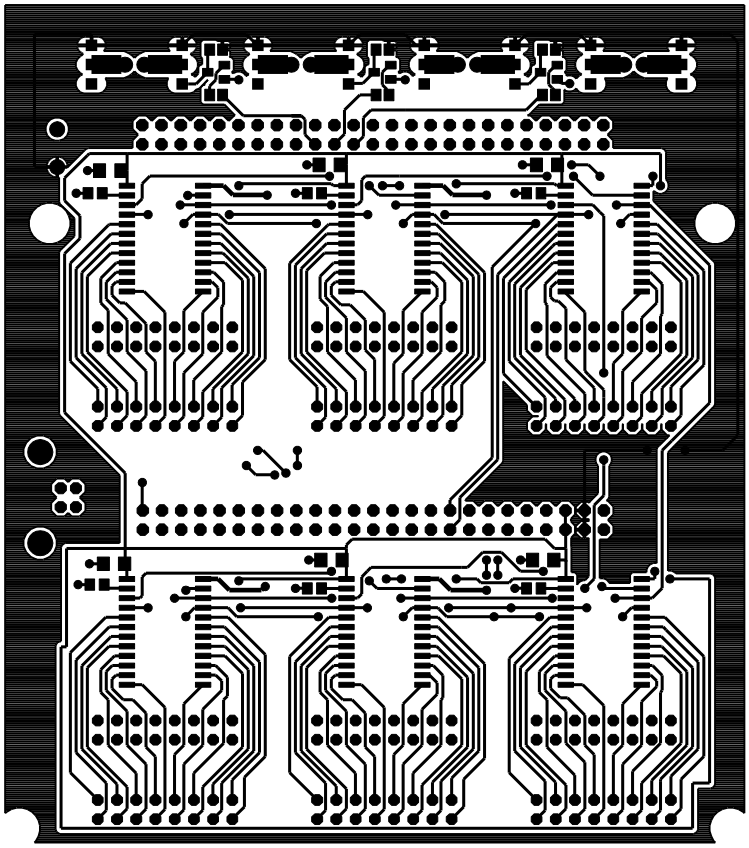
DPS regulátoru napětí, maska



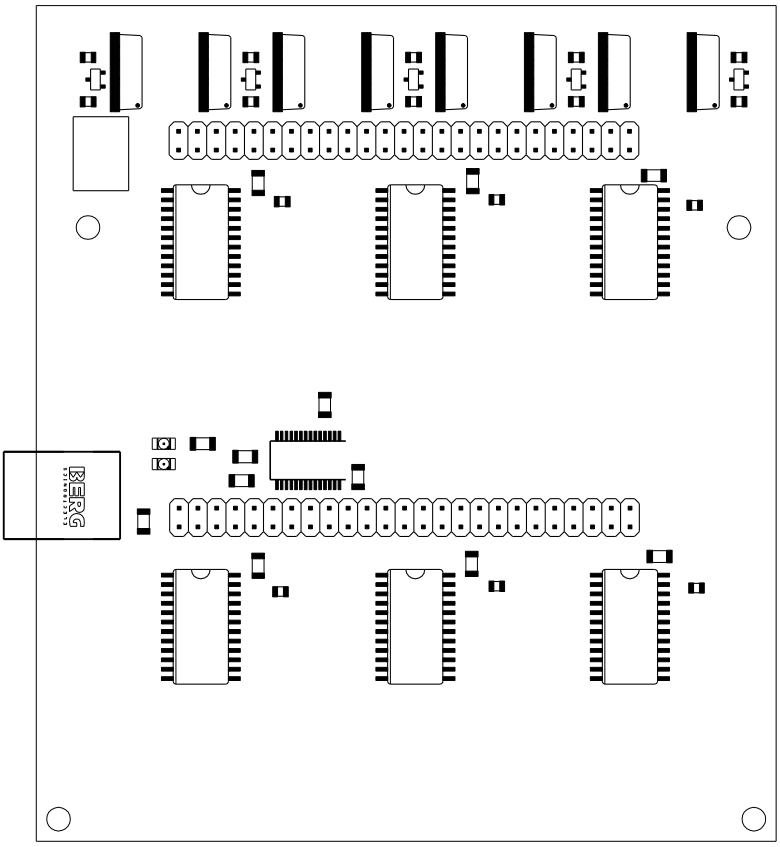
Osazovací plán vrchní strana



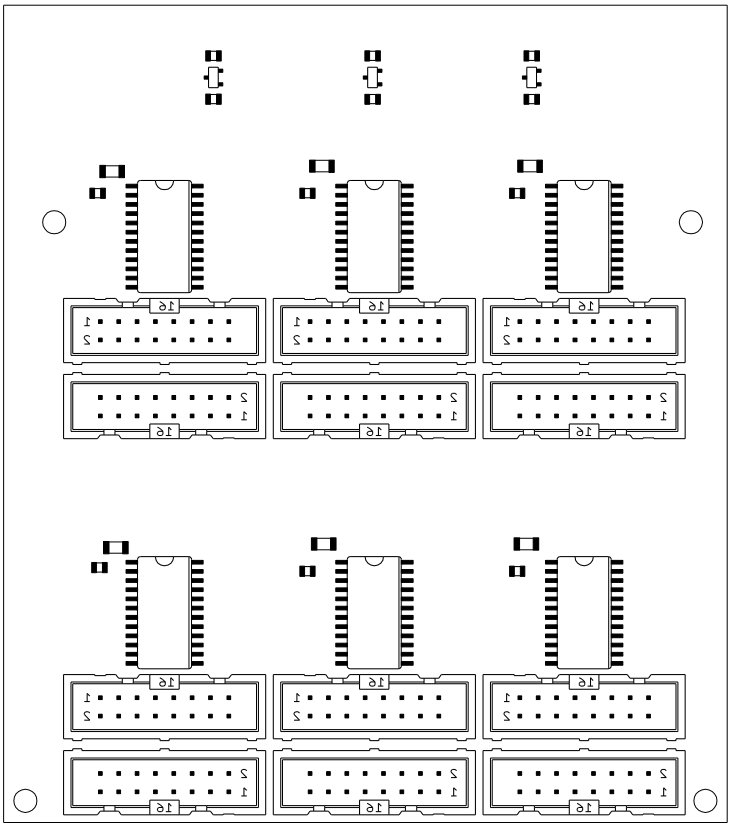
Vrchní maska DPS



Spodní maska DPS



Osazovací plán vrchní strana



Osazovací plán spodní strana