Středoškolská technika 2012

Setkání a prezentace prací středoškolských studentů na ČVUT

DIGITÁLNÍ ŘÍZENÍ NAVÍJEČKY CÍVEK

Vilém Kárský

Vyšší odborná škola a Střední průmyslová škola Šumperk

Generála Krátkého 1, Šumperk

Děkuji ing. Vítu Krňávkovi za obětavou pomoc a podnětné připomínky, které mi během práce poskytoval.
Děkuji Hynku Kárskému za pomoc s mechanickou částí práce.

Anotace: Elektronické řešení digitálního řízení navíječky cívek pomocí jednočipového mikropočítače Atmega8. Digital řízení umožňuje navíjení i odvíjení cívek. Při navíjení je hlídán posun vodiče, aby byly závity úhledně vinuty.

Klíčová slova: Atmega8, Navíječka cívek, Navíjení cívek, Digitální řízení,

Anotation: Electronic solutions of digital control coil winder by single-chip microcontroller Atmega8. Digital control allows winding the coils and unwinding the coils. Cable is monitored during widing coil, to neatly coils winding.

Key words: Atmega8, Coil winder, Coil winding, digital control

Obsah

1. Úvod

V této práci je řešena problematika návrhu digitálního řízení navíječky cívek. Práce vznikla v podstatě náhodou, když byla nalezena jedna kostra navíječky cívek. Jen tak mezi řečí, padlo, že by bylo zajímavé postavit k té kostře počítadlo. Při dalším přemýšlení přibývaly postupně nápady. Napřed, že by měla umět i odečítat otáčky při odvíjení, pak že by bylo dobré si moci zadat cílový počet závitů, další nápad byl, že by mohla sama i ovládat motor navíjení (odvíjení) a nakonec, že by bylo vhodné vyřešit i automatický posuv drátu, aby se drát nenavíjel jako chuchvalec, ale pěkně závit vedle závitu. A takto vznikla má maturitní práce. Dále zde budou rozebrány druhy navíječek, jejich výhody a nevýhody, požadavky na navíjení a způsob, jak jsem tuto problematiku řešil já.

2. Požadavky na navíjení cívek

Navíjení cívek se dá řešit mnoha způsoby, od ručního navíjení počínaje, až po složité automatické navíječky, které umí navíjet i toroidní cívky.

2.1. Ruční navíjení

Při úplném ručním navíjení, uživatel cívku navíjí sám pouze rukou. Navinuté závity si počítá uživatel také sám, může si pomoci tak, že si bude dělat čárku na papír např. po 10 závitech. Tento způsob je nejjednodušší ze všech, není k němu nic zapotřebí, není vhodný na navíjení cívek o velkém počtu závitů, protože uživatel, se může ztratit v počtu závitů, které již navinul. Tato metoda je vhodná např. pro vf cívky, které mají málo závitů (většinou jednotky až desítky závitů), ale vodič může být většího průměru (takového, jaký je uživatel schopný navinout vlastní silou).

Toto ruční navíjení se dá vylepšit tak, že jádro, na které cívku navíjíme, připojíme k elektromotoru. Navinuté závity musí uživatel buď počítat ručně, nebo může na hřídel připojit mechanické počítadlo. Takové navíjení je rychlejší než úplně manuální, ale posun vodiče, aby byl závit vedle závitu, opět zůstává na uživateli. Tímto způsobem navíjení se dají navíjet již cívky s větším počtem závitů (řádově až stovky). Průměr vodiče, který budeme schopni navíjet, závisí na výkonu použitého motoru.

2.2. Automatické navíjení

Nejjednodušší automatické navíjení je, pokud k jádru cívky připojíme elektromotor a elektronické počítadlo, na kterém nastavíme cílový počet závitů. Při navíjení se hodnota na počítadle dekrementuje a po dosažení nuly na počítadle se motor zastaví. Aby byl vodič na cívce dostatečně utažen, je nutné vodič napínat. Při napínání vodiče může dojít i k jeho zaseknutí a tak je vhodné u automatických systémů snímat, jestli vodič není zaseknutý a pokud je, tak zastavit navíjení. U těchto jednoduchých systémů si vystačíme i s jednoduchou logikou bez použití jednočipových mikropočítačů.

Složitější navíječky řeší i posun vodiče, aby byla cívka pěkně závit vedle závitu. Toto již jsou poměrně složité systémy, ve kterých je většinou nutné ovládat dva elektromotory – jeden navíjecí a druhý, který obstarává posun vodiče. U těchto systémů si již s jednoduchou logikou nevystačíme, a proto se zde již používají jednočipové mikropočítače a nebo PLC automaty.

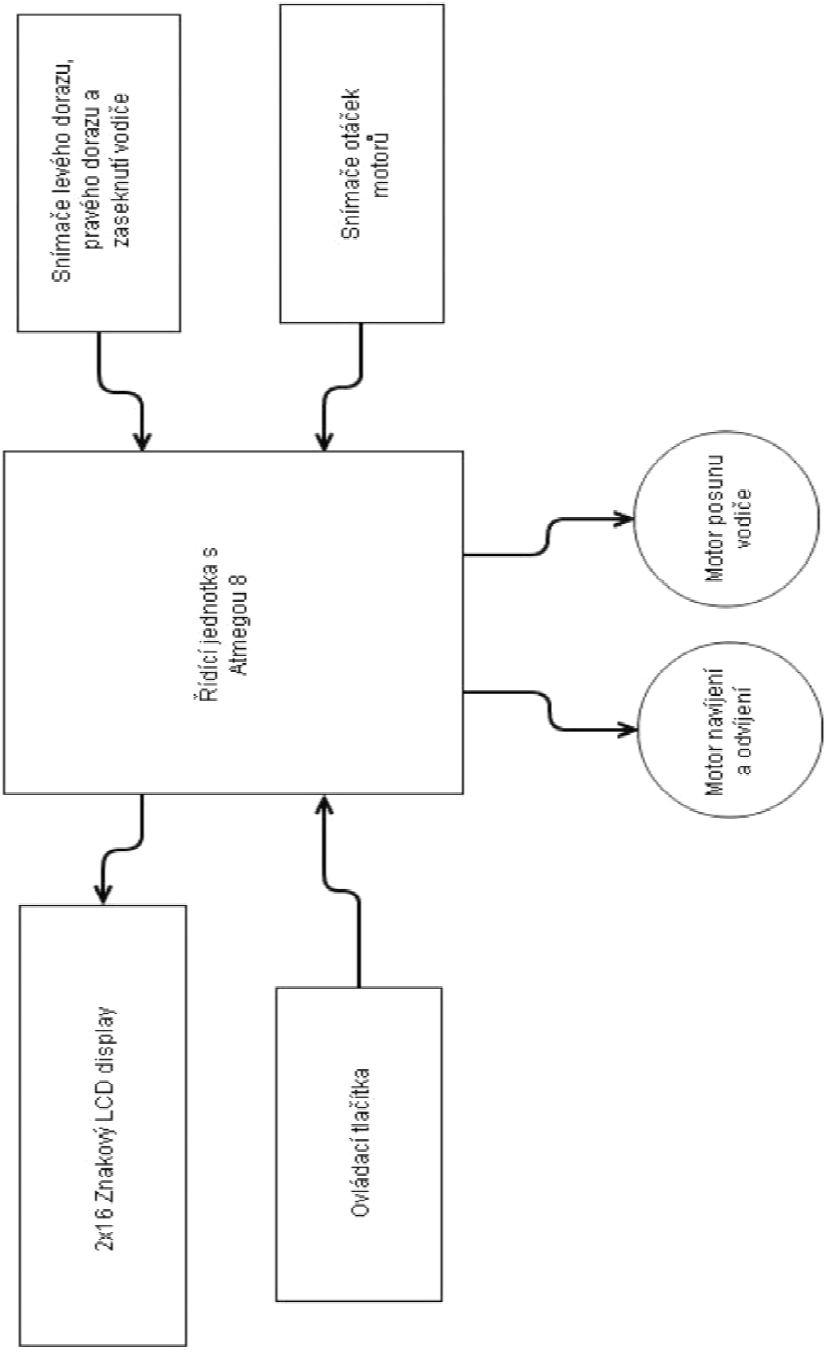
Nejsložitější jsou linky, které umí navíjet i toroidní cívky. Tyto systémy jsou velmi složité, rozměrné a nákladné. Řízení těchto navíjecích automatů bude obstarávat PLC automat. Tyto navíjecí automaty již dovedou navíjet téměř jakékoliv cívky.

3. Návrh digitálního řízení

Od mnou realizované navíječky očekávám, že bude umět navíjet a odvíjet cívky zcela automaticky. Uživatel si nejprve vybere, zda chce cívku navíjet nebo odvíjet. Následně vybere počet závitů. Pokud vybral navíjení, musí ještě zadat délku cívky, průřez vodiče a nastavit kladku na začátek cívky. Dále by navíječka měla zobrazovat počet navinutých závitů a počet závitů, které zbývají navinout.
Celá navíječka je řízena jednočipovým mikropočítačem od firmy Atmel. Vybral jsem si čip z řady Atmega, z důvodu dostatečného výkonu a dostatečného počtu vstupně výstupních linek. Konkrétně jsem si vybral mikropočítač Atmega8, je to nejmenší zástupce řady Atmega, ale na tuto aplikaci dostačuje.
Jako budiče jsem si vybral integrované obvody L293B, což jsou dva výkonové H-můstky pro buzení motorů v jednom pouzdře.
Pro zobrazeníi je použit LCD display 2x16 znaků MC1602E.

Navíječka se ovládá pomocí čtyř mikrospínačů (nahoru, dolů, potvrzení, zpět/zrušení)

Pro snímání otáček může být použita optická brána, jedna na cívce, na kterou se navíjí a druhá na cívce, ze které je odvíjeno. Dále je připojen spínač na kladku, přes kterou jde vodič, pokud se vodič zasekne, kladka rozepne kontakt a zastaví navíjení.



OBR 1- Blokové schéma

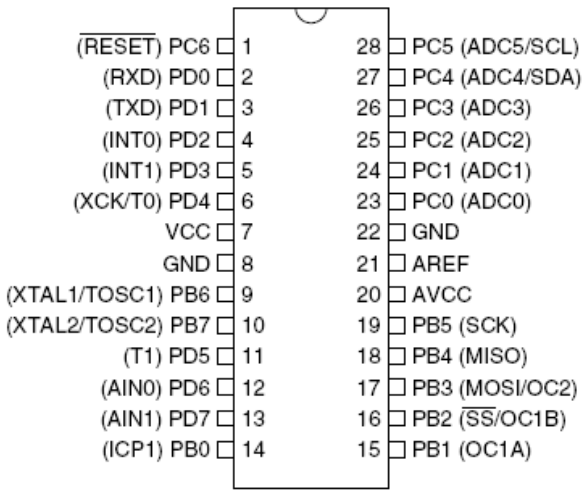
3.1. Použité součástky

3.1.1. Atmega 8

Atmega8 je nejmenší zástupce jednočipových mikropočítačů řady Mega firmy Atmel. Je to 8-bitový mikropočítač, má 8kB paměti flash pro program, 1kB paměti RAM a 512B paměti EEPROM. Je to RISC-ový mikropočítač, to znamená, že má redukovanou sadu instrukcí (má jich 130). Může být taktována až na 16MHz. Obsahuje 23 vstupně výstupních linek a většina z těchto linek může mít i specifické využití. Dále obsahuje:

- 2 x 8-bit časovač se samostatnou předděličkou
- 1 x 16-bit časovač se samostatnou předděličkou
- 2 x vstup externího přerušení
- 3 x PWM (Pulse with modulation) kanály
- 6 x A/D (analog to digital) převodník
- analogový komparátor
- programovatelnou sériovou linku (USART)
- vnitřní oscilátor 1; 2; 4; 8; MHz

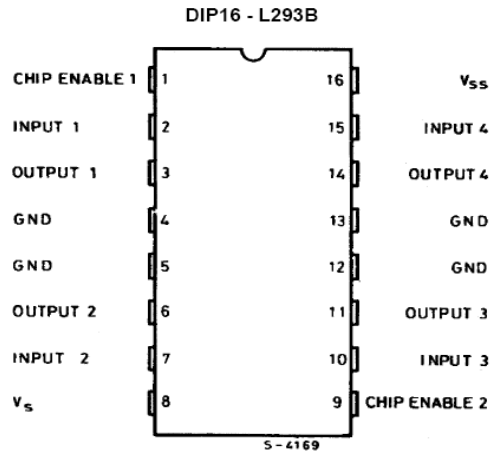
a další….



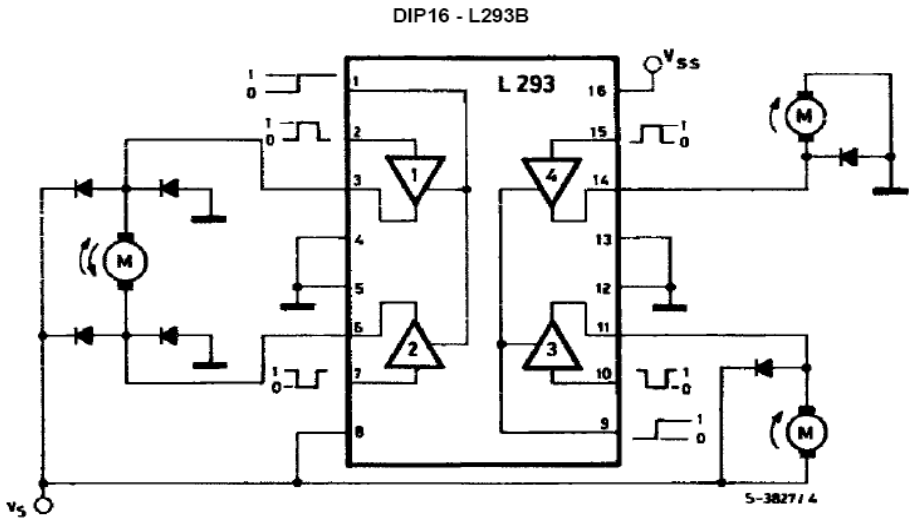
OBR 2 - Atmega8 rozložení pinů

3.1.2. L293B

L293B je budič krokových motorů. Obsahuje dvojici výkonových H-můstků, podle katalogu tyto H-můstky mohou dát s chladičem proud až 1A na kanál a dokáže napájet motory až do 36V. U obou H-můstků lze povolit nebo zakázat jejich činnost vstupem CE (Chip enable). Pokud přivedeme na vstup Input1 LOG1 na výstupu Output1 Budič přivede napětí, které je připojeno na vstup V_S. Pokud je na vstupu Input1 LOG0, je i na výstupu Output1 nulové napětí. Toto platí analogicky i pro ostatní vstupy a výstupy.



OBR 3 - L293B rozložení pinů



OBR 4 - L293B vnitřní blokové schéma

3.1.3. MC1602E

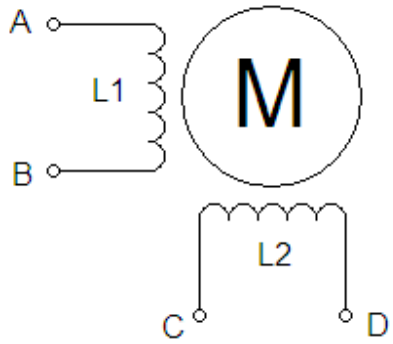
MC1602E je znakový LCD display. Má dva řádky po 16 znacích. Je modře podsvícen pomocí LED. Display je řízen řadičem S6A0069 a každý znak je zobrazen na poli 5x8 bodů. Příkazy jsou posílány paralelně. Display umožňuje buď osmibitovou komunikaci, nebo čtyřbitovou komunikaci.

OBR 5 - MC1602E sada znaků

3.1.4. Krokové motory

Jako krokové motory byly použity bipolární krokové motory z tiskárny, které mi věnoval spolužák. Jeden krokový motor má 47 kroků na otáčku a větší sílu, proto byl použit na otáčení cívkou při navíjení, jelikož není potřeba takové přesnosti a je potřeba větší síly. Druhý krokový motor má 97 kroků na otáčku a menší sílu. Tento krokový motor byl použit na posun kladky, která vede vodič při navíjení, aby byl na výsledné cívce závit vedle závitu, zde je potřeba větší přesnosti a není potřeba velké síly.

Bipolární krokové motory nemají vyvedený střed každé cívky, takže se musí ovládat tak, že u každé cívky přepínáme polaritu napětí. Což je žádá složitější budič – většinou se používá plný H-můstek na každou cívku.



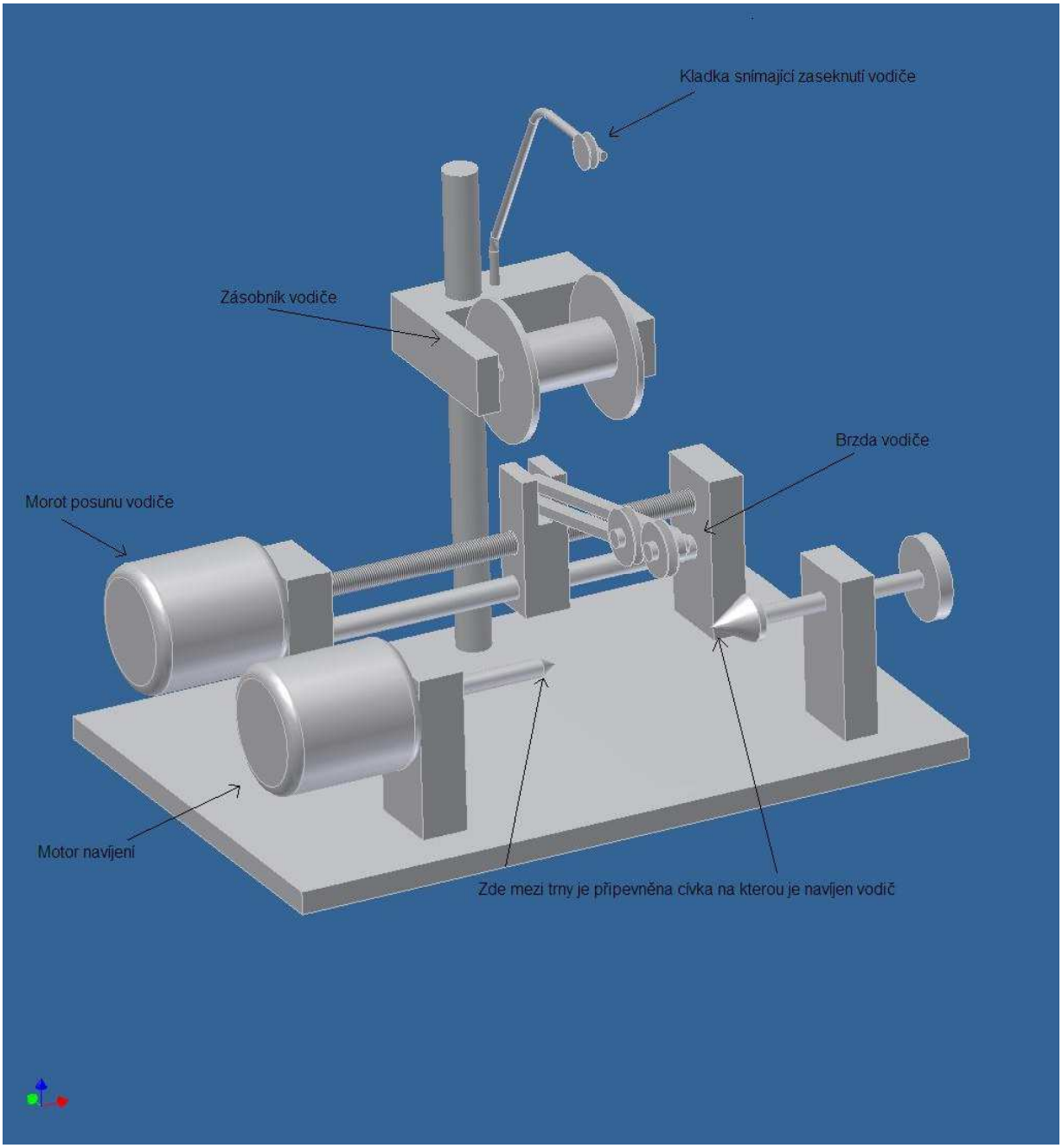
OBR 6 - Vnitřní zapojení bipolárního krokového motoru

A	B	C	D
1	0	1	0
0	1	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1 – přivedeno napájecí napětí 0 – přivedeno nulové napětí			

OBR 7 - Řízení bipolárního motoru dvoufázově s plným krokem

4. Návrh mechanické konstrukce navíječky

Bohužel úprava původní kostry by byla příliš složitá a bylo by jednodušší vyrobit novou, tak vznikl tento návrh, jak by navíječka mohla vypadat.



OBR 8 - Nákres možného mechanického řešení navíječky

5. Dokumentace

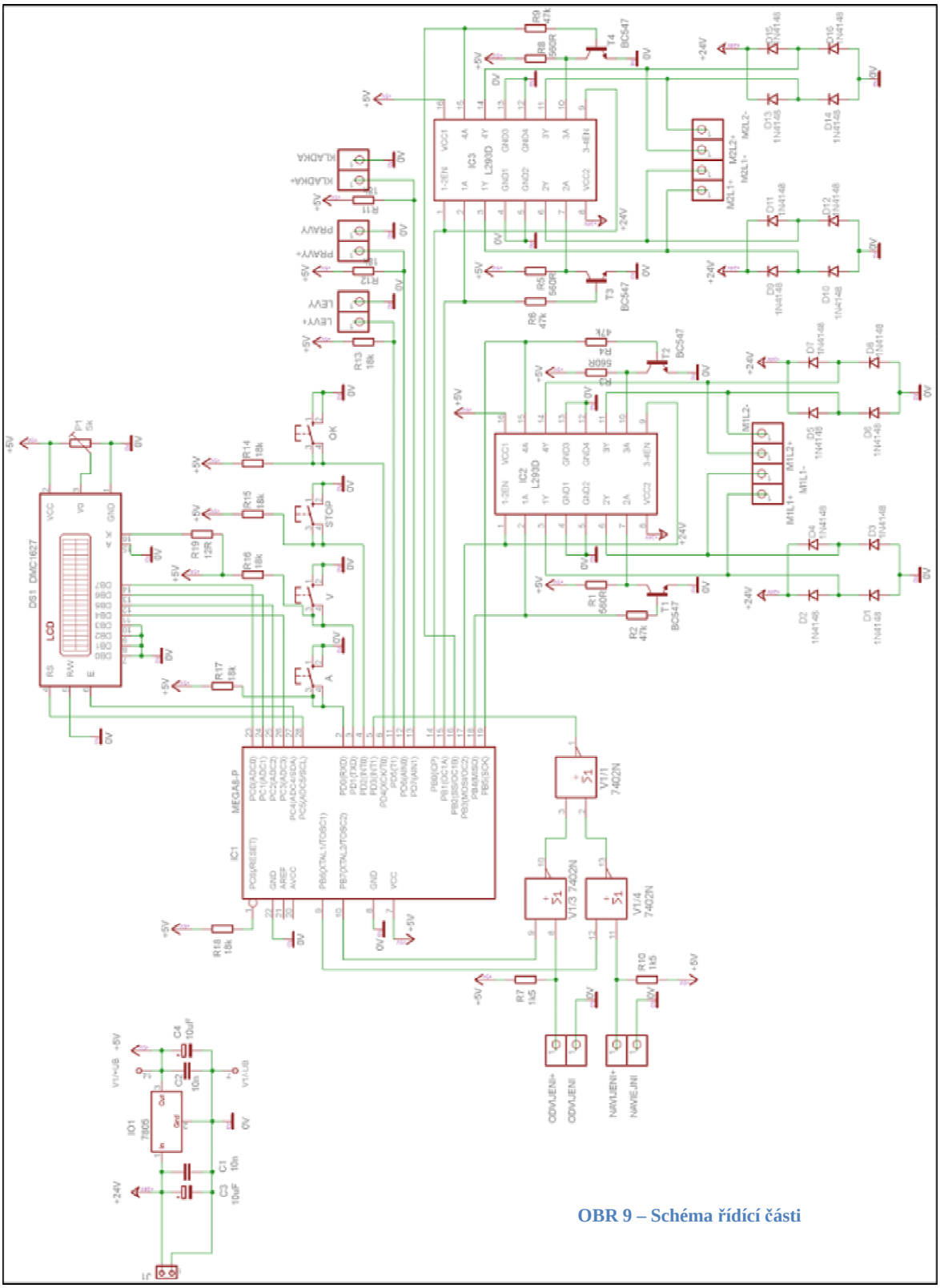
5.1. Elektronická část

Schéma a plošný spoj jsem navrhl v programu Eagle. Na obrázku níže je celé schéma digitálního řízení navíječky. Celá navíječka je řízena jednočipovým mikropočítačem Atmega8. V Atmeze8 je pro časování využíván vnitřní oscilátor, ten sice není tak přesný, jako kdyby byl použit externí krystalový oscilátor, ale v tomto zapojení není nic závislé na přesném časování. Tím jsou ušetřeny dva piny.

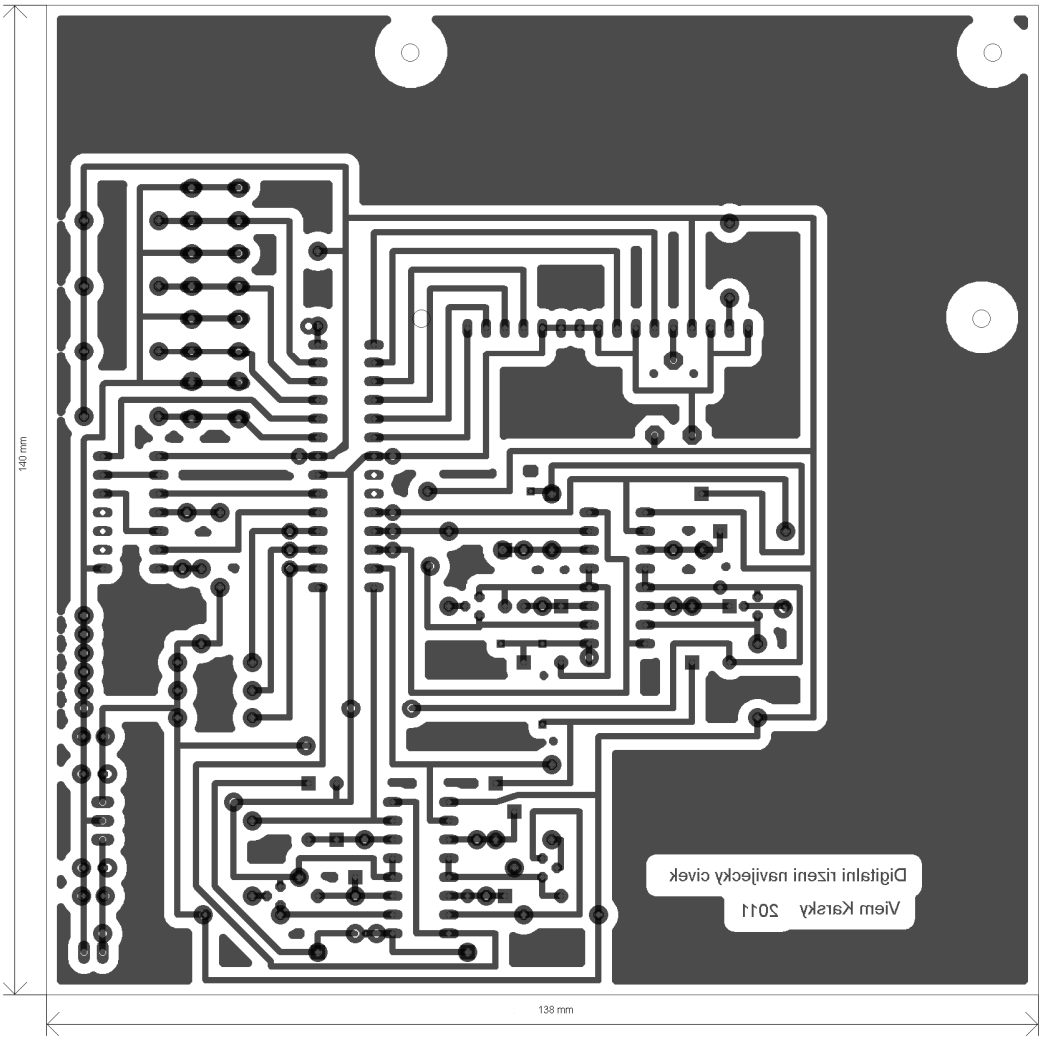
O zobrazení údajů se stará LCD display, který je připojený čtyř vodičově na portC. PortC atmegy8 je pouze 7 pinový (ostatní porty jsou 8 pinové). Z toho je jeden pin využíván jako reset při programování, tudíž zbývá 6 linek, které display obsadí všechny. Trimrem P1 nastavíme optimální jas display.

Ovládací tlačítka jsou připojena proti zemi a linky, na kterých jsou tlačítka připojena, jsou drženy v LOG1 pomocí pullup rezistorů o hodnotě 18k. V katalogu je udáváno, že maximální vstupní proud I_{iH} je menší než 1uA, to znamená, že na rezistoru vzniká úbytek napětí menší než 18mV, takže LOG1 na vstupu je zaručena. Při stisku tlačítka dojde k uzemnění vstupu, a jelikož je odpor sepnutého spínače zanedbatelný vůči upínacímu rezistoru, je na vstupu LOG0. Tlačítko stop by mělo být realizováno rozpínacím kontaktem, ale sehnal jsem mikrospínače pouze se spínacím kontaktem, tak je i na stop spínač použit. Veškeré vstupy od snímačů jsou ošetřeny stejným způsobem, aby pokud není připojen snímač, nedocházelo k hazardním stavům a byla na vstupu jasně definovaná úroveň. Celý port D je nastaven jako vstupní, jelikož jsou na něj připojena ovládací tlačítka a senzory. Konkrétně tlačítko „nahoru“ je na PD0, tlačítko „dolů“ je na PD1, tlačítko „potvrzení“ je na PD4 a tlačítko „stop“ je na PD2 což je jeden ze dvou vstupů přerušení. Vstup přerušení, který je PD2 je INT0 a má vyšší prioritu než přerušení INT1 na vstupu PD3. Na vstup PD3 je připojeno přerušení od optické závory, snímající otáčku. Dále jsou na port D připojeny koncové spínače u konzole pro posun drátu při navíjení a spínač, indikující zaseknutí vodiče. Levý doraz je na PD5, pravý doraz je na PD6 a spínač zaseknutého vodiče je na PD7.

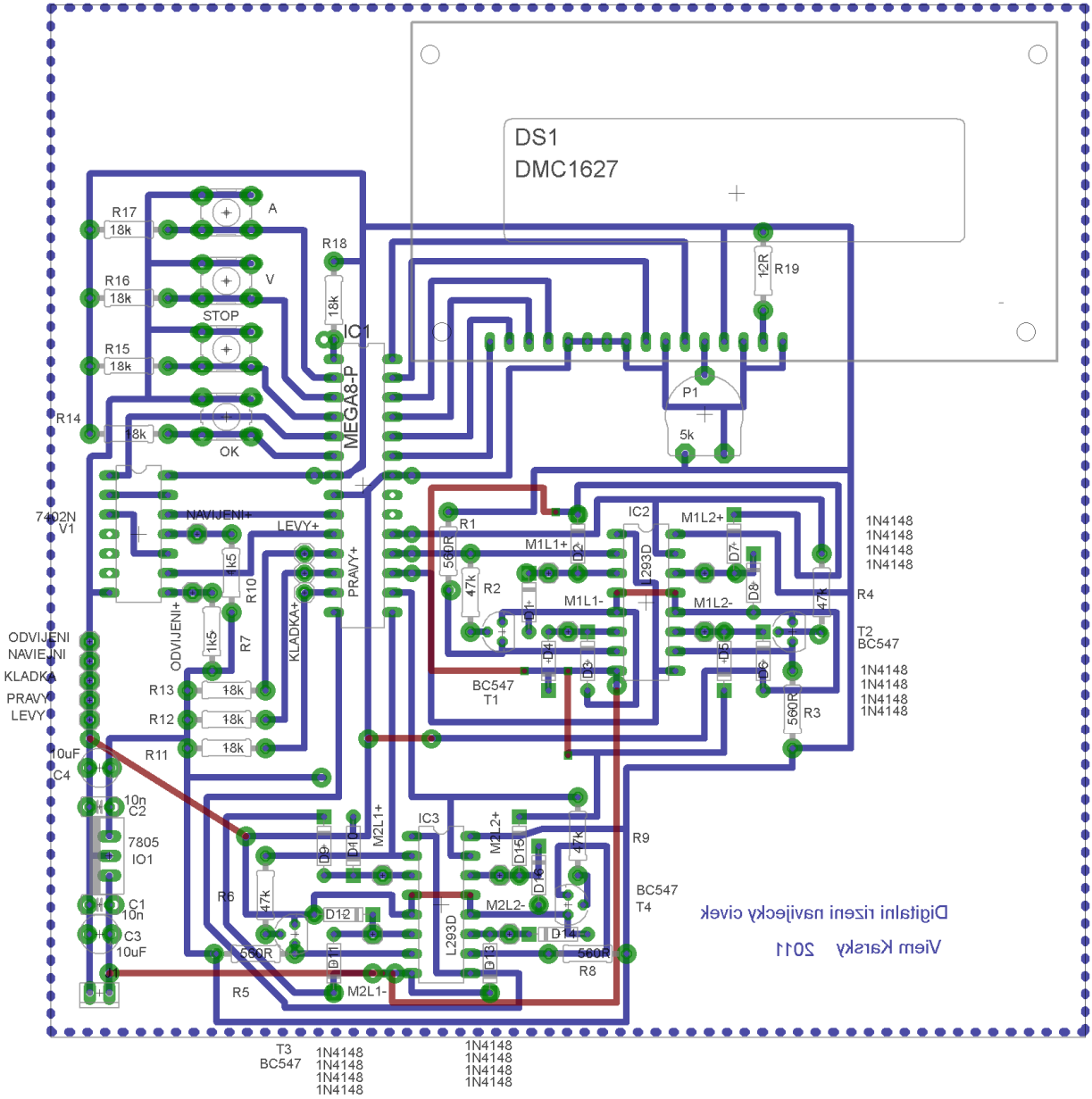
Port B je nastaven jako výstupní, protože jsou k němu připojeny budiče krokových motorů (L293B) a ovládání, od které optické brány je bráno přerušení.



OBR 9 – Schéma řídící části



OBR 10 - Předloha pro desku plošných spojů



OBR 11 - Osazovací schéma

5.1.1. Dekodér přerušení

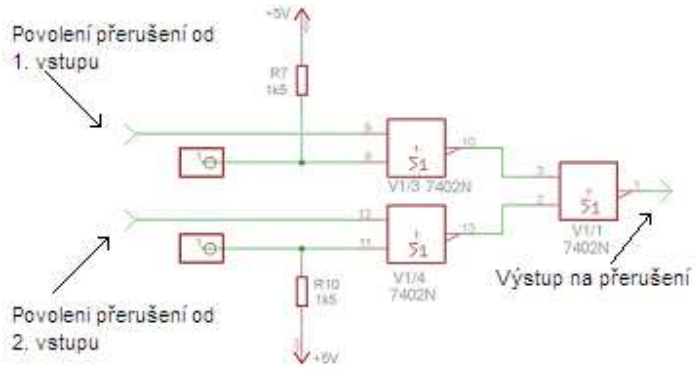
Při navíjení je cívka, na kterou je navíjen vodič umístěna na hřídeli, na kterou je připojen krokový motor a na druhé hřídeli, které je volná, je připevněn zásobník vodiče. Otáčky tedy snímáme na cívce, ke které je připojen motor. Při odvíjení je cívka připevněna na místě pro zásobník vodiče a vodič z ní je odvíjen na cívku, která je na hřídeli s motorem. Takže při odvíjení musíme snímat otáčky na cívce, která je na místě zásobníku vodiče. Z toho plyne, že potřebujeme dva vstupy do jednočipového mikropočítače. Z důvodu snadnější programové obsluhy jsem chtěl snímače otáček připojit na vstupy přerušení Atmegy8. Bohužel Atmega8 má vstupy přerušení pouze dva a na jeden je již připojeno stop tlačítko, takže pro snímače otáček bylo nutné vymyslet dekodér přerušení.

K vytvoření dekodéru přerušení jsem použil 3 hradla logického členu NOR.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

OBR 12 - Pravdivostní tabulka NOR

Z tabulky je vidět, že pokud je alespoň na jednom vstupu hradla NOR LOG1, tak je na výstupu vždy LOG0 a této vlastnosti hradla NOR je využito v dekodéru přerušení.



OBR 13 - Schéma dekodéru přerušení

Podle schématu vidíme, že pokud je na vstupu „povolení přerušení“ LOG1, je na výstupu hradla vždy LOG0, ať je na druhém vstupu jakákoli logická úroveň. Takže LOG1 na vstupu „povolení přerušení“ nám blokuje vstup přerušení. Pokud je na obou vstupech „povolení přerušení“ LOG1, je na obou výstupech hradel (a zároveň na vstupech třetího hradla) LOG0, tudíž je na výstupu LOG1.

Pokud přivedeme na vstup „povolení přerušení od 1. vstupu“ LOG0 a na vstup „povolení přerušení od 2. vstupu“ LOG1, je signál z 1. vstupu negován 1. hradlem NOR. Na výstupu druhého hradla je stále LOG0, takže tento vstup neovlivní výstup třetího hradla. A třetí hradlo znovu neguje signál z prvního hradla, to znamená, že se signál dostal z 1. vstupu na výstup. Toto platí analogicky i pro druhý vstup. Pokud bude na obou vstupech povolení přerušení LOG0, bude na výstupu 3. hradla logický součin obou vstupů přerušení. A to proto, že oba vstupy přerušení jsou hradly NOR napřed negovány, pak třetím hradlem NOR logicky sečteny a opět negovány, což odpovídá De Morganovým zákonům a Booleově algebře. De morganovy zákony říkají: $\overline{\bar{A} + \bar{B}} = \overline{\overline{A \cdot B}}$. A podle zákonů Booleovy algebry, konkrétně podle zákonu dvojité negace, platí, že $\overline{\overline{A \cdot B}} = A \cdot B$. Z toho vyplívá, že $\overline{\bar{A} + \bar{B}} = A \cdot B$.

Pov. Př. 1	Pov. Př. 2	Vstup 1	Vstup 2	Výstup	Popis
1	1	X	X	1	Oba vstupy povolení = 1 => výstup = 1
0	1	0	X	0	Povolení 1 = 0 a Povolení 2 = 1 => výstup = vstup 1
0	1	1	X	1	
1	0	X	0	0	Povolení 1 = 1 a Povolení 2 = 0 => výstup = vstup 2
1	0	X	1	1	
0	0	0	X	0	Oba vstupy povolení = 0 => výstup je logický součin obou vstupů (nevhodné pro dekodér přerušení – tento stav by neměl nikdy nastat)
0	0	X	0	0	
0	0	1	1	1	
„X“ - znamená, že logická úroveň tohoto vstupu neovlivní výstup					

OBR 14 - Pravdivostní tabulka dekodéru přerušení

5.1.2. Buzení krokových motorů

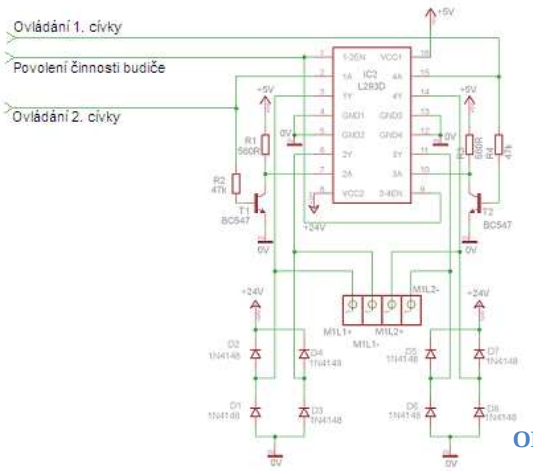
Jak bylo napsáno výše, pro buzení krokových motorů používám H-můstkový budič L293B. V jednom pouzdře jsou dva celé H-můstky. Jako krokové motory používám bipolární krokové motory, které mají dvě cívky. Na každou cívku je potřeba jeden H-můstek. Takže na každý motor je potřeba jeden integrovaný obvod. Jak jsem psal výše, k řízení jednoho H-můstku jsou potřeba dohromady tři signály (Povolení činnosti H-můstku, sepnutí jedné poloviny H-můstky, sepnutí druhé poloviny H-můstku). Toto se dá zjednodušit, když si uvědomíme, že stačí, když cívku připojíme jednou stranou na plné napájecí napětí a druhou na nulové napětí a poté polaritu otočíme, nebudeme využívat stavu, kdy jednou cívkou neteče proud, takže stále oběma cívkami teče proud. Takovéto řízení se nazývá dvoufázové řízení s plným krokem. Když využijeme tohoto řízení, můžeme vždy na jeden vstup H-můstku přivést signál a zároveň tento signál negovat a přivést jej na druhý vstup. Tím budeme moci každý H-můstek ovládat pouze jedním signálem místo dvou signálů. To nám umožní řídit dva krokové motory čtyřmi vodiči místo osmi, sice tím přijdeme o

možnost řízení motorů s polovičním krokem a tím zvýšit počet kroků na otáčku na dvojnásobek, ale v této aplikaci to není nutné.

Negovat řídící signály jde v podstatě dvěma způsoby buď pomocí dalšího integrovaného obvodu anebo pomocí tranzistoru v zapojení se společným emitorem ve spínacím režimu. V této aplikaci jsem využil tranzistoru a to z toho důvodu, že jsem nechtěl použit další integrovaný obvod a použití tranzistorů zjednoduší návrh plošného spoje.

K negování signálů byl použit tranzistor BC547 h_{21e} katalog udává 250 – 400 výpočet jsem prováděl s průměrnou hodnotou 325. Rezistor do kolektoru R_c jsem zvolil 560R. Na rezistoru o hodnotě 560R vzniká úbytek napětí 5V, pokud ním prochází proud 8,9mA. Zanedbávám zde úbytek napětí na přechodu CE tranzistoru, protože potřebuji, aby se otevřel naplno a v hluboké saturaci může být úbytek na přechodu CE i 0,2V, pak by mohl být úbytek napětí na rezistoru maximálně 4,8V. Když budu počítat s proudem, který odpovídá úbytku napětí 5V na rezistoru o hodnotě 560R, mám jistotu, že se tranzistor určitě dostatečně otevře a navíc, pokud potřebuji tranzistor ve spínacím režimu, volí se proud do báze 3x – 5x větší, než je nutné na otevření tranzistoru. Takže I_c = 8,9mA. Proud do báze tranzistoru se vypočítá podle vztahu —— ——— . Tranzistor je spínán z výstupu Atmegy8. Katalog udává, že atmega poskytuje v LOG1 4,2V. Úbytek na přechodu BE je 0,6V úbytek na rezistoru R_b = 4,2-0,6 = 3,6V. Rezistor má hodnotu —— —— . Třetinový odpor má hodnotu 44,3k. Nejbližší odpor co jsem měl, byl 47k.

Pokud hodnotu rezistoru dosadím do výpočtu, vyjde I_b=76,6µA. I_c počítaný pro nejhorší případ (nejmenší h_{21e}) je . 19,15mA > 8,9mA tudíž je zaručeno, že se tranzistor vždy otevře.



OBR 15 - Zapojení budiče krokového motoru

Jak bylo vysvětleno výše, při dvoufázovém ovládání krokových motorů s plným krokem se vždy mění stav pouze jedné cívky a na ovládání jedné cívky je potřeba dvou vodičů. Dále jsem vysvětlil, že je možné každou cívku ovládat jedním vodičem a celý krokový motor dvěma vodiči. Kombinace stavů dvou vodičů jsou celkem čtyři, přičemž se mění, při každém kroku, signál vždy jen na jednom vodiči jedná se o Grayův kód. Další tabulka ukazuje jednotlivé stavy vodičů a jejich posloupnost, aby se krokový motor roztočil.

Číslo kroku	Ovládání 1. cívky	Ovládání 2. cívky
1.	0	0
2.	0	1
3.	1	1
4.	1	0
5.	0	0
1 – k cívce je připojeno napájecí napětí 0 – k cívce není připojeno napájecí napětí		

OBR 16 - Jednotlivé stavy vodičů a jejich posloupnost, aby se motor roztočil

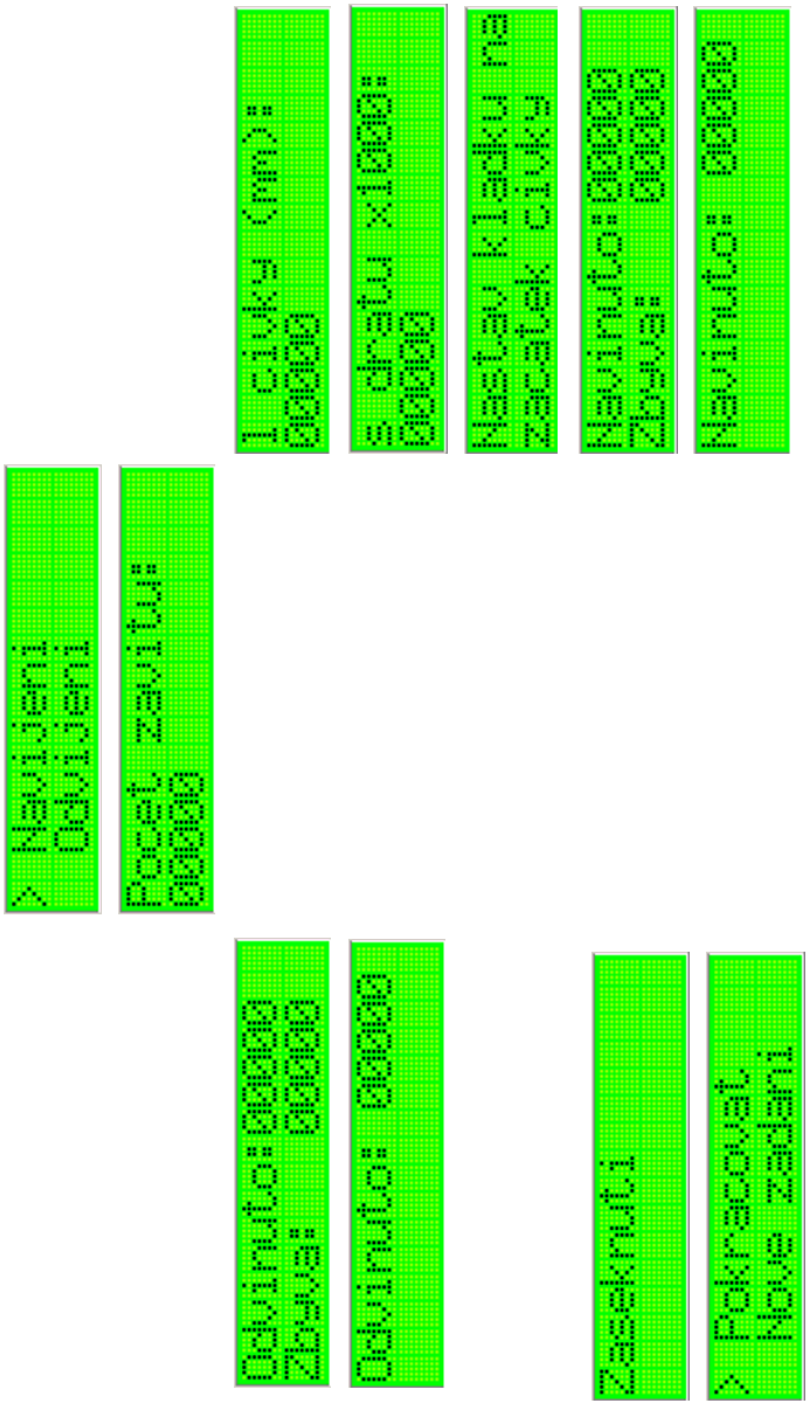
5.2. Programové řešení

Program do jednočipového mikropočítače byl napsán a přeložen ve vývojovém prostředí Bascom-AVR 2.0.5.0. Bascom-AVR je klon jazyka BASIC.

Po zapnutí navíječky a zobrazení úvodní obrazovky, je uživatel vyzván k zadání zda, si přeje cívku navíjet nebo odvíjet. Pak je vyzván k zadání počtu závitů. Pokud bylo zadáno odvíjení, objeví se počítadlo s počtem odvinutých závitů a počtem zbývajících závitů. Pokud bylo zadáno navíjení, objeví se výzva k zadání průřezu vodiče, poté výzva k zadání délky cívky. Následně musí uživatel nastavit kladku na začátek cívky, odkud si přeje, aby byla cívka navíjena. A na konec se také zobrazí počítadlo, ukazující počet navinutých a počet zbývajících závitů. Pokud se během navíjení, nebo odvíjení zasekne vodič, okamžitě se zastaví motory a na display se objeví hláška „Zaseknutí“. Uživatel zde může buď tlačítkem „stop“ navíjení přerušit a vrátit se k úvodnímu menu, nebo odstranit příčinu zaseknutí vodiče a tlačítkem „potvrzení“ pokračovat v navíjení nebo odvíjení. Pokud je během navíjení nebo odvíjení stisknuto tlačítko „stop“, opět se celá navíječka zastaví a na display se zobrazí menu, kde si uživatel může vybrat, jestli si přeje zrušit navíjení nebo odvíjeni a skočit do úvodního menu anebo jestli si přeje v navíjení nebo odvíjení pokračovat.

5.2.1. Průběh menu

Na následujícím obrázku je průběh jednotlivých obrazovek digitálního řízení navíječky cívek.



OBR 17 - Průběh menu

5.3. Ukázky části programu

Jako první ukázku části programu jsem vybral menu, protože to mi dalo asi největší práci jej vymyslet. Těchto menu je v celém programu více, momentálně popíši to první, takto jednoduché menu, kde se vybírá pouze ze dvou položek, by se dalo realizovat i mnohem jednodušeji, ale změnou několika konstant může mít menu téměř neomezené množství položek (počet položek omezuje pouze velikost paměti jednočipového mikropočítače).

5.3.1. Menu

Hned na začátku úseku programu se do proměnné X se nastaví hodnota 1, to z důvodu, že zde začíná program a proměnná má hodnotu 0, jakmile program dalším příkazem skočí dále, kde je „přepínač“ case, ten nemá definováno jak se zachovat, pokud proměnná má hodnotu 0 a program by se mohl chovat neočekávaně. Dále program skočí příkazem „Goto“ na návěští „Menu“, kde narazí na příkaz „Select Case X“ tento příkaz vezme hodnotu uloženou v proměnné X a podle ní vykoná činnost. Zde se proměnná X rovná 1, proto se provedou příkazy v bloku za Case 1. Vymaže se display a vypíše se na první řádek „> Navíjení“ a na druhý řádek se vypíše „Odvíjení“. Příkazy za „Case 2“ jsou ignorovány.
Následně program skočí na návěští „Zacatek“. Zde je nekonečná smyčka, která testuje, jestli bylo stisknuto tlačítko „Dolu“, „Nahoru“ nebo „Entr“. Jak jsem psal výše, pokud je stisknuto tlačítko, objeví se na vstupu LOG0, proto mám v podmínce testující stisk tlačítka výraz „alias pinu“ = 0. Pokud je stisknuto tlačítko „Dolu“, inkrementuje se proměnná X, takže se její obsah rovná 2. Další podmínkou je otestováno, jestli hodnota proměnné X nepřekročila rozsah a pokud ano, je nastavena na počáteční hodnotu (X=1). Následně program opět skočí na návěští „Menu“ a jelikož proměnná X má hodnotu 2, vykoná se blok příkazů následující za „Case 2“. Opět se vymaže display, na první řádek se tentokráte vypíše „> Odvíjení“ a na druhý řádek se vypíše „Navíjení“, poté program opět skočí na návěští „Zacatek“, kde dále testuje tlačítka.
Pokud opět stiskneme tlačítko „Dolu“ je proměnná X opět inkrementována a hodnota jejího obsahu je 3. Následující podmínka vyhodnotí, že je tato proměnná větší než dva a tak ji nastaví opět na 1. Poté program opět skočí na návěští menu a příkaz „Select Case“ provede příkazy za „Case 1“ a vrátí se zpět na návěští „Zacatek“.
Pokud zde stiskneme tlačítko „Nahoru“ je proměnná X dekrementována, takže má hodnotu 0, to vyhodnotí opět další podmínka, že hodnota X je menší než 1 a nastaví hodnotu X na nejvyšší hodnotu (tady u toho menu na 2) a opět skočí na návěští „Menu“. Jelikož X = 2 provede se blok příkazů za „Case 2“, poté opět skočí na návěští „Zacatek“.
Po opětovném stisku tlačítka „Nahoru“ se opět proměnná X dekrementuje, tudíž X = 1. A opět program skočí na návěští „Menu“ provede příkazy za „Case 1“ a skočí zpět.
Po stisku tlačítka „Entr“ program skočí na návěští „Vyber“ a podle hodnoty X skočí na další

návěští. Zde pokud je hodnota X = 1, skočí na návěští „Zadani_navijeni“ a pokud je hodnota X = 2, skočí na návěští „Zadani_Odvijeni“.

Zde je výpis popisované části programu:

```
'-----menu-----
X = 1
Goto Menu

Zacatek:
Do
If Dolu = 0 Then
    X = X + 1
        If X > 2 Then
        X = 1
        End If
    Goto Menu
End If

If Nahoru = 0 Then
    X = X – 1
        If X <1 Then
        X = 2
        End If
    Goto Menu
End If

If Entr = 0 Then
Goto Vyber
End If
Loop

Menu:
Select Case X
        Case 1
            Cls
            Locate 1 , 1
            Lcd "> Navijeni"
            Locate 2 , 3
            Lcd "Odvijeni"
        Case 2
            Cls
            Locate 1 , 1
            Lcd "> Odvijeni"
            Locate 2 , 3
            Lcd "Navijeni"
End Select
Goto Zacatek

Vyber:
If X = 1 Then
Goto Zadani_navijeni
End If

If X = 2 Then
Goto Zadani_odvijeni
End If
```

5.3.2. Ovládání krokových motorů

Další část programu je způsob, jakým jsem řešil ovládání krokových motorů. Jak jsem popsal výše, krokový motor ovládám dvěma vodiči, ale bohužel nejsou ovládány obyčejným binárním kódem ale Grayovým kódem. Zde vidíte, jakým způsobem jsem tento problém řešil já, toto řešení ovšem není vhodné, pokud by bylo nutné generovat Grayův kód pro více bitů, protože s každým bitem se zdvojnásobí počet možných stavů.
Tato část programu běží neustále v cyklu, dokud není splněna podmínka, že počet navinutý závitů je roven počtu závitů zadaných. Jakmile je tato podmínka splněna cyklus se dále neprovádí a program pokračuje příkazy za tímto cyklem. Počet navinutých závitů se počítá pomocí přerušení v jiné části programu, zde se pouze vyhodnocuje podmínka.
Před začátkem cyklu je vynulován obsah proměnné Y, která slouží uvnitř cyklu k počítání, který krok se má provést. Pokud vynulování obsahu proměnné před začátkem cyklu neprovedeme a budeme proměnou používat i na jiném místě v programu, může v některých cyklech neočekávaná hodnota proměnné způsobit špatnou funkci programu. V tomto cyklu je sice hodnota proměnné ošetřena podmínkami na začátku cyklu, ale obecně je lepší vynulování provést.
První příkaz uvnitř cyklu „Povol_nav = 0“ způsobí, že se objeví LOG0 na pinu, kde je připojen dekodér přerušení a tím povolí počítání otáček při navíjení. Další příkaz „Mot1_e = 1“ nastaví LOG1 na pin, na který jsou připojeny povolovací vstupy H-můstkového budiče motoru, který otáčí cívkou. Pokud na tento pin nenastavíme LOG1, motor se nebude točit, protože budič na něj nebude přivádět napětí.
Další příkaz inkrementuje obsah proměnné Y, protože na začátku se obsah proměnné rovnal nule, je teď hodnota obsahu rovna jedné. Následující podmínka kontroluje, jestli hodnota proměnné není větší než počet kroků, v tomto případě jestli není hodnota proměnné větší než čtyři. Pokud by byla hodnota proměnné větší než čtyři, byla by změněna na jedna. Další podmínka kontroluje, jestli hodnota proměnné není menší než 1, pokud ano, je změněna její hodnota na 1.
Následující příkaz je „Select Case Y“, který provede příkazy podle hodnoty Y. Nyní, když je hodnota Y = 1, provedou se příkazy následující za „Case 1“. Těmito příkazy jsou nastaveny výstupní linky, které ovládají budič krokového motoru, do LOG0. Další příkaz, který se provede je příkaz „Waitms 35“, ten způsobí, že program 35ms čeká. Čekání je nutné, jelikož Atmega8 by jinak měnila hodnotu na výstupech příliš rychle a krokový motor by nestačil reagovat.
Dále program na příkaz Wend, ten mu říká, že má opět skočit na začátek cyklu. Opět zkontroluje, zdali není splněna podmínka a pokud není, dále pokračuje tělem cyklu. Opět nastaví Povol_nav = 0 a Mot_e1=1, pokud stav pinů, kterým odpovídají tyto aliasy, nebyl změně, nic se nezmění. Dále inkrementuje hodnotu proměnné Y (hodnota Y by se měla rovna dvěma), která je porovnána podmínkami, jestli vyhovuje rozsahu. Příkaz „Select Case Y“ tentokrát vykoná část za „Case 2“. Tato část nastaví pátý pin portu B Atmegy8 do LOG0 a čtvrtý pin portu B Atmegy8 do LOG1.
Opět proběhne čekání, skok na začátek cyklu, porovnání, jestli není splněna podmínka a pokud ne, je opět inkrementována hodnota proměnné Y. Takže hodnota proměnné Y je 3. Příkaz „Select Case

Y“ provede příkazy za „Case 3“. Tyto příkazy nastaví piny čtyři a pět portu B Atmegy8 oba do LOG1.
Dále to jde stejně a hodnota proměnné Y je 4, příkaz „Select Case Y“ provede příkazy tentokrát za „Case 4“. Zde je pin pět portu B Atmegy8 nastaven do LOG1 a pin čtyři portu B Atmegy8 do LOG0.
Při dalším průběhu, je opět hodnota proměnné Y inkrementována a Y má hodnotu 5. Protože 5 je větší než 4, vyhodnotí to první podmínka uvnitř cyklu a nastaví hodnotu proměnné Y na 1. Dále již je to stejné jako u prvního průběhu cyklem. Tento cyklus se neustále opakuje, dokud není dosažen cílový počet závitů.
Dále je uvedena popisovaná část programu:

```
Y = 0
While Navinute_zavity = Zadany_pocet_zavitu
    Povol_nav = 0
    Mot1_e = 1
    Incr Y
        If Y > 4 Then
            Y = 1
        End If
        If Y < 1 Then
            Y = 1
        End If
    Select Case Y
        Case 1
        Portb.5 = 0
        Portb.4 = 0
        Case 2
        Portb.5 = 0
        Portb.4 = 1
        Case 3
        Portb.5 = 1
        Portb.4 = 1
        Case 4
        Portb.5 = 1
        Portb.4 = 0
    End Select
    Waitms 35
Wend
```

6. Ověření funkcí řízení navíječky

Veškeré funkce byly ručně otestovány na testovacím přípravku. Jako koncové spínače a snímač zaseknutí vodiče byly použity přepínače a jako senzory otáček byly použity tlačítka. Na hřídele motorů byly připevněny terče, aby bylo dobře patrné, zda se motor otáčí, případně na kterou stranu. Veškeré funkce fungují tak jak bylo stanoveno.

7. Fotografie zařízení

OBR 18 - Řídící část

OBR 19 - Tlačítka

OBR 20 - Pohled na celé zařízení

8. Resumé

U navíječky se mi povedlo úspěšně realizovat veškeré funkce, které byly stanoveny. Navíječka cívek umožňuje jak navíjení, tak odvíjení cívek do maximálního počtu závitů 99 999. Při odvíjení uživatel zadá pouze počet závitů, které mají být odvinuty. Při navíjení uživatel zadá počet závitů, které mají být navinuty, průřez vodiče, délku cívky a nastaví kladku posunu vodiče na začátek cívky.
Při zkoušení vše funguje, jak bylo zadáno, pouze při použití ve skutečné navíječce by bylo dobré vyměnit budiče krokových motorů za výkonnější, aby mohly být použity silnější krokové motory.

I realized all funcions wich was assign. Coil winder allows winding the coils and unwinding the coils. In unwinding coils user enter a nubmber of coils, wich he wants to unwide. In windingcoils user enter a number of coils, cross section of cabel, lenght of coil and set roller to start of coil.
All functions works as was assign, only if it uses in real coil winder, it will be good to replace driver of stepper motor with a more powerful driver and use more powerful stepper motor.

9. Seznam obrázků

10. Použitá literatura

1. Katalogové listy Atmega8 - http://www.atmel.com/Images/doc2486.pdf
2. Katalogové listy L293B - http://www.gme.cz/_dokumentace/dokumenty/399/399-010/dsh.399-010.1.pdf
3. Katalogové listy MC1602E - http://www.gme.cz/_dokumentace/dokumenty/513/513-128/dsh.513-128.1.pdf
4. Odkaz na program Bascom AVR - http://www.mcselec.com/index.php?option=com_content&task=view&id=14&Itemid=41
5. Odkaz na program Eagle - http://www.eagle.cz/download.htm