VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ROZŠIŘUJÍCÍ VÝUKOVÝ MODUL K MIKROKONTROLÉRU ATMEGA

EXTENSION EDUCATIONAL MODULE TO A MICROCONTROLLER ATMEGA

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MIROSLAV NEVRKLA
AUTHOR

VEDOUCÍ PRÁCE — ING. DANIEL ZUTH, PH.D
SUPERVISOR

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ, FAKULTA STROJNÍHO INŽENÝRSTVÍ

Ústav automatizace a informatiky
Akademický rok: 2013/2014

ZADÁNÍ BAKALÁRSKÉ PRÁCE

student(ka): Miroslav Nevrkla

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a
zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Rozšiřující výukový modul k mikrokontroléru ATmega

v anglickém jazyce:

Extension educational module to a microcontroller ATmega

Stručná charakteristika problematiky úkolu:

Práce se bude zabývat návrhem a realizací rozšiřujícího výukového modulu k mikrokontroléru
ATmega128. Rozšiřující modul bude obsahovat periferie pro ovládáni DC motoru. Součástí práce
budou vzorové příklady k navrženým periferiím. Práce je součástí řešení projektu IGA VUT Brno,
FSI-S-11-31, Aplikace metod umělé inteligence.

Cíle bakalářské práce:

- Seznámení s problematikou mikrokontroléru
- Navrhnout vhodné prvky pro realizaci
- Navrhnout schéma a DPS modulu
- Vytvořit vzorové programy
- Realizovat a otestovat funkci navrženého modulu

SEZNAM ODBORNÉ LITERATURY:

[1] MATOUŠEK, David. Práce s mikrokontroléry ATMEL AVR . 2. vyd. Praha: BEN, 2006, 375 s. ISBN 80-730-0209-4.
[2] MANN, Burkhard. C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy -
linkery, práce s ATMEL AVR a MSC-51, příklady programování v jazyce C, nástroje pro
programování, tipy a triky. Vyd. 1. Praha: BEN, 2003, 279 s. ISBN 80-730-0077-6.

Vedoucí bakalářské Vedoucí bakalářské práce: Ing. Daniel Zuth, Ph. D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2013/2014.

V Brně, dne 24.11.2013

L.S.

Ing. Jan Roupec, Ph. D. Ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. Děkan fakulty

Abstrakt

Předmětem bakalářské práce je návrh a realizace rozšiřujícího výukového modulu pro řízení stejnosměrných a krokových motorů a následné ověření funkčnosti řízení pomocí mikrokontroléru ATmega128. Tato práce je součástí řešení projektu IGA VUT BRNO, FSI-S-11-31, aplikace metod umělé inteligence.

Abstract

Subject of this bachelor thesis is design and realization extension educational module for drive direct current and stepper motors and subsequent verification of functionality by microcontroller ATmega128. This thesis is part of solving project IGA VUT BRNO, FSI-S-11-31, application of methods artificial intelligence.

KLÍČOVÁ SLOVA

Řízení stejnosměrného a krokového motoru, ATmega128, výukový modul

KEYWORDS

Direct current and steeper motor control, ATmega128, educational module

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně dle pokynů vedoucího a s použitím uvedené odborné literatury.

V Brně, dne 27.5.2014

BIBLIOGRAFICKÁ CITACE

NEVRKLA, M. *Rozšiřující výukový modul k mikrokontroléru ATmega.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014. 49 s. Vedoucí bakalářské práce Ing. Daniel Zuth, Ph.D..

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Danielu Zuthovi, Ph. D za odborné vedení, ochotu, trpělivost a čas při vytváření této bakalářské práce.

Obsah

1 ÚVOD

Cílem této práce je seznámit se s problematikou ovládání motorů pomocí mikrokontroléru a realizovat rozšiřující modul, který bude dále využíván v předmětu Hardware a mikroprocesorová technika.

Hlavním úkolem rozšiřujícího výukového modulu je seznámit studenty s ovládáním stejnosměrných a krokových motorů. Mohou si zde vyzkoušet pulsně šířkovou modulaci pro řízení rychlosti otáčení stejnosměrných motorů nebo přesně nastavovat úhel otočení pomocí krokových motorů.

Elektrické motory nacházejí široké uplatnění například ve výpočetní technice, v robotice, u obráběcích strojů, v automobilovém nebo leteckém průmyslu. Z těchto příkladů je patrné jejich široké uplatnění. Krokové motory se využívají hlavně v aplikacích, kde je požadována přesná polohovatelnost, jako je například robotika, kde jsou vysoké nároky na přesnost. U stejnosměrných motorů je důležité řídit rychlost otáčení a směr.

Studentům je tak dána možnost seznámit se s ovládáním těchto motorů a po absolvování kurzu by měli být schopni tyto motory ovládat. K řízení se využívá mikrokontrolér, kterým je možno řídit směr otáčení, rychlost otáčení a u krokových motorů navíc použít různé způsoby řízení. Mikrokontrolér je možno naprogramovat například v jazyce C nebo v assembleru.

Tato práce se v teoretické části bude zabývat popisem typů stejnosměrných a krokových motorů. Následně se práce bude zabývat problematikou změny směru otáčení motorů a rozborem jednotlivých integrovaných obvodů. Dále zde budou rozebrány jednotlivé způsoby řízení u krokových motorů a některé z nich budou prezentovány ve výsledném demonstračním programu. V Hardwarové části bude řešen návrh schématu a zvolený postup výroby plošného spoje a nakonec v Softwarové části budou podrobně rozebrány jednotlivé funkce hlavně pro řízení motorů.

2 TEORETICKÁ ČÁST

V první kapitole budou teoreticky popsány typy krokových a stejnosměrných motorů a jejich způsoby řízení. Také zde budou popsány využité integrované obvody a mikrokontrolér ATmega 128.

2.1 Stejnosměrné stroje

Stejnosměrné stroje jsou historicky nejstarší elektromagnetické stroje. Skládají se ze statoru, rotoru, sběracího mechanismu a ventilátoru.

Stator se skládá z litého nebo svařovaného magnetického věnce, na kterém se upevňují budící póly s pólovými nástavci a budícím vinutím. Stejnosměrný budící proud, který lze získat z cizího zdroje nebo vlastních svorek, vybudí v cívkách hlavní magnetický tok. Cívky musí být zapojeny tak, aby vytvářely střídavě severní a jižní pól. Mezi hlavními póly jsou umístěny komutační póly, avšak nejsou pro činnost stejnosměrného stroje nutné. Jejich účelem je vytvoření dobrých komutačních podmínek jako je zmenšení jiskření mezi komutátorem a kartáči.

Rotor se skládá z elektrotechnických plechů. V drážkách rotoru jsou uloženy cívky stejnosměrného vinutí, kterými protéká proud kotvy (rotoru). Na klínovité měděné lamely komutátoru jsou vyvedeny začátky a konce cívek . Komutátor zajišťuje, aby vodič, který se nachází pod pólem, měl stálý směr proudu a tím i tažná síla a točivý moment působily stále stejným směrem.

Sběrací mechanizmus je složen z uhlíkových kartáčů a jejich držáků. V nich jsou upevněny uhlíkové kartáče, které dosedají na komutátor. [1]

2.1.1 Dělení stejnosměrných motorů

Stejnosměrné motory můžeme dělit podle druhu buzení. Jedná se o motory s cizím buzením, s paralelním buzením (derivačním), se sériovým buzením a s se smíšeným buzením (kompaundním). [1]

2.1.1.1 Motor s cizím buzením

Motor s cizím buzením nemá propojeno buzení s obvodem kotvy, ale je napájeno vnějším zdrojem stejnosměrného proudu. Patří sem i motory s permanentními magnety namísto budících cívek. Při rozběhu a při snižování otáček je snižováno napětí na kotvě, například pomocí spouštěcího odporu. Ke zvýšení otáček, pomocí statorového vinutí, nad jmenovité otáčky je možno použít regulační odpor v obvodu budícího vinutí, kterým je možno snížit budící proud. Přes usměrňovač střídavého napětí je často napájen obvod kotvy i obvod budícího vinutí. Regulačním transformátorem nebo řízeným usměrňovačem může být snižováno napájecí i budící napětí. Cizí buzení je nezávislé na napětí na kotvě motoru a zůstává tak nezměněné i při poklesu napětí na kotvě, což neplatí u ostatních motorů. Otáčky těchto motorů jsou ve srovnání s derivačními motory ještě stabilnější při kolísání zatížení. Dále jsou používány jako pohony strojů s proměnlivým mechanickým odporem, například pohon obráběcích strojů. Při běžném zatížení a nízkých otáčkách, je potřeba intenzivnější chlazení. [2]

2.1.1.2 Derivační motor

U derivačního motoru je budící vinutí připojeno paralelně k vinutí kotvy. Otáčky mohou být regulovány spouštěcím odporem a odporem regulátoru budícího pole. Při běhu naprázdno i při zatížení se chová derivační motor stejně jako motor s cizím buzením. Motory s chováním derivačních motorů nazýváme takové, které se při běhu naprázdno nepřetočí a při rostoucím zatížení mají jen malý pokles otáček. Nelze je však řídit napětím na rotoru jako u motorů s cizím buzením. Musíme zajistit, aby nedošlo k odpojení buzení, protože by se pak mohla kotva ve slabém zbytkovém poli roztočit do příliš vysokých otáček. To samé platí i u motorů s cizím buzením se kterými má shodné použití. [2]

2.1.1.3 Motor se sériovým buzením

U Motoru se sériovým buzením je vinutí kotvy zapojeno sériově s vinutím buzení. K rozběhu a řízení otáček je použit předřazený stavitelný spouštěcí odpor. Veškerý proud kotvy protéká také budícím vinutím, což znamená, že je stejně velký. Tyto motory se vyznačují největším rozběhovým momentem v porovnání s ostatními stejnosměrnými motory. Při rozběhu bez zatížení postupně klesá proud a proto slábne budící pole, které podporuje další nárůst otáček. Motory se při rozběhu naprázdno přetočí. Dále nesmí být spojovány se zátěží plochými řemenicemi a plochými řemeny, protože plochý řemen může snadno spadnout. Pokud naroste zatížení motoru naroste i společný proud v kotvě a také v budícím vinutí. Poté klesnou otáčky a naroste točivý moment, z čehož vyplývá, že otáčky motoru jsou velmi závislé na zatížení. Je-li motor v provozu se spouštěcím odporem, pak je pokles otáček obzvláště velký. Pokud naroste proud se pak ještě zmenší napětí o úbytek na spouštěcím odporu. Otáčky poklesnou, protože kotva musí vytvářet menší protisměrné napětí. Tyto motory se používají pro pohon vozidel jako jsou například tramvaje, trolejbusy. Je-li stator motoru vyroben z elektroplechů, pracuje motor i na střídavý proud. [2]

2.1.1.4 Kompaundní motor

Kompaundní motor je stejnosměrný motor se sérioparalelním buzením. Otáčky lze regulovat odporem spouštěče nebo odporem regulátoru pole. V kompenzovaném kompaundním motoru je sériové budící vinutí zapojeno tak, že jeho magnetické pole má stejný směr jako pole paralelního vinutí. Při běhu naprázdno se chová jako derivační motor. Pokud je zatížen, otáčky klesají trochu rychleji, neboť s rostoucím proudem kotvy roste i hlavní magnetický tok. Motor je velmi nestabilní a lehce se přetočí, pokud je sériové budící vinutí zapojené tak, že jeho pole oslabuje paralelní vinutí. K takovému zapojení může dojít omylem při přepólování směru otáčení. Poté při stoupajícím proudu rostou otáčky, protože slábne hlavní pole. Toto zapojení je výjimečně používáno ke zmenšení vlivu kolísavého zatížení na otáčky motoru, i přes velkou nestabilitu naprázdno. Nárůst zatížení má snahu motor zpomalit, avšak narůst proudu doprovázející narůst zátěže oslabí hlavní pole, což vede ke snaze zvýšit otáčky, které jsou pak stabilní. Tyto motory se používají tam, kde nestačí malý rozběhový moment derivačních motorů například u zdvihacích mechanismů. Velké derivační motory s cizím buzením s výkonem nad 12 kW většinou mívají pomocné sériové budící vinutí a chovají se pak jako kompaundní motor. Bez sériového budícího vinutí by docházelo k nárůstu otáček při zatížení, protože pole kotvy oslabuje hlavní pole. [2]

2.2 Krokový motor

Stejnosměrná vinutí mohou být pomocí stejnosměrných impulzů nastavena na proměnou polaritu. K přepólování jednotlivých pólových párů dochází pomocí změny směru proudu v jednotlivých vinutích. Dochází-li k přepólování postupně v jednom směru, vznikne točivé magnetické pole, které se může měnit po krocích, nebo určitou rychlostí otáček. Kotva z permanentního magnetu je vždy natočena podle polarity pole statoru.
Vyrábějí se jako jednofázové nebo vícefázové. Polarita statorových pólů může být měněna dvěma způsoby.

Je-li každé vinutí tvořeno dvěma cívkami, jedná se o unipolární konstrukci. Každá cívka je zdrojem magnetického toku v jednom směru. Přepínáním cívek se mění polarity pólových párů statoru.

Je-li budící vinutí tvořeno pouze jednou cívkou a přepólování je realizováno změnou směru proudu v cívce jedná se o bipolární konstrukci.
Smysl otáčení lze obrátit změnou pořadí proudových impulzů. Dále krokové motory vyžadují speciální řídicí techniku. [2]

2.2.1 Krokový motor s malým krokovým úhlem

Požadovány velký počet pólů krokového motoru, vyžaduje speciální konstrukci. Pro velikost kroku 7,5° je motor konstruován na Jednopólovém principu. Na hřídeli motoru je permanentní magnet s axiálně uspořádanými póly, na jejichž čelních stranách jsou ozubená pólová kola. Zuby každého ozubeného pólového kola jsou stejné polarity. Vzájemným mechanickým pootočením ozubených kol o půl zubové rozteče je dosaženo střídání polarity na obvodu rotoru. Stator složený z plechů a má dvě vinutí. Každé se skládá ze dvou sériově zapojených cívek, které vytvářejí protikladné statorové póly. Ozubené vnitřní dělení statoru je shodné s ozubeným dělením rotoru. Ozubený rotor se vždy ustálí v poloze, ve které je magnetickému toku kladen co nejmenší odpor.
Krokový motor na jednopólovém principu má mnoha pólový rotor a umožňuje pootáčení s malým úhlem otočení. Je-li krokový moto napájen konstantním napětím, narůstá se zvětšujícím se kmitočtem jalový indukční odpor statoru. Tím klesá odběr proudu (činného) a točivý moment motoru. Pokud by měl při nárůstu kmitočtu zůstat zachován proud a tím i točivý moment, musel by být motor napájen řízeným zdrojem konstantního proudu. Při normálním zatěžovacím momentu se krokový motor vždy pootočí o krokový úhel, který odpovídá řídícímu impulzu. Může však dojít k tomu, že zatěžovací úhel naroste do velikosti jednoho krokového úhlu. Protože se však tato chyba nepřipočítává znovu s každým krokem, nemůže být na konci série impulzů chyba větší, než je krokový úhel. Při odpojení proudu statoru vznikne díky působení magnetů rotoru zároveň pevný zbytkový moment. Krokové motory s úhlem od 7,5° jsou konstruovány s jazýčkovými póly. Pro jednoduchou konstrukci a velkou spolehlivost jsou krokové motory používány pro řízené pohony, dálkové řízení, pohony tiskáren, počítadel a v jiných oblastech řídící a regulační techniky. [2]

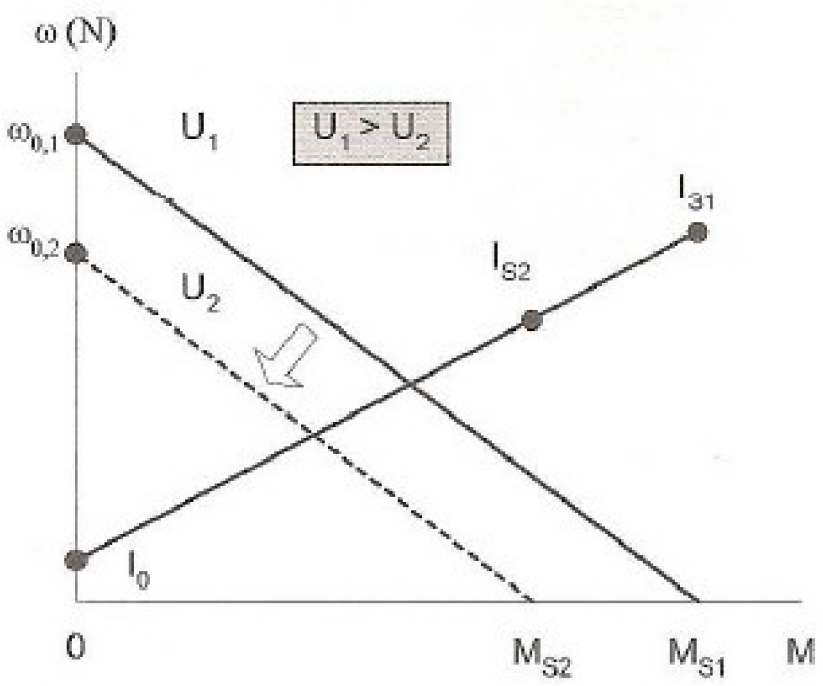
2.3 Řízení otáček stejnosměrných motorů

S narůstajícím napětím na kotvě zpočátku roste i proud a také moment, který zrychluje otáčení kotvy. S narůstajícími otáčkami roste i protisměrné napětí kotvy. Pokud je při konstantním točivém momentu motoru dosaženo zvýšení napětí na kotvě, je před změnou i po změně otáček proud stejný. Jmenovitých otáček při jmenovitém buzení motor dosáhne, je-li napětí na kotvě zvýšeno na jmenovité napětí. Obecně se dá říci, že zvyšováním napětí na kotvě jsou zvyšovány otáčky až na jmenovitou hodnotu.

Kvůli velkým tepelným ztrátám jsou spouštěcí odpory používány jen k rozběhu motorů malých výkonů. K samotnému řízení otáček pomoci změny napětí se proto používá téměř bezztrátová tyristorová regulace. Pulzující proud není pro buzení vhodný, což znamená, že pro cizí buzení může být kotva napájena buď ze stejnosměrné sítě přes pulzní měnič nebo z trojfázové nebo střídavé sítě přes řízený usměrňovač.

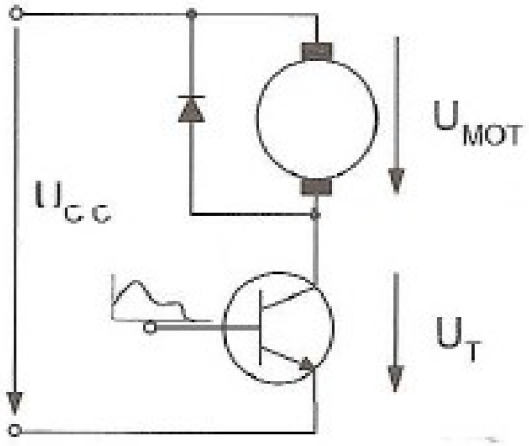
Dalším způsobem, jak zvýšit otáčky na jmenovitou hodnotu je zeslabení budícího pole. K udržení otáček při kolísavém zatížení slouží regulace budícího pole, nicméně ji nelze použít k velkému zvýšení otáček, aby nedošlo k mechanickému poškození rotoru vlivem odstředivých sil.

Změnou napájecího napětí motoru lze ovlivnit otáčky motoru, jak je znázorněno na ilustraci 1. Z této ilustrace je zřejmé, že kromě snížení otáček dojde také ke snížení momentu. Je-li potřeba snížit otáčky motoru a zároveň zvýšit jeho moment, je nutno použít mechanickou převodovku. Jmenovité otáčky těchto motorů bývají zpravidla v řádu jednotek tisíců za minutu. [2][3]



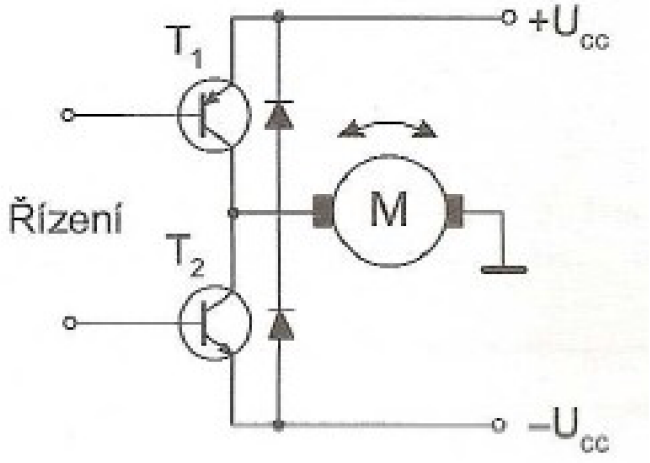
Ilustrace 1: Vliv napětí na otáčky.

Z energetického hlediska však není příliš vhodná nejjednodušší plynulá změna napětí na motoru. Jak je znázorněno na ilustraci 2, nejjednodušší je jednokvadrantový proudový regulátor. Moment a otáčky mohou být generovány v jednom směru. Řídícím napětím je ovládán výkonový operační zesilovač, případně výkonový tranzistor. Napájecí napětí Ucc se rozdělí na úbytek na motoru Umot a na úbytek na výkonovém tranzistoru Ut. Výkon se přemění v teplo. Ztráty jsou však u malých výkonů přijatelné do cca 30 – 50 W a oproti celkové účinnosti pak převáží výhoda slabého rušení zároveň s nízkou cenou výkonové jednotky. [3]

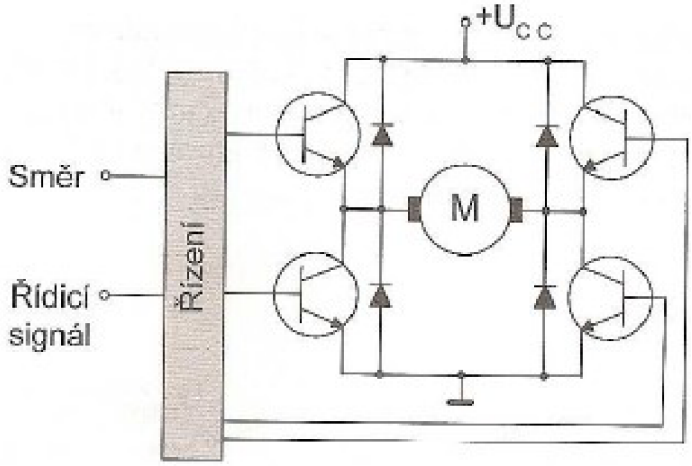


Ilustrace 2: Jednokvadrantový proudový regulátor otáček.

V případě potřeby měnit polaritu z důvodu změny smyslu otáčení musíme využít bipolárního způsobu napájení motoru. To nám umožňuje například bipolární zdroj viz. ilustrace 3, kde je jeden tranzistor řízen a druhý uzavřen a naopak. V dnešní době se používají můstkové spínače tzv. H-můstek a to v provedením lineárním pro spojité řízení velikosti proudu a nebo spínacím pro PWM řízení. Obě tato zapojení principiálně vycházejí ze zapojení na ilustraci 4 a odlišují se typy použitých tranzistorů, jako jsou lineární nebo spínací a především v bloku označeném jako řízení. [3]



Ilustrace 3: Řízení otáček a smyslu otáčení.



Ilustrace 4: Můstkový zesilovač.

Následující způsob řízení proudu je založen na rychlé změně plného napětí motoru – napětí vypnuto/zapnuto. Neměnné napájecí napětí je časově rozděleno do impulzů s konstantní frekvencí a s pulzně šířkovou modulací. Jelikož spínací výkonové tranzistory jsou buď zcela zavřeny nebo otevřeny, nejsou na nich téměř žádné ztráty. Kvůli indukčnosti na vinutí motoru proud nestačí sledovat rychlé změny napětí a proudový průběh je víceméně zvlněný. Střední hodnotě protékaného proudu budou úměrné otáčky. Lépe vyhlazeného proudu dosáhneme při vyšší frekvenci pulzů a při vyšší indukčnosti vinutí. [3]

2.4 Metody řízení krokových motorů

Pokud prochází proud právě jednou cívkou, nazýváme takové řízení unipolárním. Motor dodává nejmenší kroutící moment, avšak má také nejmenší odběr s tímto buzením. Mezi výhody tohoto řešení patří jednoduché zapojení řídící elektroniky, kde ve své podstatě stačí jeden tranzistor na každou cívku. Pro menší motory je vhodný integrovaný obvod ULN2803, kterým je možno řídit dva motory.

Pokud proud prochází vždy dvěma protilehlými cívkami, řízení nazýváme bipolární. Cívky jsou zapojeny tak, že mají opačně orientované magnetické pole. Motor dodává větší kroutící moment na rozdíl od unipolárního řízení, avšak s tím jde také vyšší spotřeba. Pro dané řízení jsou potřeba dva H-můstky, což znamená že pro každou větev je potřeba jeden. Z toho vyplývá, že zapojení, že zapojení je složitější. Pro menší motory je vhodný integrovaný obvod L293D. [4]

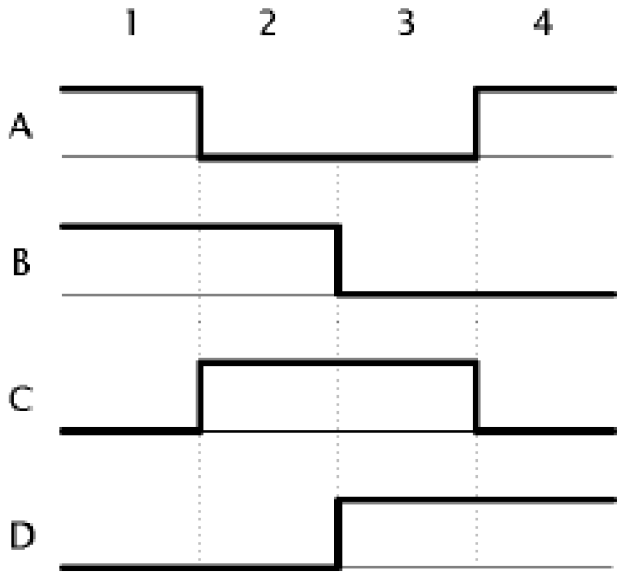
2.4.1 Čtyřtaktní řízení s magnetizací jedné fáze

Je to nejjednodušší způsob řízení pro čtyř-fázové reakční krokové motory nebo pro dvoufázové hybridní krokové motory v zapojení pro unipolární buzení viz. ilustrace 5. Pokud je fáze A buzena, vznikne elektromagnetické pole v ose pólových nástavců nesoucích vinutí fáze A. Aby došlo k pootočení rotoru, musí se nejbližší pár pólových nástavců dostat do místa vytvořeného fází A, což je oblast s největší intenzitou magnetického pole. Po vypnutí buzení fáze A se začne napájet fáze B a další pólový pár se pootočí o 15° proti směru hodinových ručiček a dostane se do nové rovnovážné

polohy. K následujícímu pootočení rotoru dojde při vypnutí buzení fáze B a připojení fáze C a následně i fáze D. Následující kroky se provádějí znovu od fáze A až po D. Změna směru lze docílit pomocí opačného pořadí spínaní rotoru od D po A. Nezáleží zde na vzájemné polaritě napájecího napětí případně proudu jednotlivých fází. [3]

2.4.2 Čtyřtaktní řízení s magnetizací dvou fází

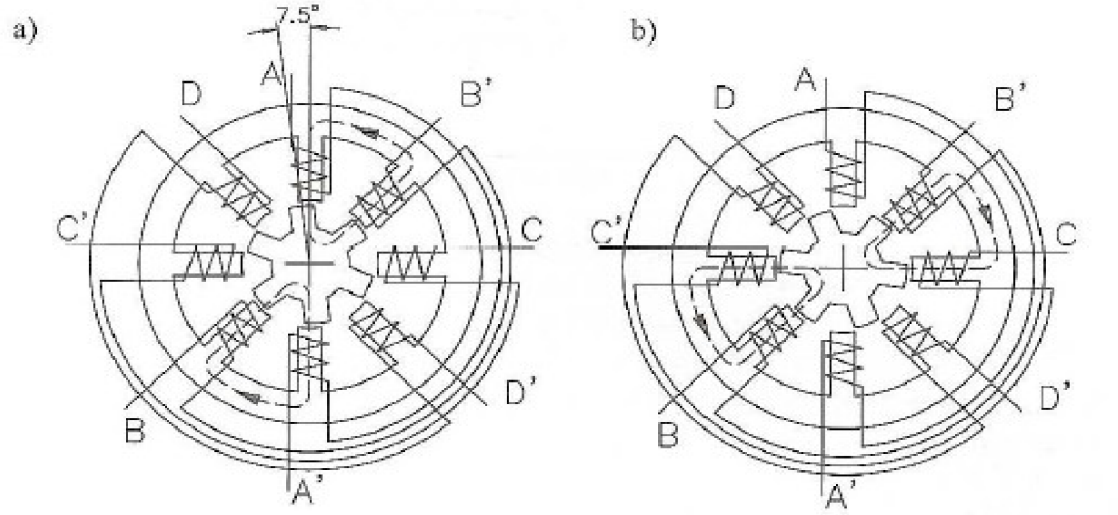
Jedná se o způsob buzení jednotlivých fází, kdy jsou napájeny dvě sousední fáze. Rovnovážná poloha rotoru se vždy nachází mezi vybuzenými sousedními pólovými nástavci. Na rozdíl od předcházejícího řízení je je fáze vychýlena o 7,5°, avšak velikost kroku je stále 15°. Průběh spínaní fází je tedy AB-BC-CD-DA a pro opačný směr AD-CD-BC-AB.



Ilustrace 5: Čtyřtaktní řízení s magnetizací dvou fází.

Na Ilustraci 6a je zachycen stav, kdy jsou napájeny fáze AB. Dále jsou zde zobrazeny uzavřené magnetické toky s označením směrů. Na ilustraci 6b je rotor v rovnovážné poloze díky sepnutým fázím BC. Zde záleží na polaritě napájení jednotlivých vinutí a tím pádem i na směru generovaného magnetického toku, nicméně toto je zajištěno konstrukčně.

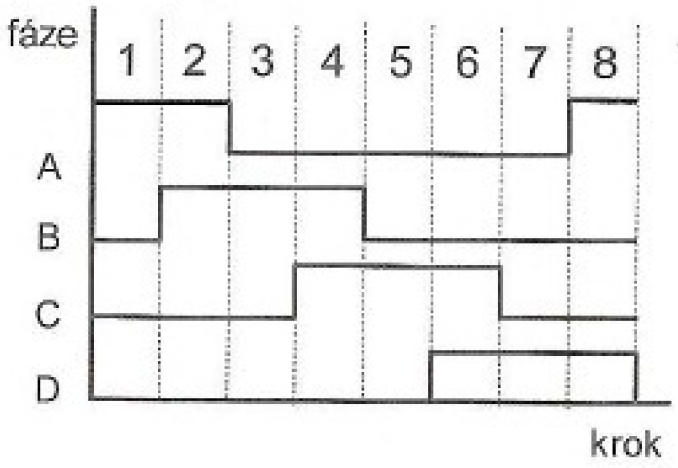
Jako výhody tohoto způsobu je například zvýšení statického vazebního momentu a větší tlumící účinky mechanických oscilací při krokování. [3]



Ilustrace 6: První a druhý krok čtyřtaktního řízení s magnetizací dvou fází pro čtyřfázový krokový motor.

2.4.3 Osmitaktní řízení

Složením čtyřtaktního řízení s buzením jedné a dvou fází vznikne řízení osmitaktní. Princip spočívá v tom, že se střídá buzení jedné a dvou fází v sekvenci A-AB-B-BC-C-CD-D-AD-... Opačný směr je v opačně sekvenci, jak bylo vysvětleno na čtyřfázovém krokovém motoru s velikostí kroku 15°. Zde se však rotor pootáčí o polovinu kroku tedy 7,5°. Sekvence kroků je znázorněna na ilustraci 7.



Ilustrace 7: Osmitaktní řízení s magnetizací jedné nebo dvou fází pro čtyřfázový krokový motor - časové průběhy.

Toto řízení nám umožňuje dvojnásobný počet kroků na otáčku. Této výhody dosahujeme bez jakékoliv úpravy budicích obvodů. Na druhou stranu toto řízení způsobí, že můžeme mít různou velikost maximálního momentu ve statické momentové charakteristice pro buzení jedné nebo dvou fází. Nicméně tato vlastnost lze odstranit tím, že budeme napájet fáze nižším proudem v krocích kdy jsou buzeny dvě. Velikost proudu závisí na velikosti kroku.[3]

2.4.4 Mikrokrokování

Pro některé aplikace je požadováno velmi jemné rozlišení polohy, například o zlomky stupně nebo je nutno snížit mechanické rázy. Jedná se především o aplikace jako jsou tiskárny, fototechnika, mikrotechnika, robotika atd. Zvýšením počtu fází nebo rotorových zubů můžeme dosáhnout zmenšení úhlu kroku, avšak v praxi není výhodné pracovat s větším počtem fází než 4 a také je obtížné vyrobit rotor s více než 100 zuby, což znamená, že motory s úhlem pod 1°jsou výjimkou. Ke zvýšení počtu kroků na otáčku je tedy vhodný způsob mikrokrokování.
Vychází z metody magnetizace dvou fází, nicméně na rozdíl od této metody jsou u mikrokrokování fáze napájeny různou velikostí proudu. Libovolné rovnovážné polohy mikrokroku mezi dvěma sousedními kroky můžeme dosáhnout zvolením vhodných vzájemných velikostí proudů.

V následující tabulce je rozdělení základního kroku na 4 mikrokroky.. Tab. 1 jsou shrnuty budící proudy pro jednotlivé mikrokroky.

Poloha mikrokroku	Proud první fáze	Proud druhé fáze
0	I	0
¼	I	kI
½	I	I
¾	kI	I
1	0	I

Tabulka 1: Rozdělení kroku na mikrokroky.

Nejprve je rotor vyrovnán fází 1 a to tak, že ve výchozí poloze je fáze 1 buzena jmenovitým proudem I a fáze 2 není buzena. Současným buzením fáze 1 proudem I a fáze 2 proudem kI je dosaženo polohy ¼ základního kroku. Koeficient k je kladný a menší jak 1. Buzením obou fází jmenovitým proudem I je dosaženo druhého kroku. Pro třetí krok je fáze 2 napájena proudem I a fáze 1 proudem kI a nakonec při vypnutí fáze 1 a plném buzení fáze 2 je dosaženo plného kroku.

Ve výše uvedeném případě je nutné mít dvouhladinový napájecí zdroj s hodnotami I a kI. Při vyšším počtu mikrokroků jsou také vyšší požadavky na napájecí a spínací obvody. Vyrábí se speciální integrované obvody pro řízení motorů s využitím mikrokrokování, což bývá řešeno pomocí D/A převodníků. Velikost maximálního úhlu nemá vliv na mikrokrokování a zůstává pro daný typ motoru konstantní.

Přínosem mikrokrokování je snížení mechanických rezonancí krokového motoru vyplývající z jeho nízkých tlumících vlastností. Způsob buzení jednotlivých fází krokového motoru a charakteristiky poháněné zátěže ovlivňují tyto rezonance. Kvůli rezonancím může nastat ztráta synchronizace a rotoru a následně i ztrátu kroku. Při použití řídícího systému bez zpětné vazby může dojít k závažné a neodstranitelné chybě polohy. K redukování další mechanických vibrací krokového motoru se dosahuje různými časovými průběhy proudů jako jsou sinusové, lichoběžníkové a trojúhelníkové. [3]

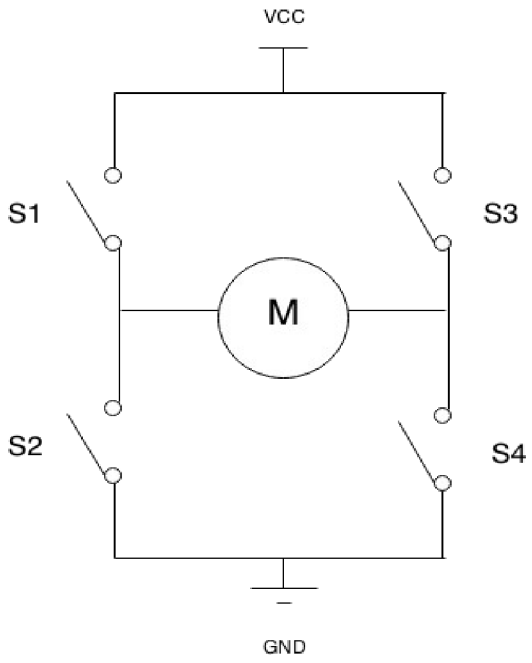
2.4.5 Řízení krokového motoru se sníženou energetickou náročností

Obvykle se u řízení krokového motoru předpokládá, že je vždy napájeno alespoň jedno vinutí. Napájený krokový motor vyvíjí statický synchronizační moment, aby byl rotor v rovnovážné poloze po vykonání kroku. Odezva pohonu na řídící impulz představuje tlumený kmitavý děj. Mírou tlumení systému je poměr doby přechodu v době, kdy pohon poprvé překmitne novým rovnovážným stavem a dobou ustálení, po které se pohon blíží k novému rovnovážnému stavu s odchylkou +-5%. Jako čas ustálení můžeme považovat aktivní činnosti pohonu. Jestliže po době ustálení mechanická část nezpůsobuje zpětné silové působení na pohon, napájení vinutí po době ustálení zvyšuje celkovou energetickou spotřebu pohonu a proto není potřeba po ustálení napájet vinutí. Připojením indexové zátěže nebo působením vnitřního magnetického pole v nenapájeném krokovém motoru, což znamená, že je na motoru stále zbytkový moment, je udržena poloha.
Zmiňovaný způsob je použitelný pouze u čtyřtaktního řízení s buzením jedné fáze a čím je frekvence řídících impulzů menší, tím je energeticky výhodnější. Díky tomuto uspořádání může dojít ke zjednodušení napájecího zdroje pro fázová vinutí krokového motoru a snížení tepelné zátěže spínacích prvků pro jednotlivá fázová vinutí. [3]

2.5 H-můstek

Jedná se o elektronický obvod, který umožňuje přepínat napětí v obou směrech. Tyto obvody jsou často používány v robotice a jiných aplikacích, kde je potřeba otáčet motory doprava nebo doleva.

Je k dispozici v integrovaných obvodech nebo je možno sestavit H-můstek z jednotlivých součástek. Termín H-můstek je odvozen z typického zapojení obvodu viz. ilustrace 8.



Ilustrace 8: Schéma H-můstku.

Je sestaven ze čtyřech přepínačů. Když jsou zavřeny spínače S1 a S4 a spínače S2 a S3 jsou otevřeny, objeví se na motoru kladné napětí. Avšak pokud jsou sepnuty

spínače S1 a S4 a tím pádem jsou zavřeny spínače S2 a S3 objeví se na motoru záporné napětí. Důležité je, aby spínače S1 a S2 nebyly zavřeny v jeden okamžik, protože by došlo ke zkratu na vstupním napěťovém zdroji. To stejné platí pro spínače S3 a S4.

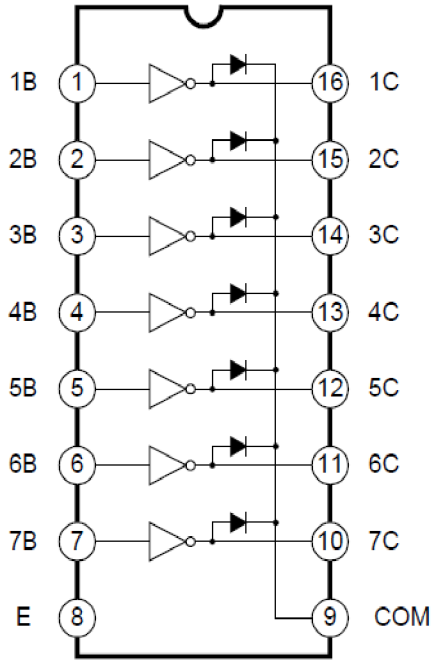
Z hlavním důvodů použití H-můstku je změna polarity napájení motoru a tím pádem změna směru otáčení motoru, ale také k brzdění motoru jak znázorňuje tabulka 2. [5] [6]

S1	S2	S3	S4	Výsledek
1	0	0	1	Otáčení doprava
0	1	1	0	Otáčení doleva
0	1	0	1	Brzdění motoru
1	0	1	0	Brzdění motoru
0	0	0	0	Volný chod
0	0	1	1	Zkrat
1	1	0	0	Zkrat
1	1	1	1	Zkrat

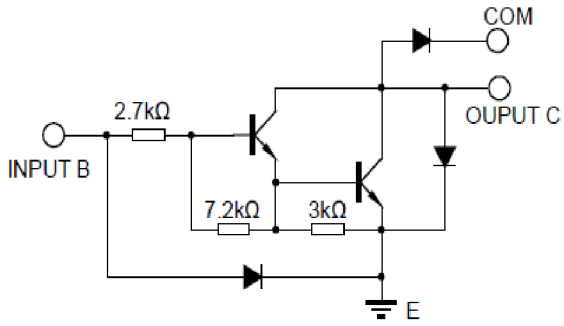
Tabulka 2: Ovládání motoru.

2.6 Integrovaný obvod ULN2003

Je tvořen monolitickým vysoko napěťovým a proudovým darlingtonovým tranzistorovým polem ilustrace 9. Obsahuje 7 NPN darlingtonových párů v provedení s otevřeným kolektorem. Pro činnost s TTL nebo 5V CMOS zařízením, používá 2.7 kW rezistor připojený sériově do báze pro každý darlingtonový pár viz. ilustrace 10. Každý kanál má jmenovitý proud 500 mA, ve špičce až 600mA, při napětí 50 V. Toto zařízení je vhodné například pro řízení stejnosměrných motorů a osvětlení LED displeje. [7]



Ilustrace 9: Schéma logiky zapojení.

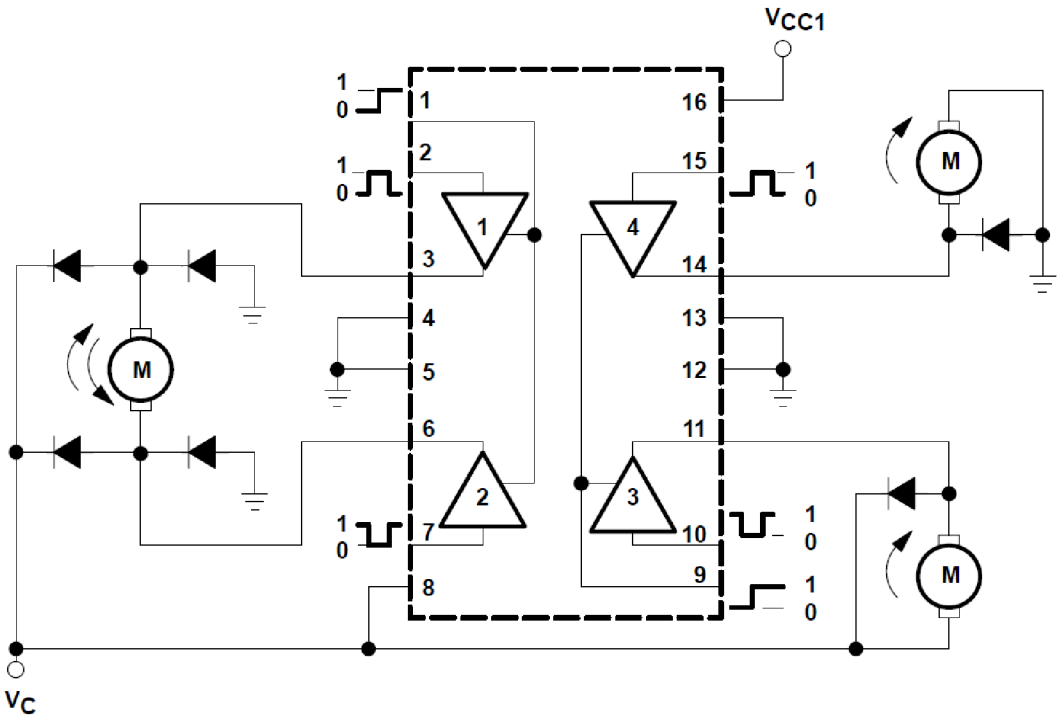


Ilustrace 10: Schéma zapojení darlingtonového. páru

2.7 Integrovaný obvod L293D

Tento integrovaný obvod je tvořen dvěma H-můstky. Je navržen pro obousměrné řízení motorů a to při proudu až 600 mA a napětí od 4.5 V do 36 V. Tento integrovaný obvod je navržen pro řízení induktivní zátěže jako je relé, solenoid, stejnosměrný a bipolární krokový motor. Všechny vstupy jsou TTL (tranzistor - tranzistor logic) kompatibilní.

Vstupy 1 a 2 jsou povoleny pomocí vstupu EN1,2 a vstupy 3 a 4 jsou povoleny pomocí vstupu EN3,4. Pokud je na vstupu EN1,2 logická 1, tak jsou přidružené vstupy povoleny. Když je na vstupu logická 0, tak jsou výstupy ve stavu vysoké impedance, tím pádem nejsou povoleny viz. Ilustrace 11. [8]



Ilustrace 11: Blokové schéma.

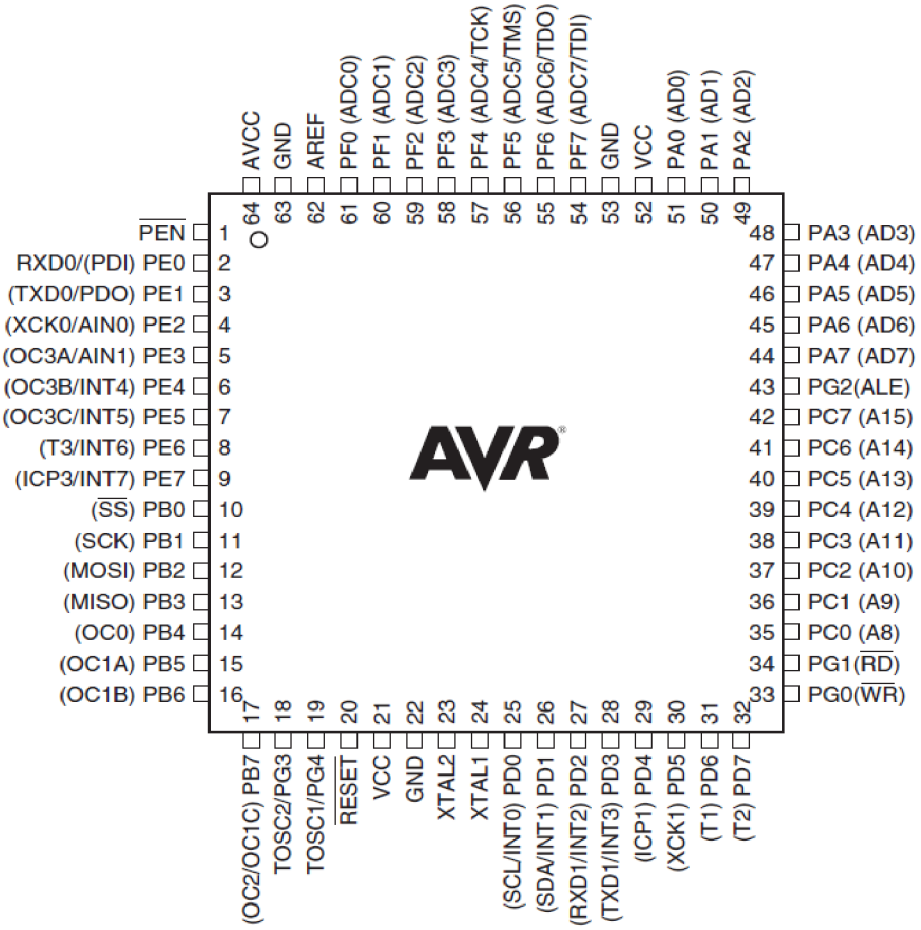
2.8 Mikrokontrolér ATmega128

Jedná se o 8 bitový mikrokontrolér založený na AVR vylepšené RISC architektuře. Instrukce jsou provedeny za jediný takt, ATmega dosahuje průchodnosti blížící se 1MIPS (Million Instruction Per Second) za MHz umožňující optimalizovat spotřebu energie a rychlost zpracování.

Atmel AVR jádro kombinuje bohatou instrukční sadu s 32 univerzálními registry. Všechny tyto registry jsou přímo propojeny s aritmeticko - logickou jednotkou (ALU), umožňující dvěma nezávislým registrům vstoupit do jedné instrukce v jednom taktu. Výsledná architektura je více efektivní co se týče kódu a dosahuje lepší propustnosti a to až 10x rychlejší než konvenční CISC mikrokontroléry. [9]

Vlastnosti mikrokontroléru:

- Výkon - 16MIPS
- 128KB Flash paměť
- 4KB EEPROM paměť
- 4KB interní SRAM paměť
- 53 programovatelných I/O obvodů
- 6PWM kanálů s programovatelným rozlišením 2-16 bitů
- Napájecí napětí 4.5 – 5.5V
- 64 pinů jak v provedení TQFP tak MLF viz. ilustrace 2



Ilustrace 12: Piny ATmegy128.

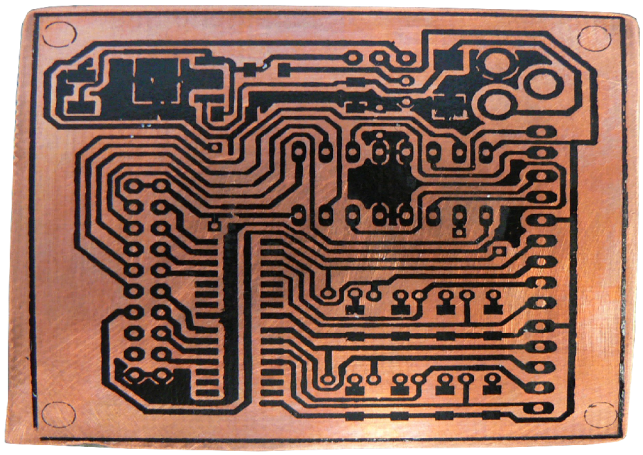
3 VÝROBA PLOŠNÉHO SPOJE

Byla zvolena metoda naţehlení pro výrobu plošného spoje. Nejdříve byl vytisknut návrh plošného spoje z pohledu součástek (neotočenou zrcadlově) na samolepící papír pomocí laserové tiskárny. Tisk proběhl na lepivou stranu papíru, aby se později sám odlepil z Cuprextitové desky.

Tato deska musela být nejprve nastříhána na odpovídající rozměry vytištěného návrhu. Dalším krokem bylo vyčištění desky od nečistot pomocí běžných čistících prostředků. Nakonec pomocí jemného smirkového papíru byly zahlazeny hrany desky a odstraněny všechny nečistoty z povrchu viz. ilustrace 13.

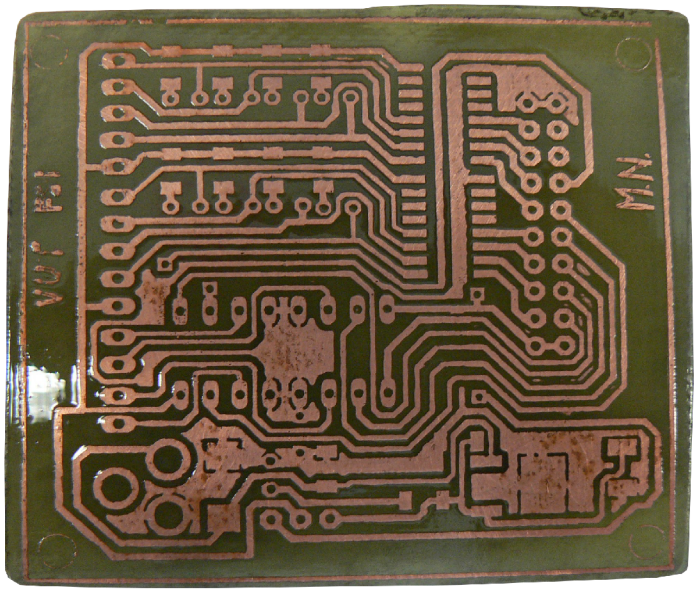
Ilustrace 13: Cuprextitová deska.

Samolepící papír ze strany vytištěné předlohy byl opatrně přiložen na měděnou část desky. Deska s předlohou byla překryta běžným papírem a následně zažehlena. Zažehlovaní probíhalo odhadem 2 minuty. Při zažehlovaní bylo potřeba střídavě vyvíjet tlak na desku s předlohou, aby se toner přilepil na měděnou část desky. Deska s naţehlenou předlohou bylo vložena do misky s vodou a bylo nutno počkat, než se samolepící papír odlepí od desky na kterém zůstane toner s požadovaným návrhem. Při prvním pokusu se toner na některých místech odlepil, proto musela být deska vyčištěna a proces se opakoval. Druhý pokus dopadl uspokojivě viz. ilustrace 14. Zde je patrná výhoda této metody například oproti fotocestě, kde až do odleptání desky není zřejmé, jestli se stala chyba například ve špatném přiložení předlohy na desku a tím pádem může nastat situace, že je deska nepoužitelná.



Ilustrace 14: Deska s nažehlenou předlohou.

Dalším krokem bylo leptání. Podle návodu bylo potřeba nahřát chlorid železitý. Poté byla deska položena nažehlenou částí na hladinu chloridu železitého, kde začala plavat díky povrchovému napětí. Pro snadnější manipulací byla na druhé straně desky přiložena lepící páska. Několikrát byla deska znovu vyjmuta, vizuálně zkontrolována, jestli je celý povrch pokryt chloridem železitým, a jestli na žádném místě nejsou vzduchové bubliny. Proces leptání trval okolo 20 minut. Během této doby byl kontrolován stav naleptání desky. Nakonec byla deska úspěšně odleptána a nalakována proti korozi viz. ilustrace 15.

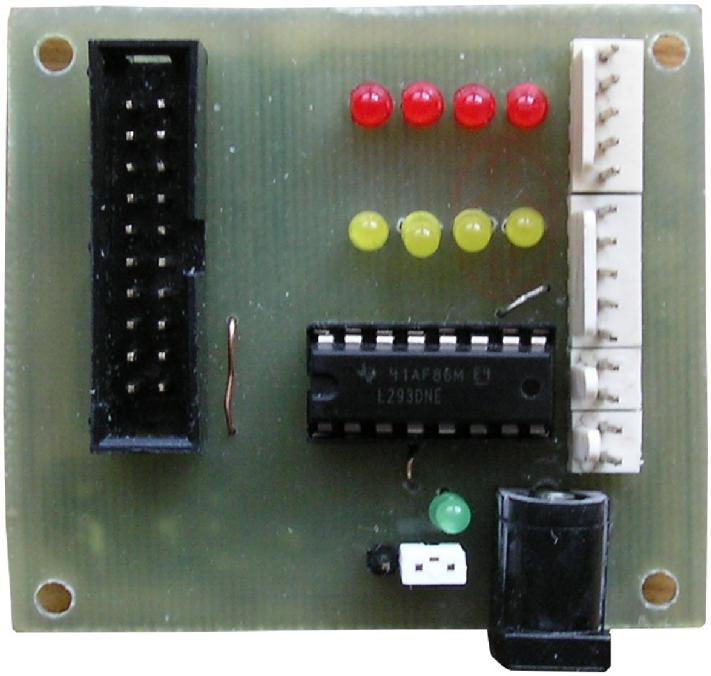


Ilustrace 15: Deska po leptání.

Dále se mohlo přistoupit k zapájení součástek a následnému oživení desky.

4 HARDWAROVÁ ČÁST

Jak je patrné na ilustraci 16, celý projekt je navržen na jedné desce plošného spoje s rozměry 5,8 x 5,3 cm. Tento plošný spoj obsahuje napájecí část, budiče, signalizační LED diody, vstupní a výstupní konektory. Obsahuje tři přemostění z důvodu úspory místa.

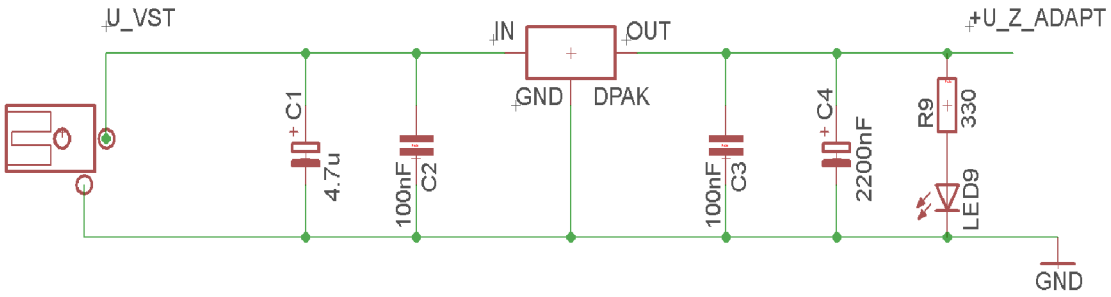


Ilustrace 16: Výukový modul.

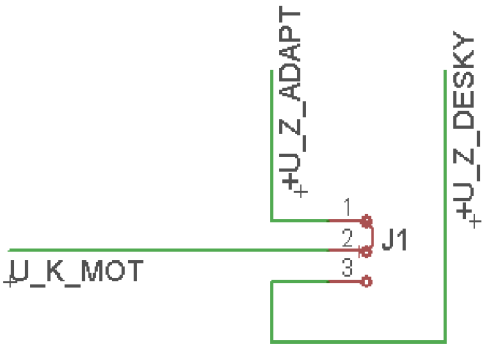
Pro komunikaci s mikrokontrolérem byl použit konektor MLW20, pomocí kterého je možno ovládat jednotlivé motory a pokud se jedná o motory s malým příkonem, mohou být i napájeny přímo z tohoto konektoru bez použití napájecí části. Jako výstupní konektory pro připojení krokových a stejnosměrných motorů byly použity konektory PSH. Pro stejnosměrné motory slouží se dvěma piny a pro krokové motory s pěti piny. U konektorů pro krokové motory, jsou umístěny LED diody, které slouží jako signalizace aktivní fáze. Maximálně je možno připojit až čtyři motory.

4.1 Napájecí část

Je tvořena napájecím souosým konektorem na který je možno přivést 7 – 9V. Kladný pól je připojen na vnitřní část konektoru (trn) a nulový potenciál na vnější část. Z tohoto konektoru je veden přes kondenzátory do integrovaného obvodu 7805. V tomto obvodu je vstupní napětí stabilizováno na 5V. Kondenzátory jsou použity z důvodu filtrace a stabilizace. Za stabilizátor je umístěna zelená LED dioda, která signalizuje připojený napájecí konektor. Na konci této části je připojen jumper viz. ilustrace 18, pomocí kterého je možné přepínat napájení buď z této napájecí části, ilustrace 17 a nebo přímo z konektoru MLW20.

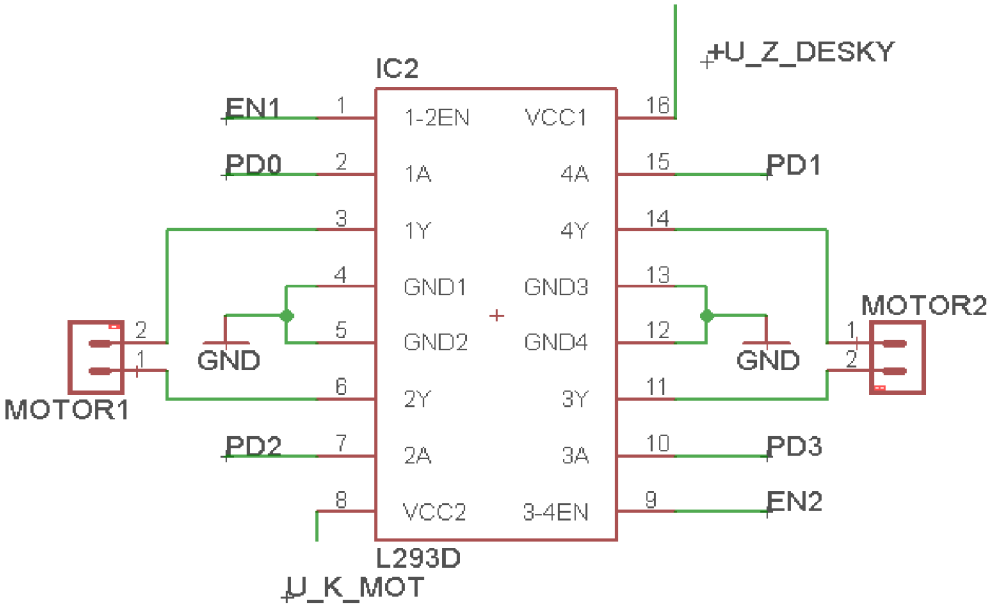


Ilustrace 17: Schéma zapojení napájecí části.



Ilustrace 18: Zapojení jumperu.

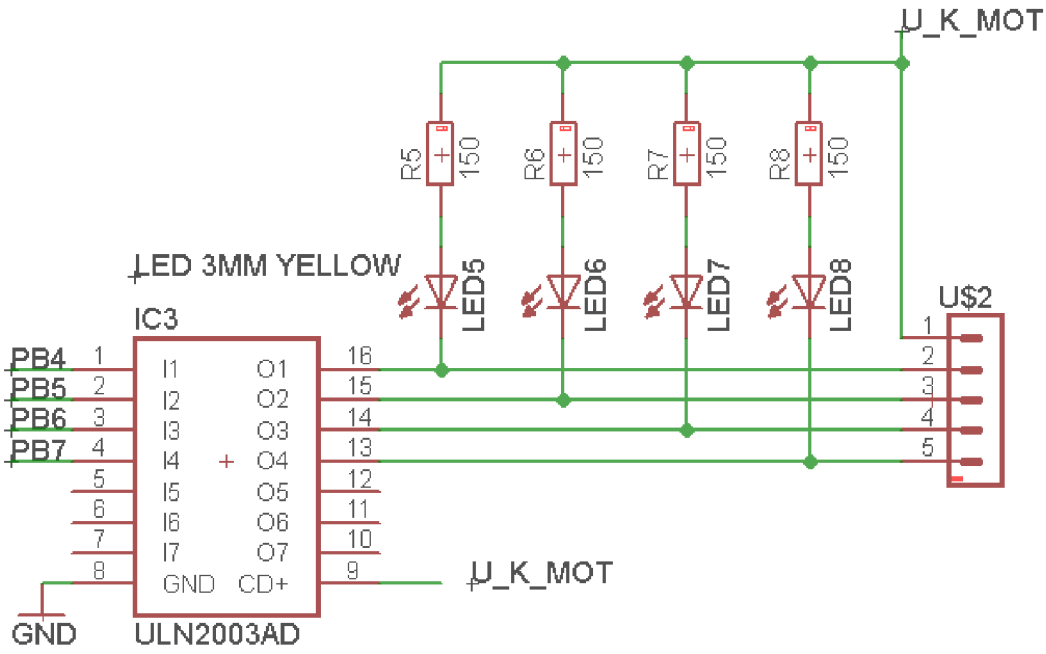
4.2 Zapojení budičů



Ilustrace 19: Zapojení L293D.

L293D na ilustraci 19 v pouzdře DIP16 je použit pro řízení stejnosměrných motorů. Přivedení logické 1 na vstup 1-2EN povolíme ovládání motoru 1. Přivedení logické 1 na vstup 1A nebo 2A můžeme motor uvést do pohybu buď po směru nebo proti směru hodinových ručiček. Analogicky funguje i druhá část budiče, kde motor 2 ovládáme pomocí vstupů 3A a 4A a povolení je pomocí 3-4EN. Budiče je standardně napájen vstupem VCC1 přivedením 5V z konektoru MLW20. Přivedením napájení na vstup VCC2 je možno napájet motor, který má vyšší odběr proudu. K tomu je určena napájecí část.

ULN2003 jsou použity pro řízení krokových motorů. Celkem jsou použity dva integrované obvody. Přivedením logické 1 na vstupy I1 – I4 je možno spínat jednotlivé fáze krokových motorů a tím pádem je ovládat. Budič je napájen na vstupu CD+, avšak samotný motor má napájení přivedeno přímo na rozdíl od L293D. Při sepnutí jednotlivých fází se rozsvítí příslušná LED dioda. Zapojení viz. Ilustrace 20.



Ilustrace 20: Zapojení ULN2003.

4.3 Základová deska

K ovládání výukového modulu je využívána základová deska od firmy PK design.

- Je osazena mikrokontrolérem ATmega128.
- Napájecí napětí pro mikrokontrolér a pro připojené moduly je +5V
- Mikrokontrolér se programuje pomocí Isp nebo JTAG programátorem připojeným přes daný konektor
- Všechny vstup/výstupní vývody Mcu jsou přístupné na popsaných konektorech, na které je možno připojit další moduly
- Má vestavěné periférie, které je možné odpojit a konfigurovat pomocí propojek.
- Zdrojem hodinového signálu je krystal 14.7456MHz, avšak může být použit i externí zdroj.
- Obsahuje dvě odpojitelné sériové rozhraní 1x RS-232 a 1x USB. Mikrokontrolér je možné resetovat tlačítkem RESET. [10]

4.4 Modul klávesnice s LCD displejem

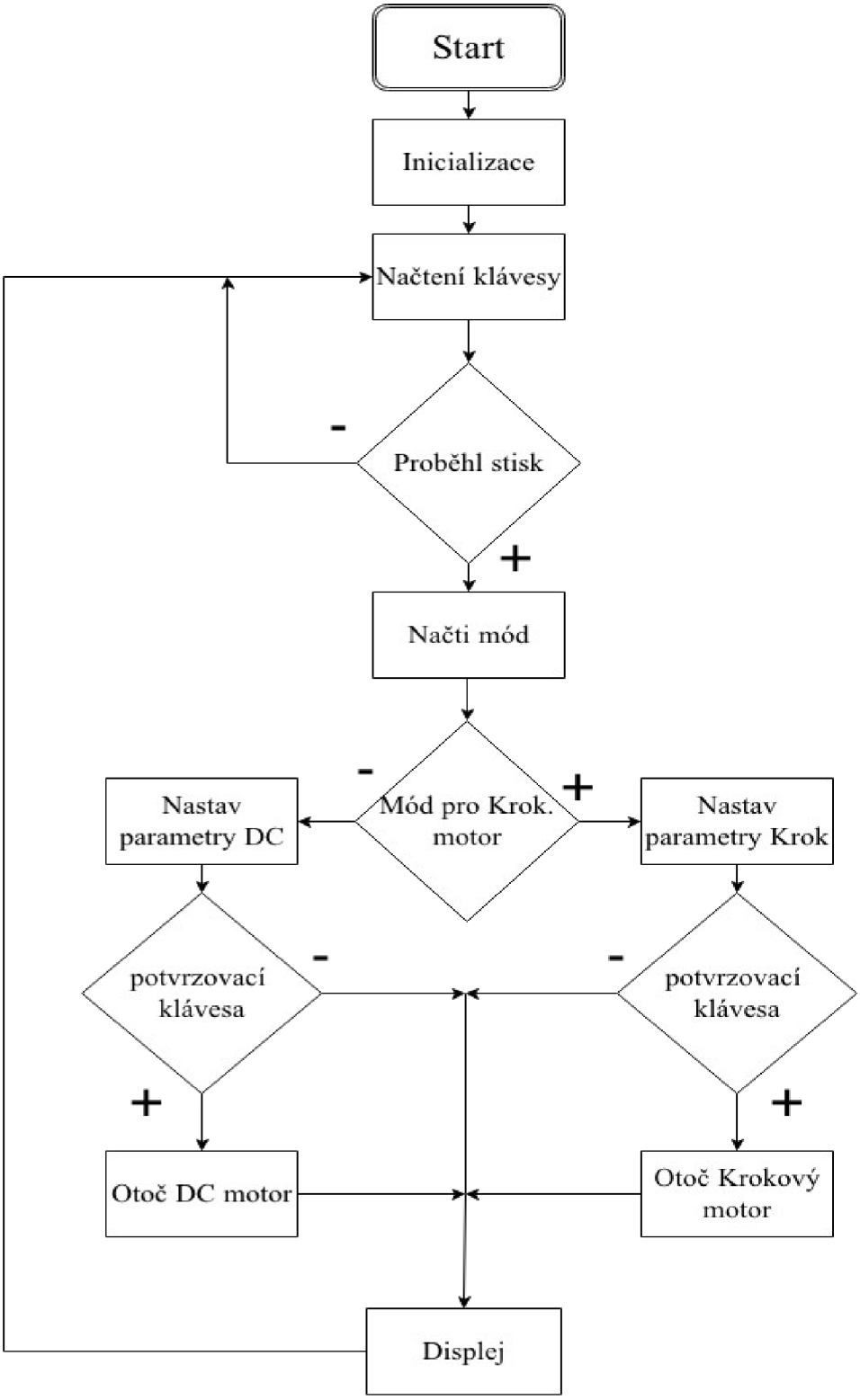
Pomocí tohoto modulu je možné nastavit rychlost a směr otáčení DC motoru. Dále je možno nastavit úhel natočení, rychlost a způsob řízení krokového motoru.

- Obsahuje 4 místný multiplexovaný LED displej, rozhraní pro LCD displej.
- V použitém modulu je displej 2x16 znaků.

Obsahuje maticovou klávesnici složenou ze 16 tlačítek a piezoelektrický měnič bez vlastního oscilátoru, pro generování zvuků. [11]

5 SOFTWAROVÁ ČÁST

Program pro procesor byl napsán v jazyce C v prostředí Atmel Studio 6.1, který byl zároveň použit jako kompilátor a Codevision 3.1 k nastavení Timeru1. Jak bylo zmíněno výše, motory jsou řízeny mikrokontrolérem ATmeag128, který byl naprogramován přes ISP rozhraní přímo na základové desce.



Ilustrace 21: Vývojový diagram hlavní části programu.

V ilustraci 21 je schématicky popsána hlavní funkce *main*. Nejprve se zavolá funkce pro počáteční inicializaci. Zde jsou nastaveny porty jako výstupní, uvítací obrazovka, nastavení timeru1 a přerušení.

Následně se provede načtení klávesy do proměnné *but*. Pokud dojde ke stisku, proměnná nabude odlišné hodnoty a proběhne indikace stisku klávesy. Pokud ne, cyklus se vrátí zpět k načtení klávesy.

Po načtení nové klávesy proběhne nejprve načtení zvoleného módu. Pokud nedojde k žádné změně módu, zůstává v proměnné *mode* defaultní nastavení. Pomocí této proměnné se určí, pro jaký motor se budou nastavovat hodnoty a které konkrétní parametry může aktuálně uživatel měnit. Pro krokový motor se nastavuje směr otáčení, způsob řízení, počet cyklů a rychlost provedení cyklu. U Stejnosměrného motoru se nastavuje směr otáčení a rychlost otáčení. Všechny tyto hodnoty jsou průběžně zobrazovány na LCD displeji. Pokud dojde ke stisku potvrzovací klávesy, provede se zadaný úkon, podle zvolených parametrů.

5.1 Funkce Keyboard

Tato funkce slouží pro čtení z maticové klávesnice 4x4 což nám umožňuje zadat 16 různých znaků. Základní princip spočívá detekci stisku nejprve po sloupcích a pak dle řádku, čímž je jednoznačně detekován stisk konkrétního tlačítka. Obecně tato funkce vrací hodnotu typu char a nemá žádné vstupní parametry.

```
char Keyboard()
{
     const int time = 500;
     unsigned char buttons = 255;
     old_ddr = KbdDDR;                  //zaloha nastaveni
DDR
     _delay_us(time);
     setb(KbdDDR,0);                    // vystupy
     setb(KbdDDR,1);
     setb(KbdDDR,2);
     setb(KbdDDR,3);
     clrb(KbdDDR,4);                    // vstupy
     clrb(KbdDDR,5);
     clrb(KbdDDR,6);
     clrb(KbdDDR,7);
     KbdPORT = 0b00001110;

     // 1 sloupec
     _delay_us(time);
     if bit_is_clear(KbdPIN,4) buttons = 1;
     if bit_is_clear(KbdPIN,5) buttons = 4;
     if bit_is_clear(KbdPIN,6) buttons = 7;
     if bit_is_clear(KbdPIN,7) buttons = 100;          // #
     // 2 sloupec
     KbdPORT=0b00001101; _delay_us(time);
     if bit_is_clear(KbdPIN,4) buttons = 2;
     if bit_is_clear(KbdPIN,5) buttons = 5;
     if bit_is_clear(KbdPIN,6) buttons = 8;
     if bit_is_clear(KbdPIN,7) buttons = 0;
```

```
    // 3 sloupec
    KbdPORT=0b00001011;
    _delay_us(time);
    if bit_is_clear(KbdPIN,4) buttons = 3;
    if bit_is_clear(KbdPIN,5) buttons = 6;
    if bit_is_clear(KbdPIN,6) buttons = 9;
    if bit_is_clear(KbdPIN,7) buttons = 101;          // *
    // 4 sloupec
    KbdPORT=0b00000111;
    _delay_us(time);
    if bit_is_clear(KbdPIN,4) buttons = 11;           //F1
    if bit_is_clear(KbdPIN,5) buttons = 12;           //F2
    if bit_is_clear(KbdPIN,6) buttons = 13;           //F3
    if bit_is_clear(KbdPIN,7) buttons = 14;           //F4

    KbdPORT = 0b00001111;          // nectu
    KbdDDR = old_ddr;              //zaloha nastaveni DDR
    return buttons;
}
```

Nejprve jsou nastaveny proměnné *time* a *buttons*. Do proměnné *old_ddr* je zálohováno nastavení DDR. Proběhne zpoždění 500 µs. Po zpoždění jsou nastaveny jednotlivé bity DDR registru na vstupní pomocí funkce *clrb* a na výstupní pomocí *setb*.

Následně jsou kontrolovány jednotlivé sloupce bit po bitu. To je provedeno následujícím způsobem. Port je nastaven na 0b00001110, čímž je zvolen první sloupec. Nula vpravo určuje zvolený sloupec což znamená, že s dalšími sloupci se posouvá nula doleva. Dojde ke zpoždění a poté jsou kontrolovány jednotlivé řádky sloupce. Pokud je v některém řádku bit roven 0, znamená to, že na daném tlačítku došlo ke stisku. Tím pádem se přiřadí zvolená hodnota do proměnné *buttons*. Stejným způsobem fungují i ostatní sloupce. Jakmile jsou zkontrolovány všechny sloupce a řádky, je port nastaven na 0b00001111, což znamená, že žádný sloupec není zvolen. Následně se do registru DDR nahraje staré nastavení a nakonec celá tato funkce vrátí hodnotu stisknutého tlačítka. Jak je patrné z kódu, tato funkce neumožňuje zaznamenat stisk dvou kláves najednou. Kdyby k takovému stisku došlo, do proměnné *buttons* se nahraje pouze poslední zaznamenaná klávesa, nicméně pro naše demonstrační účely tato funkce postačuje.

5.2 Funkce Checkmode

Funkce slouží k nastavení konkrétního módu podle hodnoty stisknutého tlačítka. Vstupní parametry jsou *prev_mode* a *btn* typu *char*. Provede se příkaz Switch, který vrátí příslušnou hodnotu módu, podle stisknutého tlačítka. Pokud nebylo stisknuto ani jedno z příslušných tlačítek tzn. nebylo stisknuto tlačítko pro změnu módu, dojde k vrácení dříve zvoleného módu. Pokud uživatel nezvolí jinak, při prvním spuštění je defaultně nastaven první mód.

K nastavení globální proměnné *cycles* slouží první mód. Druhý slouží k nastavení rychlosti otáčení. Pomocí třetího a čtvrtého módu je možno měnit směr otáčení krokového motoru. Čtvrtý a pátý je určen k změně způsobu řízení krokového motoru. Poslední dva módy určují směr otáčení stejnosměrného motoru.

U všech těchto tlačítek je stisk doprovázen zvukovou signalizací. Ta je provedena ve funkci
beep.

```
char CheckMode(char prev_mode ,char btn)
{
     switch (btn)
     {
          case 11:  beep(1); return 0; break;      // cycl v
          case 12:  beep(1); return 1; break;      // spd v
          case 7:   beep(1); return 2; break;      // dir0
          case 8:   beep(1); return 3; break;      // dir1
          case 9:   beep(1); return 4; break;      // drmod0
          case 13:  beep(1); return 5; break;      // drmod1
          case 100: beep(1); return 6; break;      // DCdir0
          case 101: beep(1); return 7; break;      // DCdir1
          default:  return prev_mode;  break;
     }
}
```

5.3 Funkce Beep

Slouží k ovládání akustického měniče pro generování zvukového signálu. Je možno pomocí této funkce měnit tón a dobu trvání tónu. Jako vstupní parametr je zde proměnná *char* pomocí které je možno nastavit opakování zvukového signálu. V cyklu *for* se nejprve nastaví DDR registr jako výstupní. Další cyklus *for* obsahuje proměnnou *lenghTone*, která slouží k nastavení délky zvukového signálu. Uvnitř tohoto cyklu se neguje příslušný port po určitou dobu. Díky této negaci portu za určitou dobu v µs dojde ke změně tónu. Funkce *_delay_ms* slouží pouze jako přerušovač mezi jednotlivými tóny.

```
void beep(char rep)
{
     for (char l=0; l<rep; l++)
     {
          setb(BeepDDR,BeepPin);
          for (char k=0;k<lenghTone;k++)          //delka
          {
               negb(BeepPORT,BeepPin);
               _delay_us(Tone);                   //ton
          }
          _delay_ms(100);
     }
}
```

5.4 Funkce ButtonValue

Je určena k nastavení globálních proměnných pro rychlost, otáčení motorů a pro počet cyklů. Vstupní parametr je hodnota stisknutého tlačítka *btn*, podle které se zvolí příslušná operace a výstupní parametr této funkce je zvolená hodnota, která se později přičte popřípadě odečte od globální proměnné.

```
signed char ButtonValue(char btn)
{
     signed char valu = 0;
     switch (btn)
     {
          case 1: valu += BtnValue1; break;
          case 2: valu += BtnValue2; break;
          case 3: valu += BtnValue3; break;
          case 4: valu -= BtnValue1; break;
          case 5: valu -= BtnValue2; break;
          case 6: valu -= BtnValue3; break;
          break;
     }
     return valu;
}
```

5.5 Funkce pro otáčení stejnosměrných motorů

Funkce *TurningDC* je určena k otáčení stejnosměrných motorů. Obsahuje dva vstupní parametry *direct* typu char, pomocí kterého se volí směr otáčení a další parametr *speed* typu int, který je určen k nastavení rychlosti otáčení motoru.

Nejprve se příkazem *sei* povolí přerušení. Následně se testuje, jaká je hodnota v proměnné *direct*. Pokud je to nula, nastaví se port na pozici *DCIn1* na logickou 1 a logická 0 na pozici *DCIn2*. Toto způsobí otáčení motoru proti směru hodinových ručiček. Pokud hodnota *direct* není nula, pak se na pozici *DCIn1* nastaví logická 0 a na pozici *DCIn2* logická 1. Motor se bude otáčet ve směru hodinových ručiček. Poslední podmínka vypne otáčení motoru tím, že na oba dva vstupy přivede logickou 0. Je aktivní pokud je hodnota *speed* nastavena na nule.

```
void TurningDC(char direct, int speed)
{
     sei();
     if (!direct)
     {
          setb(DCPORT,DCIn1);
          clrb(DCPORT,DCIn2);
     }
     else if (direct)
     {
          setb(DCPORT,DCIn2);
          clrb(DCPORT,DCIn1);
     }
     if (!speed)
```

```
    {
        clrb(DCPORT,DCIn1);
        clrb(DCPORT,DCIn2);
    }
    ICR1 = ICRDC;
    OCR1A = speed *OCRDC;
```

Poslední dva příkazy jsou určeny pro nastavení hodnot PWM. Prvně se nastaví registr ICR1. Do tohoto registru se přiřadí hodnota, do které má čítač počítat. Během počítaní je výstup *DCenable* nastaven na logickou 1. Druhý příkaz slouží k nastavení registru s názvem OCR1A. Jakmile čítač dosáhne hodnoty OCR1A vynuluje výstup *DCenable*, jak je patrné z vektorů přerušení níže.

```
ISR(TIMER1_COMPA_vect)
{
    clrb(DCPORT,DCenable);
}

ISR(TIMER1_CAPT_vect)
{
    setb(DCPORT,DCenable);
}
```

5.6 Funkce pro otáčení krokového motoru

Pomocí funkce *StepperTurning* je možno otáčet krokový motor. Má čtyři vstupní parametry *drivemode* typu char, *direct* typu char, *cycl* typu int a *speed* typu int. *Drivemode* určuje jaký způsob řízení je zvolen. Proměnná *direct* určuje, jakým směrem se bude motor otáčet. *Cycl* ukládá počet cyklů, které je potřeba provést a pomocí proměnné *speed* je možno nastavit rychlost otáčení motoru.

Nejprve se rozhoduje, jaký způsob řízení je zvolen. Jestliže je *drivemode* roven nule, pak se jedná a čtyřtaktní řízení. Uvnitř této funkce se dále nastavuje směr otáčení. Pokud je *direct* roven nule bude se motor otáčet po směru hodinových ručiček, jinak se bude otáčet proti směru hodinových ručiček. To samé platí pro druhý způsob řízení. Jestliže je *direct* roven nule, zavolá se funkce *EightCycle* tzn. funkce pro otáčení ve směru hodinových ručiček, jinak se zavolá funkce *AntiEightCycle*. Zde se jedná o osmitaktní řízení.

```
void StepperTurning(char drivemode, char direct, int cycl,
int speed)
{
    if (!drivemode)
    {
        if (!direct)
        {
            Clockwise(cycl,speed);
        }
        else if (direct)
        {
            Anticlockwise(cycl,speed);
        }
    }
```

```
     if (drivemode)
     {
          if (!direct)
          {
               EightCycle(cycl,speed);
          }
          else if (direct)
          {
               AntiEightCycle(cycl,speed);
          }
     }
}
```

5.6.1 Funkce Clockwise

Funkce je určena k otáčení motoru ve směru hodinových ručiček. Má dva vstupní parametry *cycl* a *speed*, které jsou popsány výše. Funkci lze použít pro libovolný počet fází definováním proměnných *StepFirst* a *StepLast*.

```
void Clockwise(int cycl, int speed)
{
     int pos = StepFirst;                          //0
     int prev_pos = StepFirst - 1;                 //-1
     StepPORT = 0;
     for (int i = 0;cycl > i;i++)
     {
          if (pos == StepFirst)
          {
               setb(StepPORT,pos);                 //0
               clrb(StepPORT,StepLast);            //3
               pos++;
               prev_pos++;
          }
          else if (pos != StepFirst)
          {
               clrb(StepPORT,prev_pos);
               setb(StepPORT,pos);
               if (pos == StepLast)
               {
                    pos = StepFirst - 1;
                    prev_pos = StepFirst - 2;
               }
               pos++;
               prev_pos++;
          }
          Speed(speed);
     }
}
```

iterace	pos	prev_pos	Fáze 1	Fáze 2	Fáze 3	Fáze 4
0	0	-1	1	0	0	0
1	1	0	0	1	0	0
2	2	1	0	0	1	0
3	3	2	0	0	0	1

Tabulka 3: Průběh nastavení fází pro čtyřtaktní řízení.

Ukázka konkrétního průběhu čtyř iterací pro čtyři fáze viz. tabulka 3.

Nejprve se provede inicializace pozice a předchozí pozice. Proběhne vynulování celého portu. Proměnná *cycl* nastavuje počet iterací cyklu for.

Proměnná *pos* je nastavena na hodnotu 0 (z důvodu nultého bitu na portu) a tím pádem *prev_pos* na -1. Při prvním průchodu se splní první podmínka *(pos == StepFirst)*. Zde se nastaví bit na pozici *pos* a vynuluje bit na pozici *StepLast* (v počtu 4 fází je hodnota rovna 3). Dojde k inkrementaci proměnných *pos* a *prev_pos*.

Při dalších iteracích se splní podmínka *(pos != StepFirst)* a dochází zde k vynulování bitu na pozici *prev_pos* a nastavení bitu na pozici *pos*. Nakonec dojde k inkrementaci těchto dvou proměnných při každé iteraci.

Uvnitř je vložena podmínka *(pos == StepLast)*. Ta má za úkol nastavit proměnné *pos* a *prev_pos* do počátečních hodnot na konci iterace. Poslední se provede funkce *Speed*, která slouží k nastavení zpoždění mezi jednotlivými iteracemi a v konečném důsledku tak řídí rychlost otáčení motoru.

5.6.2 Funkce Anticlockwise

Slouží k otáčení motoru proti směru hodinových ručiček. Stejně jako funkce *clockwise,* má dva vstupní parametry *cycl* a *speed* a je možno ji použít pro libovolný počet fází.

```
void Anticlockwise (int cycl, int speed)
{
     int pos = StepLast;                              //3
     int prev_pos = StepLast + 1;                     //4
     StepPORT = 0;
     for (int i = 0;cycl > i;i++)
     {
          if (pos == StepLast)                        //3
          {
               setb(StepPORT,pos);
               clrb(StepPORT,StepFirst);              //0
               pos--;
               prev_pos--;
          }
          else if (pos != StepLast)
          {
               clrb(StepPORT,prev_pos);
               setb(StepPORT,pos);
               if (pos == StepFirst)
               {
```

```
                    pos = StepLast + 1;          //4
                    prev_pos = StepLast + 2;     //5
               }
               pos--;
               prev_pos--;
          }
          Speed(speed);
     }
}
```

iterace	pos	prev_pos	Fáze 1	Fáze 2	Fáze 3	Fáze 4
0	3	4	0	0	0	1
1	2	3	0	0	1	0
2	1	2	0	1	0	0
3	0	1	1	0	0	0

Tabulka 4: Průběh nastavení fází pro čtyřtaktní řízení.

Ukázka konkrétního průběhu čtyř iterací pro čtyři fáze viz. tabulka 4.

Stejně jako u funkce *Clockwise* se provede inicializace proměnných, avšak nyní na hodnoty *StepLast* a *StepLast* +1. Funkce pracuje obdobně jako předchozí funkce, avšak dochází zde k dekrementaci a podmínky pracují více s proměnnou *StepLast*.

Nejprve je splněna první podmínka *(pos == StepLast)*. Nastaví bit na pozici *pos* a vynuluje bit na pozici *StepFirst*. Poté dekrementuje proměnné *pos* a *prev_pos*.

Následující iterace probíhají za splnění podmínky *(pos != StepLast)*. Provede se vynulování bitu na pozici *prev_pos* a nastavení bitu na pozici *pos* a následná dekrementace těchto proměnných.

Tak jako u předešlé funkce je tu kontrolní podmínka nyní ve tvaru (pos == StepFirst). Slouží k nastavení proměnných do defaultních hodnot. Stejně tak funkce *Speed* funguje jak v předešlé funkci.

5.6.3 Funkce Eightcycle

Je určena k otáčení motoru po směru hodinových ručiček pro osmitaktní řízení. Tak jako předešlé funkce obsahuje dva vstupní parametry *cycl a speed* a je možno ji použít pro libovolný počet fází.

```
void EightCycle(int cycl, int speed)
{
     char pos = StepFirst;                       //0
     char prev_pos;                              //-1

     StepPORT = 0;
     for (int i = 0;cycl > i;i++)
     {
          if (pos == StepFirst)
          {
               prev_pos = StepFirst - 1;         //-1
               setb(StepPORT,StepFirst);         //0
               clrb(StepPORT,StepLast);          //3
               pos++;
```

```
            prev_pos++;
        }
        else if (!(i % 2))                          //sudá
        {
            clrb(StepPORT,prev_pos);
            prev_pos++;
        }
        else if (i % 2)                             //lichá
        {
            if (pos != (StepLast + 1))              //4
            setb(StepPORT,pos);
            else
            {
                pos = StepFirst - 1;                //-1
                setb(StepPORT,StepFirst);           //0
            }
            pos++;
        }
        Speed(speed);
    }
}
```

iterace	pos	prev_pos	Fáze 1	Fáze 2	Fáze 3	Fáze 4
0	0	-1	1	0	0	0
1	1	0	1	1	0	0
2	2	0	0	1	0	0
3	2	1	0	1	1	0
4	3	1	0	0	1	0
5	3	2	0	0	1	1
6	4	2	0	0	0	1
7	4	3	1	0	0	1

Tabulka 5: Průběh nastavení fází pro osmitaktní řízení.

Ukázka konkrétního průběhu sedmi iterací pro čtyři fáze viz. tabulka 5.

Jako u předešlých funkcí dojde k inicializaci proměnných *pos* a *prev_pos* a vynulování portu. Proměnná *pos* je nastavena na hodnotu *StepFirst* a *prev_pos*, která je inicializována později.

Při první iteraci je splněna podmínka *(pos == StepFirst)*. Zde se nastaví *prev_pos* na *StepFirst* -1, následně se nastaví bit na pozici *StepFirst* a vynuluje na pozici *StepLast*.

Jakmile se nastaví iterace na liché číslo je splněna podmínka *(i %2)*. Zde se nejprve testuje, jestli platí podmínka *(pos != StepLast +1)*. Pokud ano, dojde k nastavení bitu na pozici *pos*. Pokud ne, dojde k nastavení bitu na pozici *StepFirst* a do proměnné *pos* se nastaví defaultní hodnota na konci této iterace.

Pokud je iterace sudá dojde ke splnění podmínky (!(i % 2)). Zde dojde k vynulování bitu na pozici *prev_pos* a následně je tato proměnná inkrementována.

5.6.4 Funkce AntiEightCycle

Pomocí této funkce je možno otáčet motor proti směru hodinových ručiček pro osmitaktní řízení. Vstupní parametry jsou stejné jako u předchozích funkcí.

```
void AntiEightCycle(int cycl, int speed)
{
     int pos = StepLast;
     char prev_pos = StepLast + 1;
     StepPORT = 0;
     for (int i = 0;cycl > i;i++)
     {
          if (pos == StepLast)
          {
               prev_pos = StepLast + 1;
               setb(StepPORT,pos);
               clrb(StepPORT,StepFirst);          //0
               pos--;
               prev_pos--;
          }
          else if (!(i % 2))                      //sudá
          {
               clrb(StepPORT,prev_pos);
               prev_pos--;
          }
          else if (i % 2)                         //lichá
          {
               if (pos != (StepFirst - 1))
               setb(StepPORT,pos);
               else
               {
                    pos = StepLast + 1;           //4
                    setb(StepPORT,StepLast);      //3
               }
               pos--;
          }
          Speed(speed);
     }
}
```

iterace	pos	prev_pos	Fáze 1	Fáze 2	Fáze 3	Fáze 4
0	3	4	0	0	0	1
1	2	3	0	0	1	1
2	1	3	0	0	1	0
3	1	2	0	1	1	0
4	0	2	0	1	0	0
5	0	1	1	1	0	0
6	-1	1	1	0	0	0
7	3	0	1	0	0	1

Tabulka 6: Průběh nastavení fází pro osmitaktní řízení.

Ukázka konkrétního průběhu sedmi iterací pro čtyři fáze viz. tabulka 6.

Na začátku zde také dochází k inicializaci proměnných *pos* na hodnotu *StepLast* a *prev_pos* na hodnotu *StepLast* + 1 a dojde k vynulování portu.

Při první iteraci se splní podmínka *(pos == StepLast)*. Uvnitř se naplní *prev_pos* na hodnotu *StepLast* +1. Dále se se nastaví bit na pozici *pos* a vynuluje bit na pozici *StepFirst*. Nakonec dojde k dekrementaci proměnných *pos* a *prev_pos*.

Při další iteraci se splní podmínka lichosti *(i % 2)*. Uvnitř se nejprve zkontroluje, zda došlo k splnění podmínky *(pos != (StepFirst -1))*. Pokud ano, nastaví se bit na pozici *pos*. Pokud ne, proměnná *pos* se nastaví na defaultní hodnotu na konci cyklu a port na pozici *StepLast*.

Při sudé iteraci se splní podmínka (i % 2) a dojde k vynulování bitu na pozici *prev_pos* a následné dekrementaci této proměnné.

6 ZÁVĚR

Ke splnění cíle bakalářské práce bylo potřeba nejdříve provést rozbor problematiky motorů a příslušných integrovaných obvodů včetně mikrokontroléru. Popsány jsou zde způsoby řízení stejnosměrných a krokových motorů. U krokových motorů jsou zde uvedeny způsoby řízení čtyřtaktní, osmitaktní, mikrokování a řízení se sníženou energetickou zátěží.

Návrh schématu zapojení a samotného plošného spoje byl proveden v programu Eagle. Výroba byla provedena pomocí metody nažehlování a osazena z poloviny součástkami typu SMD z důvodu zmenšení modulu. Pro řízení motorů byl použit mikrokontrolér Atmega128, který je osazen na základové desce od firmy PKDesign. Také byl využit modul s LCD displejem a klávesnicí od této firmy, aby bylo možné nastavit rychlost a směr otáčení u stejnosměrných motorů a rychlost, počet cyklů, směr a způsob řízení u krokových motorů.

Dále je v práci řešen řídící program pro ovládání těchto motorů. Zdrojový kód byl napsán v jazyce C ve vývojovém prostředí Atmel studio a Codevision. Program je rozdělen do několika funkcí, které se zabývají konkrétními rutinami. Jsou zde funkce pro inicializaci, ovládání klávesnice, zobrazení hodnot na displej a v neposlední řadě pro ovládání stejnosměrného a krokového motoru. Pro ovládání rychlosti otáčení stejnosměrného motoru byla využita pulsně šířková modulace, která byla realizována pomocí časovačů. Funkce pro řízení krokových motorů umožňuje čtyřtaktní nebo osmitaktní řízení.

Cíle práce se podařilo splnit a vyrobený modul může být dále použit pro výuku mikroprocesorové techniky.

Kdybych měl modul měl za úkol tento modul realizovat znovu, zaměřil bych se více na návrh zapojení integrovaných. Přidal bych pull-up rezistory pro pro budící obvody. V programové části bych se snažil co nejvíce zredukovat globální proměnné, aby program zabíral co nejméně paměti.

7 SEZNAM POUŽITÉ LITERATURY

[1] HAMMER. *Elektrotechnika a elektronika: přednášky*. Vyd. 1. Brno: Akademické nakladatelství CERM, 2006. Učební texty vysokých škol (Vysoké učení technické v Brně). ISBN 80-214-3334-5.

[2] TKOTZ, Klaus. *Příručka pro elektrotechnika*. Vyd. 1. Praha: Europa-Sobotáles, 2002. ISBN 80-86706-00-1.

[3] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. Vyd. 1. Praha: BEN - technická literatura, 2004, 247 s. ISBN 80-730-0141-1.

[4] ŘEZÁČ, Kamil. Krokové motory. [online]. 2002 [cit. 2014-04-07]. Dostupné z: http://robotika.cz/articles/steppers/cs

[5] WILLIAMS, Al. *Microcontroller projects using the Basic Stamp*. 2nd ed. Lawrence, Kan.: CMP Books, c2002, xvi, 456 p. ISBN 15-782-0101-2.

[6] BROWN, Jim a Bob JORDAN. DPRG: Brief H-Bridge Theory of Operation. *DPRG Dallas Personal Robotics Group* [online]. 1998 [cit. 2014-05-04]. Dostupné z: www.dprg.org/tutorials/1998-04a/

[7] Datasheet search site. STMICROELECTRONICS. *ULN2003 datasheet* [online]. 2002 [cit. 2014-05-03]. Dostupné z: http://www.alldatasheet.com/view.jsp?Searchword=ULN2003

[8] Datasheet search site. TEXAS INSTRUMENTS. *L293D datasheet* [online]. Dallas, Texas, 1998 [cit.2014-05-03]. Dostupné z :http://www.alldatasheet.com/view.jsp Searchword=L293d%20datasheet

[9] ATMEGA128 datasheet. In: *>ALLDATASHEET.COM - Datasheet search site for Electronic Components and Semiconductors and other semiconductors* [online]. 2011 [cit. 2014-05-13]. Dostupné z: http://www.alldatasheet.com/view.jsp? Searchword=ATMEGA128

[10] PK Design - Atmel AVR, Xilinx CPLD, FPGA Development (EVM) Boards. *Základová deska MB-ATmega128 v4.0* [online]. 2008 [cit. 2014-05-14]. Dostupné z: http://www.pk-design.net/HtmlCz/MB_ATmega128v4.html

[11] PK Design - Atmel AVR, Xilinx CPLD, FPGA Development (EVM) Boards. *Modul LED a LCD displejů* [online]. 2005 [cit. 2014-05-14]. Dostupné z: http://www.pk-design.net/HtmlCz/Modules.html#DispModules

8 SEZNAM PŘÍLOH

8.1 CD s přiloženými soubory
8.1.1 Elektronická verze bakalářské práce
8.1.2 Zdrojový kód programu pro řízení motorů
8.1.3 Návrh schématu a plošného spoje
8.1.4 Firemní dokumentace použitých integrovaných obvodů