Západočeská univerzita v Plzni

FAKULTA PEDAGOGICKÁ

KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

STAVEBNICE MAT - MODUL KROKOVÉ MOTORY

BAKALÁŘSKÁ PRÁCE

Jan Král

Přírodovědná studia, obor Informatika se zaměřením na vzdělání

léta studia (2009 - 2012)

Vedoucí práce: *Ing. Petr Michalík, Ph.D.*

Mítov, 22. června 2012

Poděkování:

Na tomto místě bych rád poděkoval vedoucímu mé práce, panu Ing. Petru Michalíkovi, Ph.D., za cenné připomínky, náměty a za čas, který mi věnoval. Dále bych rád poděkoval panu Mgr. Tomáši Jakešovi za poskytnutí dodatečných materiálů pro tvorbu schémat.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

Plzeň, 22. červen 2012

……
vlastnoruční podpis

OBSAH

1 ÚVOD

Tématem této bakalářské práce je přídavný modul Krokový motor (Stepper motor) pro stavebnici MAT.

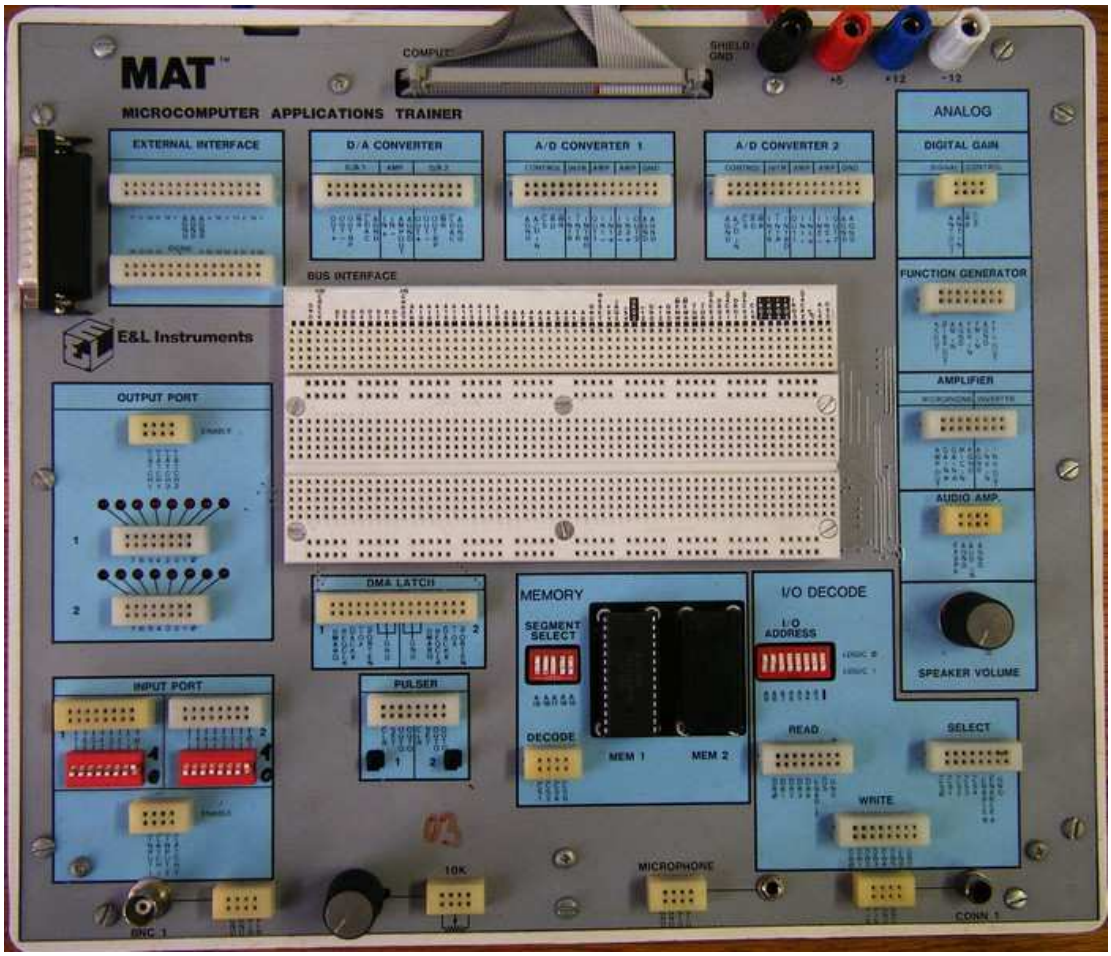
První část práce se zabývá teoretickým popisem přídavného modulu Krokový motor (dále KRM). Nejprve je zde krátce představena stavebnice MAT, jenž je nutná pro práci s přídavným modulem KRM. Druhá kapitola je věnována tomu, co krokový motor vlastně je, využití krokových motorů a definici kroku a jeho velikosti. Dále je zde popsán princip pohybu krokových motorů. Další kapitola je věnována samotnému modulu KRM. Zabývá se základním popisem částí modulu a signálů. Dále jsou podrobně vysvětleny principy ovládání modulu KRM. Poslední kapitola je věnována testovacímu a pomocnému programu, jenž byly pro práci s modulem KRM vytvořeny.

Druhá část práce je věnována sadě úloh pro modul KRM. Z hlediska programovacího prostředí stavebnice MAT je výhodný programovací jazyk Pascal. Práce obsahuje sadu sedmi úloh, které by měly pomoci pochopit principy fungování krokových motorů, možnosti propojení s jinými částmi stavebnice MAT a měly by pomoci s rozvojem technické představivosti. U všech úloh je formulováno zadání, úkoly a u každé úlohy je také příklad zdrojového kódu programu a popis fungování programu.

2 PŘÍDAVNÝ MODUL STAVEBNICE MAT - KROKOVÝ MOTOR

2.1 MAT

MAT je zkratkou z anglického názvu MICROCOMPUTER APPLICATION TRAINER. Jedná se o stavebnici určenou pro provádění experimentů v oblasti mikropočítačových technologií a řízení. Stavebnice MAT je připojena k počítači a jeho pomocí je také řízena. Stavebnice samotná obsahuje několik funkčních bloků, které lze propojovat a kombinovat a vytvářet tak různá zařízení. Mezi funkční bloky patří například A/D a D/A převodníky, vstupní porty s přepínači, výstupní porty s diodami, funkční generátor, bezzákmitová tlačítka, reproduktor a další. K MATu lze připojovat externí součástky, či další přídavné moduly. Mezi přídavné moduly patří například klávesnice, display, krokový motor a servomotor. Na FPE se stavebnice MAT využívá pro výuku předmětu Technika počítačů 2. Více informací o stavebnici MAT lze nalézt např. v diplomové práci Výukový materiál o počítačové stavebnici MAT [1].



Obr. 1 – Fotografie stavebnice MAT

2.2 KROKOVÝ MOTOR

2.2.1 CO JE KROKOVÝ MOTOR

Krokový motor je speciální druh motoru, který se neotáčí spojitě, jako je tomu u běžných motorů, ale pohybuje se po krocích. To znamená, že kdykoliv je k tomu vybuzen, pootočí se o přesně daný úhel. Rychlost otáčení krokových motorů je dána frekvencí řídicích impulzů. Díky svému nespojitému pohybu jej lze označit jako digitální motor. Samotný motor vyžaduje řídicí elektroniku, která řídí buzení motoru a tím jeho chod.

2.2.2 HISTORIE

První zmínky o krokovém motoru pochází z roku 1919 z Anglie a z roku 1920 z USA. První využití našly krokové motory ve vojenské technice. Například jako součást dálkového navádění torpéd. Ke komerčnímu využití dochází v šedesátých letech. Využití krokových motorů jde ruku v ruce s vývojem polovodičových technologií, nebo-li s rozvojem výpočetní techniky [2].

2.2.3 VYUŽITÍ KROKOVÝCH MOTORŮ

Krokové motory našly využití v mnoha oborech. Používají se v případech, kde je potřeba přesně řízené činnosti, například různá polohovací zařízení. Využívají se v letectví, v pohonech NC strojů, robotů, či manipulačních ramen. Také se využívají v počítačích, kde mohou zajišťovat pohon pevných disků, v tiskárnách a dalších periferních zařízení. V minulosti se používaly i pro polohování hlaviček pevných disků.

2.2.4 DEFINICE KROKU

Krokové motory, kdykoliv jsou k tomu vybuzeny, se otočí o daný úhel. Tento úhel se nazývá krok. Může být definován jako mechanická odezva motoru na jeden impulz řídicího obvodu. Jeho velikost je dána fyzickou konstrukcí motoru.

Obecně lze vypočítat velikost kroku podle vzorce [3]:

$$\alpha = \frac{360°}{m * N_r}$$

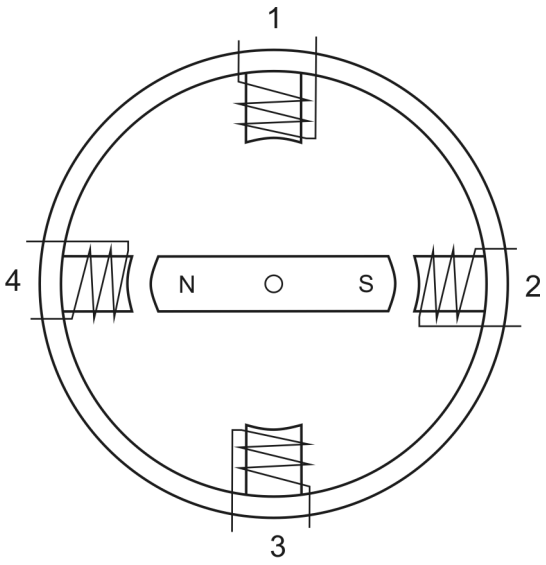
kde: m = počet fází statoru; N_r = počet zubů rotoru

2.2.5 PRINCIP POHYBU KROKOVÝCH MOTORŮ

KROKOVÝ MOTOR S AKTIVNÍM ROTOREM

Princip pohybu krokových motorů je poměrně jednoduchý. Pro názornost zde bude zjednodušeně vysvětlen na krokovém motoru s permanentním magnetem a čtyřmi cívkami. Toto je nejjednodušší typ krokového motoru, ale princip fungování zůstává obdobný pro všechny ostatní typy krokových motorů [4].

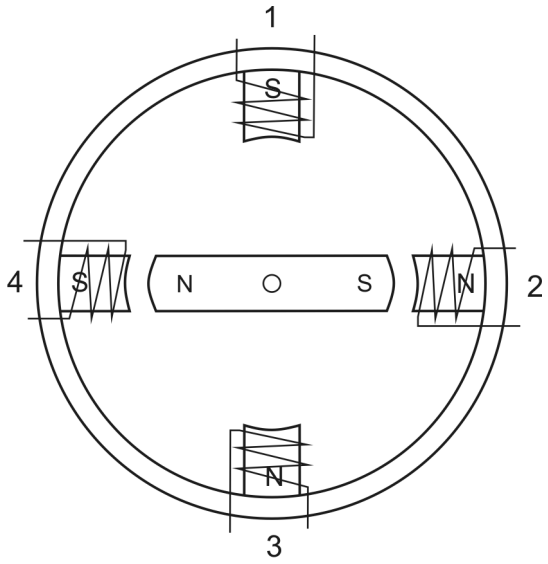
Stator tohoto motoru je tvořen čtyřmi cívkami navinutými na čtyři pólové nástavce. Každou cívku lze budit samostatně a cívky jsou vzájemně natočeny o 90°. Rotor je tvořen magnetem umístěným na ose rotoru (obr. 2).



Obr. 2 – Krokový motor s aktivním rotorem

Motor zůstává v klidovém stavu, dokud nezačne cívkami protékat budící proud.

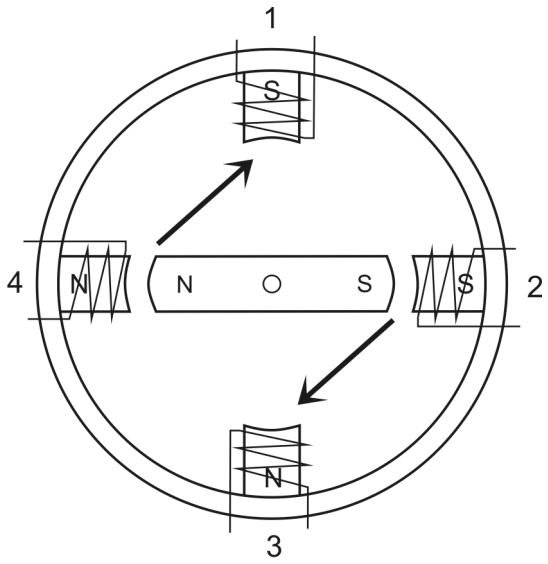
Buzením cívek tak, jak je vyobrazeno na obr. 3, bude motor pevně setrvávat ve své poloze, kvůli silně působícím přitažlivým silám na pólových nástavcích 4 a 2. Magnetické pole jižního pólového nástavce 4 přitahuje severní pól rotoru a magnetické pole severního pólového nástavce 2 přitahuje jižní pól rotoru.



Obr. 3 – Princip pohybu krokového motoru s aktivním rotorem a)

Působení pólů 1 a 3 vyvolává nepatrný točivý moment po směru hodinových ručiček, ale tato síla je zanedbatelná v porovnání se silovým působením pólů 4 a 2.

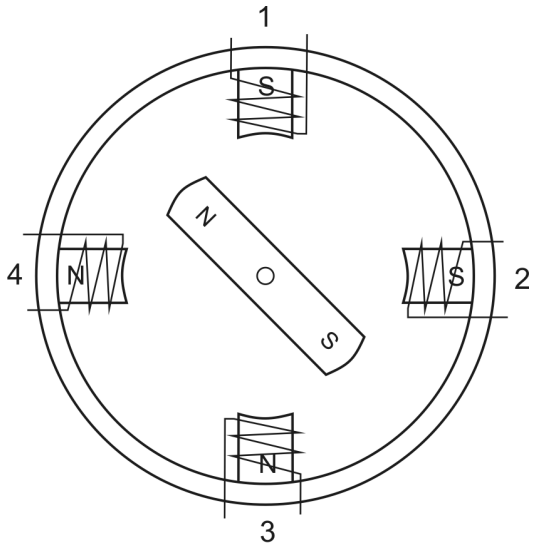
Změnou buzení cívek se změní také síly působící na rotor. Rotor se přiměje k otáčení po směru hodinových ručiček tak, že se změní orientace pólu 4 z jižní na severní a pólu 2 ze severní na jižní.



Obr. 4 – Princip pohybu krokového motoru s aktivním rotorem b)

Tato změna zruší silné přitažlivé síly na pólových nástavcích 4 a 2 a nechá působit síly na pólových nástavcích 1 a 3, což způsobí otáčení po směru hodinových ručiček.

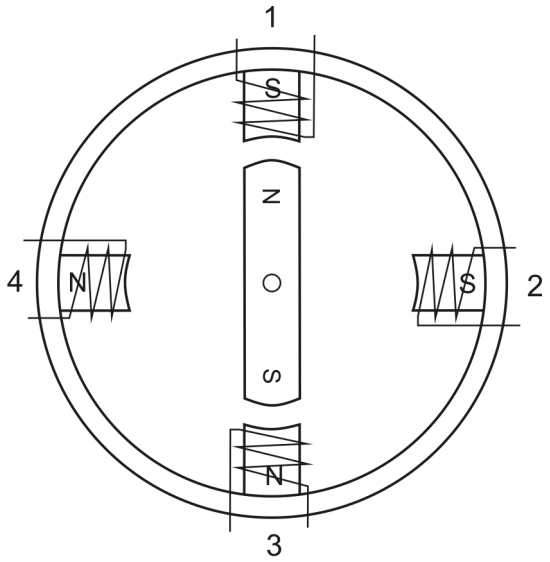
Motor se dal do pohybu působením silných odpudivých sil na pólových nástavcích 4 a 2. Dochází k výraznému zrychlenému pohybu rotoru směrem od těchto pólových nástavců po směru hodinových ručiček. Směr pohybu je dán přitažlivými silami působícími na pólových nástavcích 1 a 3.



Obr. 5 – Princip pohybu krokového motoru s aktivním rotorem c)

Přitažlivé síly na pólových nástavcích 1 a 3 rostou, protože se k nim rotor přibližuje, čímž se zrychluje otáčivý pohyb rotoru po směru hodinových ručiček.

Rotor pokračuje v otáčivém pohybu, dokud se nedostane do roviny shodné s rovinou pólových nástavců 1 a 3. V tomto bodě silné přitažlivé síly pólových nástavců 1 a 3 zastaví rotor v této poloze.



Obr. 6 – Princip pohybu krokového motoru s aktivním rotorem d)

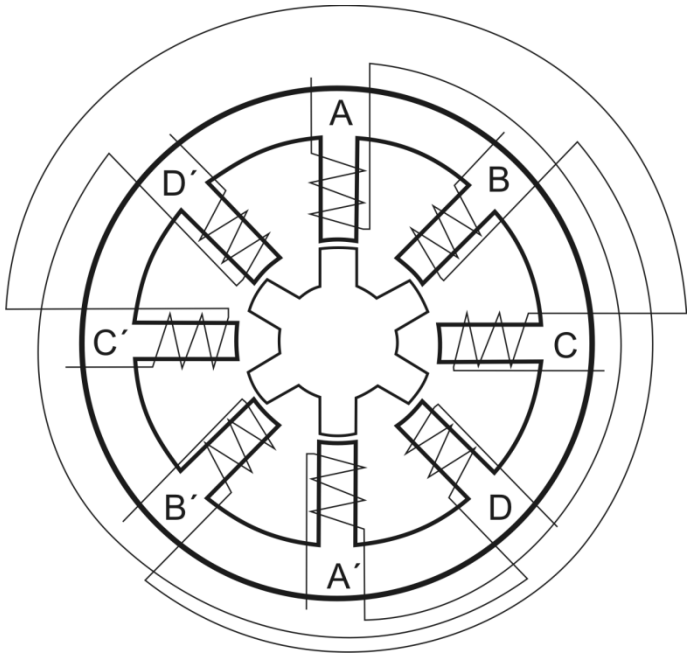
Ve skutečnosti rotor překmitne kvůli svému setrvačnému momentu, ale silné přitažlivé síly pólových nástavců 1 a 3 jej vrátí velmi rychle zpět do vyrovnané polohy.

Další krok a veškerá následující krokování motoru jsou opakováním toho, k čemu již došlo, pouze s různým nastavením buzení cívek v obdobném sledu.

KROKOVÝ MOTOR S PASIVNÍM ROTOREM

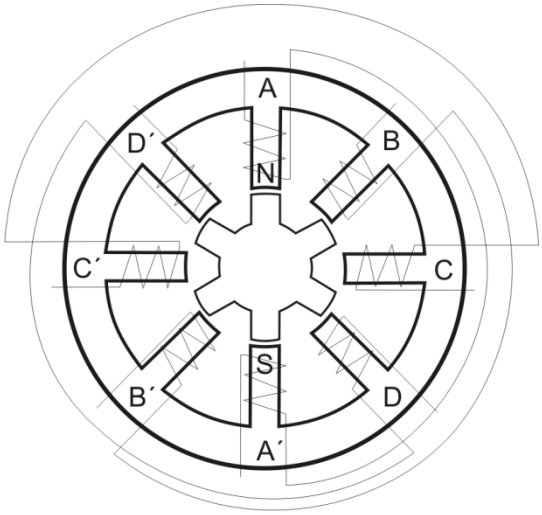
Princip pohybu krokového motoru s pasivním rotorem funguje na podobné principu jako u motoru s aktivním rotorem. Místo magnetických přitažlivých a odpudivých sil se zde využívá principu změn reluktance magnetického obvodu při otáčení rotoru [5]. Reluktance je magnetický odpor. Pro účely této práce stačí pochopit, že reluktance bude nejmenší v případě, kdy jsou pólové nástavce rotoru v jedné ose s pólovými nástavci vybuzené fáze. Jestliže v této poloze nejsou, vzniká moment, který otáčí rotorem. Stav nejmenší reluktance se nazývá stabilní magnetická poloha.

Na obr. 7 je reakční krokový motor se čtyřmi budícími fázemi [2]. Každá fáze je tvořena dvěma protilehlými cívkami, které jsou zapojeny do série a při buzení tvoří severní a jižní pól. Cívky jsou navinuty na statorových pólových nástavcích. Stator je tvořen svazkem ocelových plechů. Rotor má šest pólových nástavců tvořených z ocele či svazku ocelových plechů jako stator. Velikost kroku tohoto motoru je 15°.



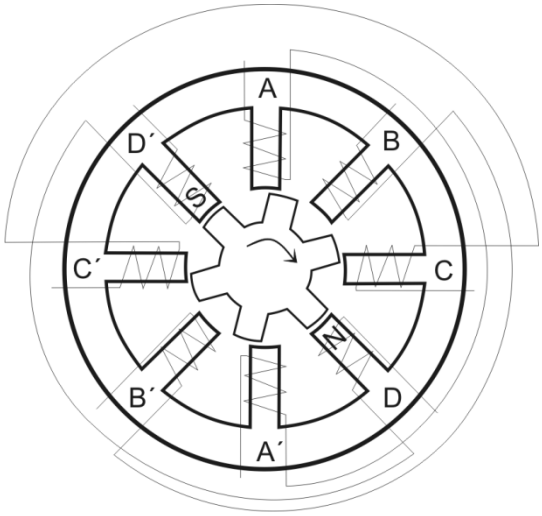
Obr. 7 – Krokový motor s pasivním rotorem

Při buzení fáze A se na pólovém nástavci A vytvoří severní pól a na nástavci A´ bude pól jižní (obr. 8). Rotor je v tuto chvíli ve stabilní magnetické poloze a nepůsobí na něj žádná síla, která by tuto stabilní polohu narušila.



Obr. 8 – Princip pohybu krokového motoru s pasivním rotorem a

Pro provedení kroku ve směru hodinových ručiček je nutné aktivovat buzení fáze D. V této chvíli je důležitá poloha rotoru vůči pólovému nástavci D a D´. Nástavce rotoru a nástavce pólové dvojice D-D´ se částečně překrývají. Dojde ke změně působení sil a rotor se pootočí tak, aby nalezl novou stabilní magnetickou polohu.



Obr. 9 – Princip pohybu krokového motoru s pasivním rotorem b

Nyní je motor ve stabilní magnetické poloze ve fázi D. Pro další krok se bude vše opakovat jako při buzení fáze D ale tentokrát pro fázi C. Pak bude následovat buzení fáze B a po ní fáze A, načež se bude celý cyklus buzení fází opakovat.

2.2.6 TYPY KROKOVÝCH MOTORŮ

KROKOVÝ MOTOR S AKTIVNÍM ROTOREM

Jako krokové motory s aktivním rotorem označujeme motory, jenž mají rotor tvořen permanentními magnety. Póly rotoru jsou tvořeny různým uspořádáním permanentních magnetů v rotoru.

Princip pohybu je založen na působení magnetických sil.

Jejich výhodou oproti motorům s pasivním rotorem je ta, že mají větší krouticí moment. Motory s aktivním rotorem také vytváří tzv. klidový moment, i když motor není napájen.

Obecně má tento typ krokových motorů větší velikost kroků. Obvyklé velikosti kroku jsou ve stupních až v desítkách stupňů.

KROKOVÝ MOTOR S PASIVNÍM ROTOREM

Krokové motory s pasivním rotorem jsou obvykle označovány jako reluktanční či reakční motory. Rotor i stator motoru jsou obvykle složeny z ocelových plechů. Plechy rotoru tvoří pólové nástavce. Stator má tří až pěti fázové buzení. Fáze statoru je tvořena dvěma protilehlými pólovými nástavci zapojenými do série.

Princip pohybu je založen na změně reluktance (magnetického odporu).

Krokové motory s pasivním jádrem mají relativně malou velikost kroku. Obvyklé velikosti jsou od desetin stupně po jednotky stupňů.

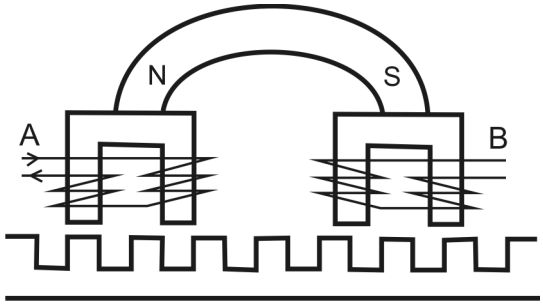
HYBRIDNÍ KROKOVÝ MOTOR

Hybridní krokový motor je kombinací dvou předešlých motorů. Rotor je zde tvořen permanentním magnetem kruhového tvaru nasunutým na ose z nemagnetického materiálu. Na pólech magnetu jsou nasazeny rotorové nástavce z měkkého železa, které na sobě mají drážky (zuby). Rotorové nástavce musí být nasazeny tak, aby drážky jednoho rotorového nástavce byly proti zubům druhého.

Velikost kroku bývá obdobná jako u krokových motorů s pasivním jádrem.

LINEÁRNÍ KROKOVÝ MOTOR

Jedná se o krokové motory, jež nerealizují otáčivý pohyb, ale pohyb lineární. To znamená, že se motor může pohybovat dvěma směry v jedné ose. Princip je obdobný jako u krokových motorů s pasivním rotorem.

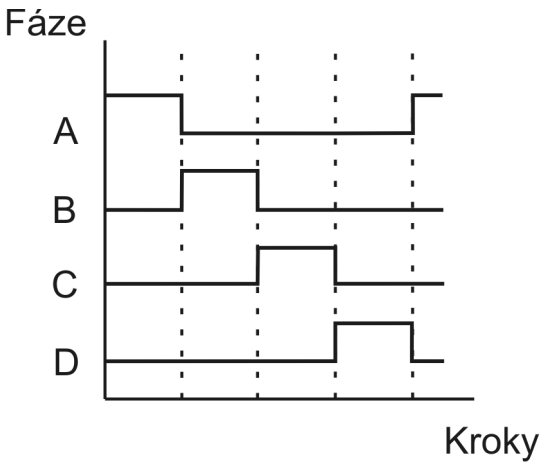


Obr. 10 – Lineární krokový motor

2.2.7 TYPY ŘÍZENÍ KROKOVÝCH MOTORŮ

ČTYŘTAKTNÍ ŘÍZENÍ S MAGNETIZACÍ JEDNÉ FÁZE

Jedná se o nejjednodušší způsob řízení krokových motorů. Dochází k postupnému buzení čtyř fází. Vždy je vybuzena pouze jedna fáze.

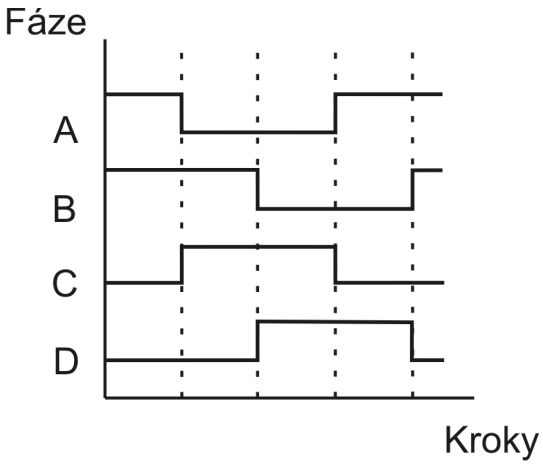


Obr. 11 – Čtyřtaktní řízení s magnetizací jedné fáze

Sekvence fází je následující: A – B – C – D

ČTYŘTAKTNÍ ŘÍZENÍ S MAGNETIZACÍ DVOU FÁZÍ

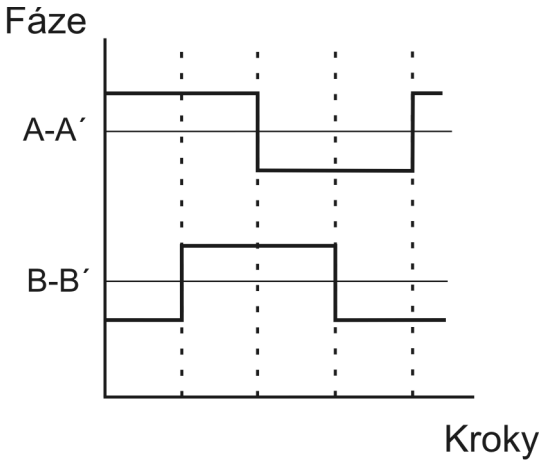
V tomto způsobu řízení jsou napájeny vždy dvě sousední fáze. Stabilní magnetická (klidová) poloha je mezi fázemi, nebo-li mezi vybuzenými pólovými nástavci.



Obr. 12 – Čtyřtaktní řízení s magnetizací dvou fází pro čtyřfázový krokový motor

Pro čtyřfázový krokový motor je sekvence fází následující: AB – BC – CD – DA

Jinak je tomu u dvoufázových hybridních krokových motorů. V tomto případě je kromě buzení fází ještě měněna jejich polarita. Místo čtyř fází jsou zde dvě fáze plus jejich obrácená polarita.



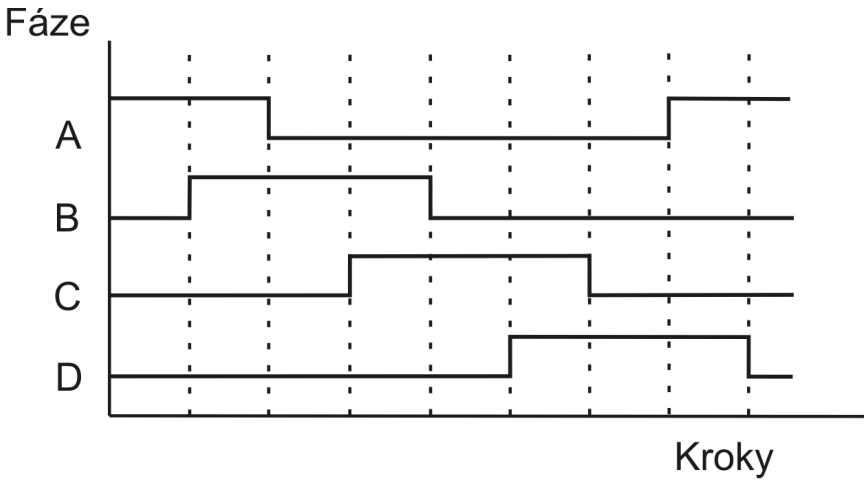
Obr. 13 – Čtyřtaktní řízení s magnetizací dvou fází pro dvoufázové hybridní motory

Pro dvoufázový hybridní krokový motor je sekvence fází následující:

AB´ – AB – BA´ – B´A´

OSMITAKTNÍ ŘÍZENÍ

Osmitaktní řízení vznikne složením čtyřtaktního řízení s magnetizací jedné a dvou fází. Dochází k postupnému střídání buzení jedné a dvou fází.



Obr. 14 – Osmitaktní řízení

Sekvence fází je následující: A – AB – B – BC – C – CD – D – DA

Při použití osmitaktního řízení bude u stejného krokového motoru dvojnásobný počet kroků a velikost kroku se zmenší na polovinu.

MIKROKROKOVÁNÍ

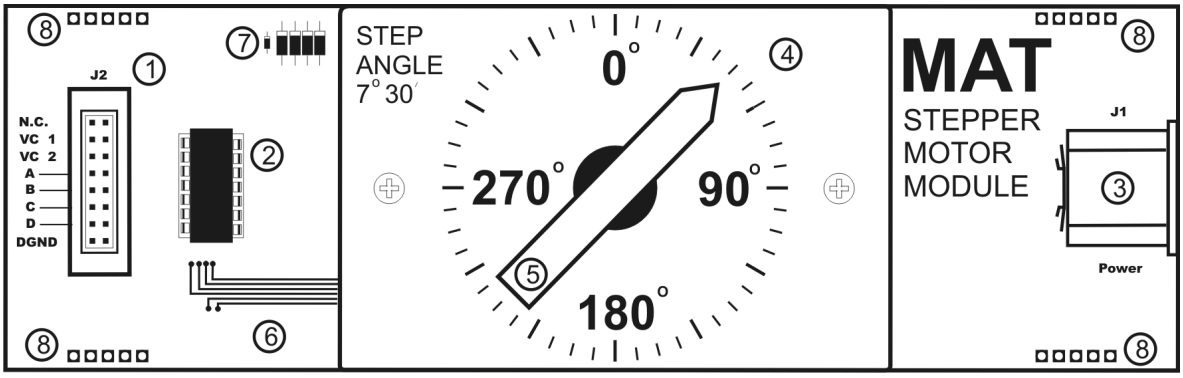
Mikrokrokování se používá v případech, kdy je potřeba použít krokování o velmi malých úhlech, například desetiny stupně. Tato metoda řízení krokových motorů vychází z magnetizace dvou fází. Při metodě magnetizace dvou fází jsou obě fáze napájeny stejně velkým proudem. V případě mikrokrokování mají proudy, které napájí fáze, rozdílnou velikost. Vhodnou volbou velikostí proudů lze dosáhnout libovolné rovnovážné polohy mezi dvěma pólovými nástavci. Touto metodou lze běžně zvětšit počet kroků na čtyřnásobek.

Dnes se používají třífázové motory, které díky mikrokrokování dosahují velmi jemných kroků. U nejmodernějších třífázových motorů se dosahuje až desítek tisíc kroků na otáčku [3].

2.3 MODUL KRM

Modul krokový motor je přídavným modulem ke stavebnici MAT. Jedná se o externí modul určený pro používání spolu se stavebnicí MAT. Využití modulu KRM bez stavebnice MAT je sice realizovatelné, ale pro naše účely zbytečné a komplikované. Modul KRM slouží pro provádění experimentů v oblasti řízení krokových motorů.

2.3.1 STRUČNÝ POPIS MODULU KRM



Obr. 15 – Modul KRM při pohledu shora

1. Konektor řídicích signálů – Slouží pro připojení signálů, jimiž se řídí činnost krokového motoru.
2. Integrovaný obvod – Součást řídicího obvodu.
3. Napájecí konektor – Slouží pro připojení napájecího adaptéru.
4. Ukazatel – Plechová destička s vyobrazením úhlové stupnice. Jeden dílek se rovná úhlu 7,5°.
5. Šipka ukazatele – Slouží jako ukazatel úhlu. Šipka je přímo připojená na osu krokového motorku. Otočením motoru se otočí také šipka.
6. Vodiče – Vodiče vedoucí do krokového motoru. Slouží k buzení krokového motoru.
7. Elektronické součástky – Součást řídicího obvodu.
8. Polohové konektory – Slouží pro připojení přídavného modulu KRM k nepájivému poli stavebnice MAT. Polohové konektory nevedou žádný signál.

   Všechny součásti jsou umístěny na desce plošných spojů.

Krokový motor je ve válcovém pouzdře umístěném mezi deskou plošných spojů a ukazatelem. Jedná se o krokový motor s aktivním rotorem se čtyřfázovým buzením.

2.3.2 ŘÍDICÍ SIGNÁLY

Konektor řídicích signálů umožňuje připojení osmi signálů.

N.C. – nezapojeno (No connect).

VC 1; **VC 2** – Používají se při experimentech s rychlostí otáčení krokového motoru. V této práce nejsou využity [4].

A až **D** – Řídicí signály, jejichž pomocí se řídí činnost krokového motoru.

DGND – Uzemnění. Na tento signál musí být přivedena zem ze stavebnice MAT, jinak modul KRM nebude fungovat.

2.3.3 ZÁKLADNÍ PŘIPOJENÍ MODULU KRM

Zde je základní funkční připojení modulu KRM ke stavebnici MAT. Modul je před připojením signálů nutné umístit na nepájivé kontaktní pole uprostřed stavebnice.

Tab. 1 – Tabulka zapojení modulu KRM ke stavebnici MAT

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1

Negované signály jsou značeny podtržítkem.

Modul KRM je napájen externím 12 V zdrojem. Tento zdroj je nutné před spuštěním připojit, jinak modul nebude fungovat. Externí zdroj se připojuje do konektoru J1, na obr. 14 značen číslicí 3.

2.3.4 PARAMETRY MODULU KRM

Krokový motor používaný v modulu KRM má:

- 12 pólových nástavců v horní polovině skříně
- 12 pólových nástavců v dolní polovině skříně
- 24 pólových nástavců ve středovém rámu

Všechny tyto pólové nástavce jsou navzájem stupňovitě uspořádané, takže tvoří kostru se 48 pólovými nástavci. Rotor má válcovitý tvar z magnetického materiálu, který je v osovém směru zmagnetizován dvanácti póly [4].

Pro buzení se používají čtyři fáze dle specifikací výrobce. Velikost kroku je 7,5°.

Obvod pro pohon modulu KRM využívá unipolární pohon místo bipolárního. Unipolární budí pouze 2 cívky současně a lze jej provozovat z jednoho proudového zdroje.

2.3.5 FÁZE BUZENÍ KROKOVÉHO MOTORU

Řízení modulu KRM je realizováno pomocí čtyř signálů, které lze reprezentovat čtyřmi bity. Tyto čtyři bity tvoří datové slovo v binární podobě. Pro aktivaci určitých pólových nástavců je potřeba předem dané kombinace signálů, nebo-li fáze. Tyto fáze jsou pevně dány výrobcem. U modulu KRM používáme čtyři fáze buzení (tzv. čtyřtaktní řízení s magnetizací jedné fáze).

Tab. 2 – Tabulka fází pro standardní zapojení modulu KRM

Fáze	Signál A	Signál B	Signál C	Signál D	DS (Binární)	DS (Dekadická)
1.	0	1	0	1	1010	10
2.	0	1	1	0	0110	6
3.	1	0	1	0	0101	5
4.	1	0	0	1	1001	9

Dle standardního zapojení zde signál A reprezentuje nejméně významový bit a naopak signál D reprezentuje nejvíce významový bit. Pokud se zapojení otočí a signál A bude jako nejvíce významový, vznikne nová tabulka budicích fází:

Tab. 3 – Tabulka fází pro alternativní zapojení modulu KRM

Fáze	Signál A	Signál B	Signál C	Signál D	DS (Binární)	DS (Dekadická)
1.	0	1	0	1	0101	5
2.	0	1	1	0	0110	6
3.	1	0	1	0	1010	10
4.	1	0	0	1	1001	9

V tomto případě by se používala datová slova v jiném pořadí. Ale směr pohybu by zůstal zachován. Z toho vyplývá, že fáze buzení krokového motoru jsou závislé na správnosti zapojení, tudíž je potřeba na toto dbát.

Tyto čtyři budicí fáze tvoří takzvanou krokovací sekvenci.

2.3.6 KROKOVACÍ SEKVENCE

Tento pojem se v této práci vyskytuje velmi často. Každá fáze je reprezentována datovým slovem. Lze tedy říci, že k aktivaci (buzení) daných pólových nástavců jsou použita určitá datová slova. Princip krokování, jak bylo uvedeno, spočívá v aktivaci po sobě jdoucích pólových nástavců. V našem případě se jedná o čtyři různé kombinace signálů, nebo-li čtyři datová slova. Pro další pohyb se datová slova opakují. Tato čtyři datová slova se nazývají krokovací sekvence.

V programech jsou použita datová slova v desítkové soustavě. Je mnohem jednoduší pracovat se čtyřmi zapamatovatelnými čísly v desítkové soustavě, než je zde zadávat v hexadecimální soustavě, či si je zapamatovávat v binární podobě. Takovýto zápis se snadno rozliší, je krátký, snadno zapamatovatelný a pochopitelný. Převod datového slova z desítkové soustavy do binární je automatický a programátor ani uživatel se s ním nemusí zabývat.

2.3.7 ŘÍZENÍ MODULU KRM

Vykonáním jedné fáze se krokový motor posune o jeden krok. Tudíž vykonáním čtyř po sobě jdoucích fází, neboli jedné krokovací sekvence, se posune o čtyři kroky, které se rovnají 30°. Pro otočení o 360° se musí vykonat 48 kroků. To je tedy 12 krokovacích sekvencí.

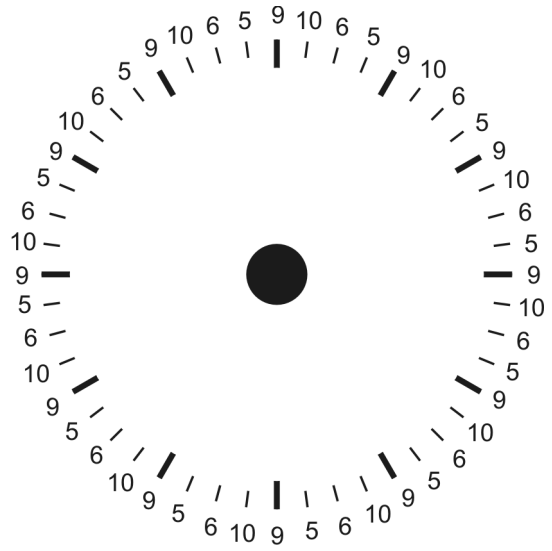
Směr otáčení krokového motoru je dán pořadím fází. Použijí-li se fáze tak, jak jsou definovány v tab. 2, motor se bude otáčet doprava, nebo-li ve směru hodinových ručiček. Pokud se pořadí fází otočí a krokování bude začínat čtvrtou fází, motor bude krokovat opačným směrem, to znamená proti směru hodinových ručiček. Převedením na krokovací sekvence vznikne sekvence 10, 6, 5, 9 pro pohyb doprava a sekvence 9, 5, 6, 10 pro pohyb doleva.

Pokud se budou střídat pouze dvě budicí fáze (datová slova), například 10 a 6, dojde k tomu, že motor bude vykonávat jeden krok doprava a jeden doleva. Dalo by se říci, že bude „kmitat“ mezi dvěma pólovými nástavci.

V případě buzení jedné a té samé fáze několikrát po sobě, nedojde k žádnému pohybu. Pokaždé se aktivuje stejný pólový nástavec a rotor zůstává ve stávající pozici.

Speciálním případem je postupné buzení fází ob jednu fázi. Například buzení fází 10 a po ní fáze 5. Jedná se o to, že krokový motor je nucen přeskočit jednu fázi (krok). Krokový motor by se musel posunout o dva pólové nástavce. Zde záleží na konstrukci motoru. Některé krokové motory tento skok provedou. Motor používaný v modulu KRM by tento skok udělat neměl. Testováním se zjistilo, že v tomto případě zůstane krokový motor ve své stávající pozici.

Pro názorné zobrazení jednotlivých fází je možné jednotlivým pólovým nástavcům přiřadit pevně daná datová slova, jak je tomu na obrázku níže.



Obr. 16 – Přiřazení datových slov pólovým nástavcům

2.3.8 ŘÍZENÍ S POLOVIČNÍM KROKEM

Jedná se o osmitaktní řízení obecně nazývané řízení s polovičním krokem. Princip osmitaktního řízení je popsán výše v kapitole 2.2.7 Typy řízení krokových motorů. V praxi jde o to, že krok není nejmenší úhel, o který se krokový motor může pootočit. V jedné z úloh je použito právě řízení s polovičním krokem.

Jedná se o případ, kdy jsou do standardní krokovací sekvence přidány půlkroky. Půlkrok vznikne vybuzením dvou sousedních pólových nástavců najednou, nebo-li aktivací dvou sousedních fází. Tím se docílí toho, že se krokový motor pootočí mezi tyto dva aktivované pólové nástavce. Velikost půlkroku, jak už název napovídá, je poloviční než velikost kroku. Takto se docílí zdvojnásobení počtu kroků a tím jemnějšího krokování, aniž by se musel upravovat krokový motor.

Řízení s polovičním krokem ale není možné použít vždy. Záleží na konstrukci motoru a řídicích obvodů.

Pro řízení s polovičním krokem se musí vytvořit speciální krokovací sekvence. Tato sekvence se sestaví tak, že mezi jednotlivé kroky se vloží půlkroky. Půlkroky jsou definovány jako logický součet sousedních fází. Takto se vytvoří nová tabulka s osmi fázemi (tab. 4).

Tab. 4 – Tabulka fází pro řízení s polovičním krokem

Fáze	Signál A	Signál B	Signál C	Signál D	DS (Binární)	DS (Dekadická)
1.	0	1	0	1	1010	10
2.	0	1	1	1	1110	14
3.	0	1	1	0	0110	6
4.	1	1	1	0	0111	7
5.	1	0	1	0	0101	5
6.	1	0	1	1	1101	13
7.	1	0	0	1	1001	9
8.	1	1	0	1	1011	11

S touto novou krokovací sekvencí se pracuje stejně jako s původní. Pouze se musí počítat s tím, že na stejný úhel je teď potřeba dvojnásobného počtu kroků.

2.4 POMOCNÉ PROGRAMY

Pro práci s modulem KRM byly vytvořeny a na CD přiloženy dva pomocné programy. Prvním pomocným programem je program TEST, který slouží k testování správnosti zapojení modulu KRM. Po zapojení modulu stačí tento program spustit a kontrolovat jeho činnost s informacemi vypisovanými na obrazovku.

Druhý program se nazývá DOKROKOVANI. Tento program je určen pro dokrokování šipky ukazatele na výchozí nulovou pozici. Po spuštění programu se stačí držet pokynů vypisovaných na obrazovku. Tento program využívá speciální krokovací sekvenci, pomocí níž motor dokrokuje na blízký vyšší násobek úhlu 30°. Pokud nedojde k přesnému zarovnání, nejedná se o chybu programu, ale o fyzické pootočení a je nezbytně nutné srovnat šipku ručně.

3 SADA ÚLOH PRO PŘÍDAVNÝ MODUL KROKOVÝ MOTOR

Sada úloh pro přídavný modul KRM obsahuje sedm základních úloh, po jejichž úspěšném naprogramování by měl student bez problémů zvládat práci s modulem KRM.

Jednotlivé úlohy jsou seřazeny od nejjednodušších po ty složitější. Určit složitost příkladu je ovšem poměrně obtížné. Některé příklady jsou náročnější na vytváření základních funkčních prvků, jiné příklady vyžadují větší pečlivost při tvorbě. Z hlediska principů práce s KRM lze považovat příklady za nepříliš obtížné. Základní princip řízení KRM je u všech příkladů stejný a zbytek je pouze otázkou technické představivosti a znalostí programování.

Základním předpokladem pro tvorbu příkladů je znalost programovacího jazyka Pascal a alespoň elementární znalost práce se stavebnicí MAT. U všech příkladů je nutné mít určitou úroveň technické představivosti.

U každého příkladu je zadání, kterého je nutné se držet. Zadáno je pouze to, co by měl každý program provádět, ale nikoliv jakým způsobem toho má být docíleno. Jelikož způsob, jakým program dosahuje požadovaných cílů, může být zcela individuální, je studentovi ponechána velká volnost při tvorbě programů.

Každá úloha má uvedeno řešení. Je zde příklad možného zdrojového kódu s popisem programu a popisem fungování po spuštění. Všechny úlohy jsou v přílohách na CD ve funkční podobě a po zapojení je možné jejich spuštění v programu Turbo Pascal.

3.1 ÚLOHA 1 – ZÁKLADY ŘÍZENÍ MODULU KRM

3.1.1 CÍL ÚLOHY

Hlavním cílem je pochopení základních principů fungování krokového motoru. Dílčím cílem je naučit se vytvářet základní programové struktury pro ovládání modulu KRM.

3.1.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky.
2. Prostudujte a vyzkoušejte přiložený program ULOHA_1_ZADANI.
3. Modifikujte předchozí program tak, aby se ukazatel mohl otáčet na obě strany. Směr otáčení zadá uživatel.
4. Opět upravte předchozí program tak, aby uživatel mohl měnit rychlost s jakou se krokový motor otáčí.

3.1.3 ZAPOJENÍ

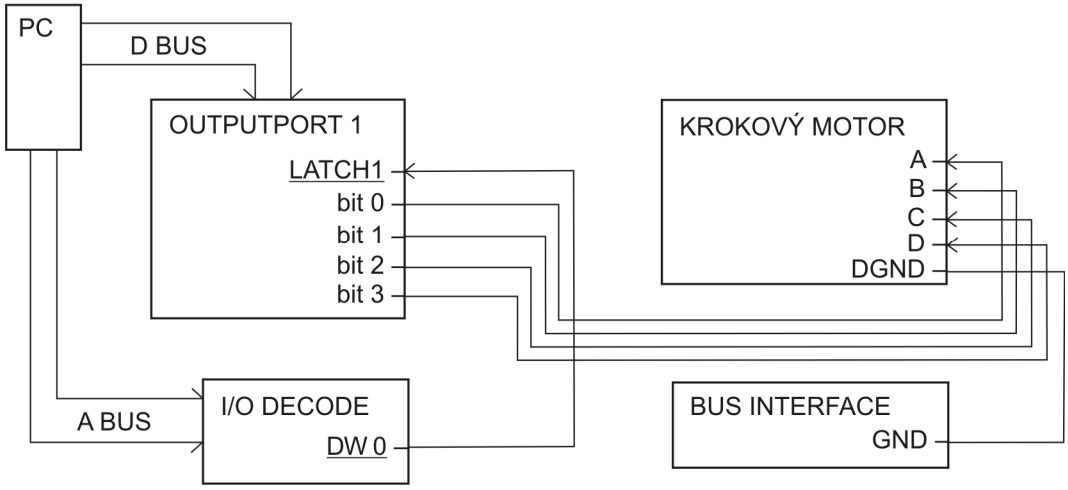
Standardní zapojení krokového motoru.

Tab. 5 – Tabulka zapojení úlohy 1

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1

Negované signály jsou značeny podtržítkem.

3.1.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 17 – Schéma zapojení úlohy 1

3.1.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_1;
uses crt, newdelay;
var I, J, otacky, DY : integer;
    smer : char;

begin
     (*Úvodní výpis programu*)
     writeln('MAT – Pridavny modul Krokovy motor');
     writeln('Uloha 1 – Zaklady rizeni modulu KRM ');
     writeln('Pro spusteni stisknete <Enter>');
     readln;

     (*Zadání počtu otáček*)
     write('Zadejte pocet otacek: ');
     readln(otacky);

     (*Zadání zpoždění*)
     repeat
           write('Zadejte zpozdeni v ms (rychlost otaceni) 1-500: ');
           readln(DY);
     until (DY>0) and (DY<501);

     (*Zadání směru otáčení*)
     repeat
           write('Zadejte smer otaceni L/R (L=vlevo, R=vpravo):  ');
           readln(smer);
     until (smer='L') or (smer='l') or (smer='R') or (smer='r');

     delay(500);
     If (smer='R') or (smer='r') then
     begin
          (*Otáčení vpravo*)
          writeln('Otaceni vpravo');
          for J:=0 to otacky -1 do
               for I:=0 to 11 do
               begin
                    port[$300]:=10;
                    delay(DY);
                    port[$300]:=6;
                    delay(DY);
                    port[$300]:=5;
```

```
                    delay(DY);
                    port[$300]:=9;
                    delay(DY);
               end;
     end
     else
     begin
          (*Otáčení vlevo*)
          writeln('Otaceni vlevo');
          for J:=0 to otacky -1 do
               for I:=0 to 11 do
               begin
                    port[$300]:=5;
                    delay(DY);
                    port[$300]:=6;
                    delay(DY);
                    port[$300]:=10;
                    delay(DY);
                    port[$300]:=9;
                    delay(DY);
               end;
     end;

     (*Ukončení programu*)
     writeln('Program dokoncil otaceni.');
     writeln('Pro ukonceni stisknete <Enter>');
     readln;
end.
```

3.1.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. Proměnná *I* a *J* slouží pro inkrementaci použitého cyklu *FOR*. Proměnná *otacky* udává počet otočení šipky o 360°. *DY* je proměnná, sloužící pro stanovení velikosti čekací doby mezi jednotlivými kroky motoru. Poslední použitou proměnnou je *smer*. Jedná se o proměnnou typu char, udávající směr otáčení šipky.

Na začátku samotného programu jsou funkce, které vypíšou na obrazovku základní informace o programu a informaci o tom, že program čeká na spuštění klávesou Enter. Program výpisem na obrazovku požádá uživatele o zadání počtu otáček. Načte se počet otáček do proměnné *otacky.* Následuje další žádost na uživatele o zadání zpoždění a následné načtení hodnoty *DY.* Načtení hodnoty je kontrolováno pomocí cyklu *Repeat – Until*, který bude opakovat zadávání hodnoty *DY*, dokud nebude ve vyznačených mezích. Nakonec stejným způsobem program zažádá a načte hodnotu určující směr otáčení. Zde je také pomocí cyklu *Repeat – Until* kontrolováno, aby uživatel zadal požadované hodnoty. Následuje podmínka, která pomocí proměnné *smer* rozdělí program na dvě větve. Pokud je hodnota proměnné *smer* R nebo r vykonává se otáčení vpravo a pokud ne, vykoná se otáčení doleva. Samotné otáčení je realizováno pomocí dvou do sebe

vnořených cyklů *FOR*. Vnější cyklus slouží k vykonání příslušného počtu otáček. Respektive opakuje vnitřní cyklus tolikrát, kolik bylo nastaveno v proměnné *otacky*. Vnitřní cyklus vykoná dvanáctkrát sekvenci čtyř řídicích příkazů, které budí krokování motoru. Na výstupní port, na který je připojen KRM pošle datové slovo, které posune ukazatel o jeden krok. Sekvence je tvořena čtyřmi datovými slovy a jejich pořadí udává směr otáčení. Tato sekvence je daná a vykoná se dvanáctkrát, čímž se motor posune o 48 kroků, což je jedna otáčka o 360°. Na závěr programu se ještě uživateli zobrazí na monitoru informace o doběhnutí programu a program čeká na ukončení uživatelem. Program se ukončí stiskem klávesy Enter.

3.1.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíšou na obrazovku informace o programu a výzva ke spuštění činnosti programu. Následně je uživatel požádán o zadání počtu otáček, zpoždění a směru otáčení. Jakmile uživatel zadá poslední hodnotu, spustí se otáčení KRM. Šipka ukazatele vykoná zadaný počet otáček vlevo či vpravo dle zadaných hodnot. Rychlost otáčení bude odpovídat zadanému zpoždění. Nakonec program vypíše na obrazovku hlášení o dokončení programu a bude čekat na ukončení uživatelem.

3.2 ÚLOHA 2 – TŘÍDICÍ STROJ

3.2.1 CÍL ÚLOHY

Hlavním cílem je vyzkoušení možností reálného využití krokového motorku pro přesné řízení určité činnosti. Dílčím cílem je procvičení používání základní krokovací sekvence pro pohyb na obě strany.

3.2.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky.
2. Naprogramujte třídičku odpadů. Ukazatel krokového motorku zde představuje rameno třídícího stroje. Třídička načte data ze vstupního portu, podle kterého bude třídit odpad na jednotlivá stanoviště.

3. Na logická 1 se přemístí šipka o 120° po směru hodinových ručiček, podá uživateli hlášení o odvozu odpadu na shromaždiště 1 a vrátí se do výchozí pozice na nule. Při logické 0 se ukazatel přemístí o 120° proti směru hodinových ručiček a informuje uživatele o odvozu odpadu na shromaždiště 0. Následně se opět vrátí do výchozí pozice.
4. Program bude obsahovat záměrná vhodná zpomalení.
5. Program bude komunikovat s uživatelem pomocí obrazovky.

3.2.3 ZAPOJENÍ

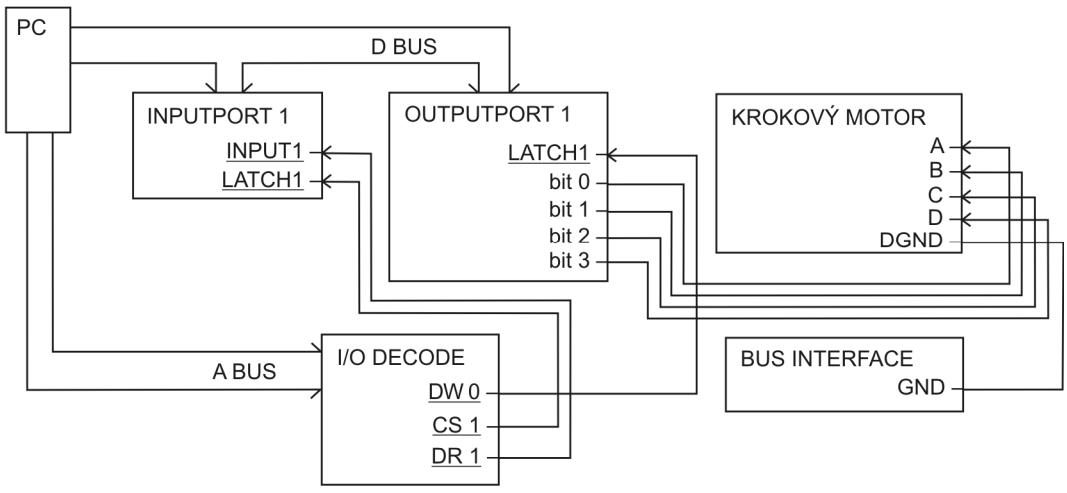
Standardní zapojení krokového motoru s připojením vstupního portu.

Tab. 6 – Tabulka zapojení úlohy 2

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1
I/O DECODE	CS1	→	INPUTPORT 1	LATCH1
I/O DECODE	DR1	→	INPUTPORT 1	INPUT1

Negované signály jsou značeny podtržítkem.

3.2.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 18 – Schéma zapojení úlohy 2

3.2.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_2;
uses crt, newdelay;
var I, J, data, delitel, shromazdiste, varka : integer;

(*Procedura která realizuje krokování na levou stranu*)
procedure Vlevo;
begin
     for J:=0 to 3 do
        begin
          port[$300]:=10;
          delay(20);
          port[$300]:=6;
          delay(20);
          port[$300]:=5;
          delay(20);
          port[$300]:=9;
          delay(20);
        end;
end;

(*Procedura která realizuje krokování na pravou stranu*)
procedure Vpravo;
begin
     for J:=0 to 3 do
        begin
          port[$300]:=5;
          delay(20);
          port[$300]:=6;
          delay(20);
          port[$300]:=10;
          delay(20);
          port[$300]:=9;
          delay(20);
        end;
end;

begin
     (*Úvodní výpis programu*)
     writeln('MAT-Pridavny modul Krokovy motor');
     writeln('Uloha 2 - Tridici stroj');
     writeln;
     writeln('Nastavte hodnotu na vstupním portu');
     writeln('Pro spusteni stisknete prosim <Enter>...');
     readln;
     Writeln('Stroj zpracovava vstupni data...');
     (*Načtení datového slova z inputportu*)
     data:= port[$301];
     port[$301]:= data;
     delay(1000);

     (*nastavení počátečních hodnot čítače várky a dělitele*)
     varka:=0;
     delitel:=128;

     (*Hlavní část programu. Řídicí část otáčení motorku*)
     for I:=0 to 7 do
     begin
          varka:= varka+1;
          writeln('Nakladam odpad - ',varka,'. varka');
          delay(1500);
          (*Získani jednotlivých bitů z datového slova*)
          shromazdiste:= data div delitel;
          data:= data mod delitel;
          delitel:= delitel div 2;
```

```
     (*Určení shromaždiště a volání procedur pro krokování*)
          if shromazdiste=1 then
          begin
              Vlevo;
              writeln('Vysypavam odpad na shromazdiste 1');
              delay(1500);
              writeln('Vracim se');
              Vpravo;
          end
          else
          begin
              Vpravo;
              writeln('Vysypavam odpad na shromazdiste 0');
              delay(1500);
              writeln('Vracim se');
              Vlevo;
          end;
     end;

     (*Ukončení programu*)
     writeln('Odpad roztriden...Pro ukonceni stisknete <Enter>');
     readln;
end.
```

3.2.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. Proměnná *I* a *J* slouží pro inkrementaci použitého cyklu *FOR*. Proměnná *data* obsahuje řídicí datové slovo, načítané ze vstupního portu, které řídí vysypávání odpadu. Proměnná *delitel* slouží k získání jednotlivých bitů datového slova. Dalšími proměnnými jsou *shromazdiste* a *varka*, kde *shromazdiste* udává, na které místo se má odpad vysypat, a *varka* v sobě nese informaci o tom, kolikátá várka odpadu se právě vysypává.

Další částí programu jsou dvě procedury Vlevo a Vpravo, které slouží k otočení motoru o přesně daný počet kroků na jednu a druhou stranu. Procedury jsou bez parametrů. Otáčení je realizováno standardní řídicí sekvencí čtyř datových slov poslaných na výstupní port. Pořadí datových slov udává, na kterou stranu se bude KRM otáčet. Tato sekvence se provede třikrát, čímž se šipka ukazatele pootočí o 120° na požadovanou stranu.

Na začátku samotného programu jsou funkce, které vypíší na obrazovku základní informace o programu a žádost o nastavení řídicího datového slova na vstupní port. Přičemž program čeká na potvrzení klávesou Enter. Po stisknutí se nejprve se vypíše hlášení na obrazovku a načte se řídicí datové slovo ze vstupního portu do *data*. Pak se spustí čekací procedura *delay*, nastaví se počáteční hodnoty proměnných *varka* na 0 a

delitel na 128. Následuje *FOR* cyklus, který se provede osmkrát a realizuje se v něm samotné třídění odpadu. Nejprve se inkrementuje *varka* a na obrazovku se vypíše informace o nakládání odpadu a číslo o kolikátou várku se jedná. Opět se spustí čekací procedura. Čekací procedury jsou zde umístěny záměrně pro zdržování programu, aby uživatel stíhal sledovat činnost KRM a zároveň informace vypisované na monitor. Další úsek programu slouží pro získání jednoho bitu z proměnné data. Do proměnné *shomazdiste* se uloží dělení proměnné *data* proměnnou *delitel*. Tímto se získá nejvíce významový bit. Do proměnné *data* se uloží zbytek po celočíselném dělení *data* proměnnou *delitel.* Ještě se zmenší *delitel* na polovinu. Tímto se připravily proměnné pro výpočty v dalším opakování cyklu. V dalším cyklu se takto vypočítá z *data* druhý nejvíce významový bit. Takto se postupně získají všechny bity z *data*, se kterými se v daném cyklu dále pracuje. Po získání bitu z řídicího datového slova se *shromazdiste* testuje jako logická proměnná. Pokud se rovná jedné, je podmínka pravdivá. Realizuje se posun na shromaždiště jedna na levé straně pomocí procedury *Vlevo*. Pak následuje informativní výpis na obrazovku, čekací procedura a opět informativní výpis. Pomocí procedury *Vpravo* se šipka ukazatele vrátí do výchozí pozice. Pokud byla podmínka nepravdivá, realizuje se prakticky to samé, pouze s tím rozdílem, že se nejprve volá procedura *Vpravo* a šipka jde na shromadiště nula a pak pro návrat na původní pozici se volá procedura *Vlevo*. Zde je konec cyklu *FOR* a dochází k opakování. Po tom, co doběhne cyklus *FOR,* se vypíše informativní hlášení o ukončení programu a čeká se na stisk klávesy Enter, kterou se program ukončí.

3.2.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíší na obrazovku informace o programu a výzva k nastavení příslušného datového slova na vstupním portu, podle kterého bude třídění probíhat. Po nastavení je možné spustit samotnou činnost programu. Po odstartování programu se vypíše na obrazovku hlášení o zpracovávání dat, přičemž program krátký čas čeká. Pak je uživatel informován o tom, že je nakládána první várka odpadu. To trvá zhruba sekundu a půl. Následně se přesune šipka na stanoviště jedna či nula. To záleží na tom, jaké je nastavené datové slovo na vstupním portu. Opět se krátký čas čeká, přičemž je uživatel informován o vysypávání odpadu. Následuje návrat na počáteční

pozici, o němž je uživatel opět informován hlášením na monitoru. Po návratu na původní startovní pozici se celý proces opakuje od fáze nabírání odpadu. Po dokončení třídění odpadu je uživatel informován o dokončení práce a vyzván k ukončení programu.

3.3 ÚLOHA 3 – LOSOVACÍ PŘÍSTROJ

3.3.1 CÍL ÚLOHY

Hlavním cílem je atraktivní formou si vyzkoušet možnosti využití modulu KRM a pokročilejší práci s krokovací sekvencí.

3.3.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky.
2. Naprogramujte losovací přístroj. Po spuštění programu se ručička ukazatele roztočí a po několika otáčkách se zastaví na náhodně zvoleném úhlu. Hodnota úhlu se také zobrazí uživateli na obrazovku.
3. Vytvořte program tak, aby se losování mohlo snadno opakovat.
4. Bonus: Vytvořte program tak, aby se krokování před dojetím na cílovou hodnotu zpomalovalo.

3.3.3 ZAPOJENÍ

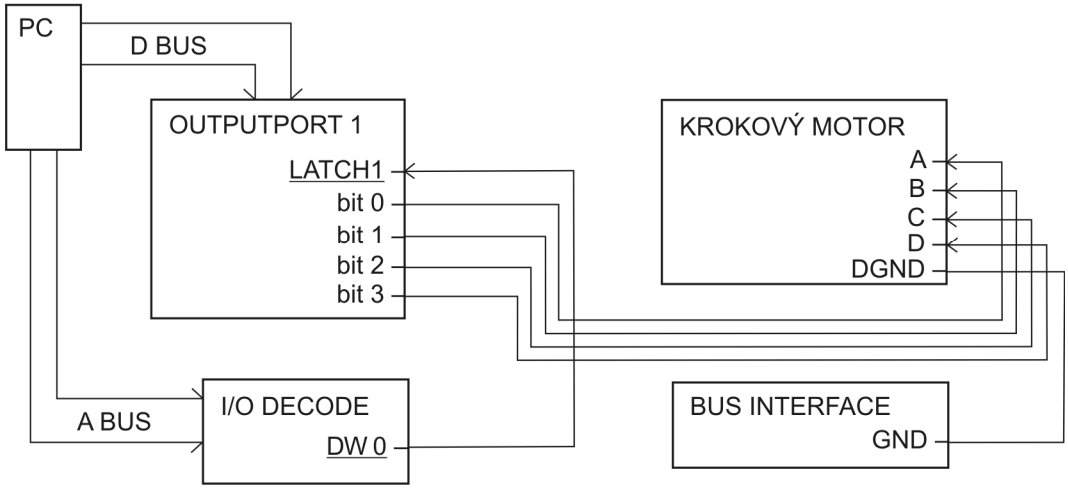
Standardní zapojení krokového motoru.

Tab. 7 – Tabulka zapojení úlohy 3

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1

Negované signály jsou značeny podtržítkem.

3.3.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 19 – Schéma zapojení úlohy 3

3.3.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_3;
uses crt, newdelay;
var I, DS, DY, odpocet, krokyC, uhel : integer;
    konec : char;
    vysledek : real;

begin
     (*Úvodní výpis programu*)
     writeln('MAT-Pridavny modul Krokovy motor');
     writeln('Uloha 3 - Losovaci pristroj');
     writeln('Pro spusteni stisknete <Enter>');
     readln;

     (*Nastavení proměnných po spustění programu*)
     uhel:=0;
     DS:=10;
     (*inicializace generátoru náhodných čísel*)
     randomize;

repeat
     (*Pseudonáhodné generování kroků*)
     krokyC:=(random(47)+1)+(48*(random(5)+1));
     (*Nastavení odpočtu pro změnu rychlosti krokování a počáteční rychlosti*)
     odpocet:=krokyC;
     DY:=30;
     (*Výpočet výsledného úhlu*)
     uhel:=((krokyC+uhel) mod 48);
     vysledek:= uhel*7.5;

     for I:=0 to krokyC -1 do
     begin
          (*Odeslání datového slova na výstupní port*)
          port[$300]:=DS;
          delay(DY);

          (*Nastavení datového slova pro příští krok*)
          if DS=10 then DS:=6
          else if DS=6 then DS:=5
          else if DS=5 then DS:=9
          else if DS=9 then DS:=10;
```

```
          (*Nastavení rychosti krokování pomocí fce delay*)
          odpocet:= odpocet -1;
          if odpocet =48 then DY:=40;
          if odpocet =24 then DY:=50;
          if odpocet =12 then DY:=70;
          if odpocet =6 then DY:=100;
          if odpocet =3 then DY:=150;
          if odpocet =1 then DY:=300;

     end;

     (*Vypsaní úhlu na obrazovku*)
     writeln('HOD: ',vysledek:3:1);
     (*realizace ukončení, nebo pokračování v programu*)
     writeln('Pro ukonceni zadejte Q+<Enter>, pro pokracovani 2x <Enter>');
     readln(konec);

until (konec='Q') or (konec='q');
end.
```

3.3.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. Proměnná *I* slouží pro inkrementaci použitého cyklu *FOR*. *DS* je řídicí datové slovo sloužící k ovládání KRM modulu. *DY* je proměnná sloužící pro stanovení velikosti čekací doby. Proměnná *odpocet* slouží pro řízení zpomalování krokování. Další proměnná je *krokyC*. Do této proměnné se ukládá celkový počet kroků, který se provede při aktuálním losování. Proměnná *uhel* je hodnota úhlu v krocích, zobrazeného v předchozím losování. Další je proměnná *konec* typu char, která slouží k ukončení programu. Poslední deklarovanou proměnnou je *vysledek*, sloužící k uložení a výpisu hodnoty úhlu, jenž ukazuje šipka.

Na začátku samotného programu jsou funkce, které vypíší na obrazovku základní informace o programu a informaci o tom, že program čeká na spuštění klávesou Enter. Po spuštění se nejprve nastaví počáteční hodnoty proměnných *uhel* na 0 a *DS* na 10. Proměnná *uhel* zde reprezentuje hodnotu předchozího vylosovaného úhlu, proto tato hodnota musí být při spuštění programu nulová. *DS* je řídicí datové slovo. Musí být nastaveno na hodnotu 10, protože je to datové slovo pro první krok při otáčení po směru hodinových ručiček. Další nastavení *DS* probíhá již v těle programu. Příkaz *randomize* slouží k inicializaci generátoru pseudonáhodných čísel. Nyní následuje cyklus *Repeat – Until*, který bude opakovat hlavní část programu. Nejprve pomocí generování náhodných čísel se získá počet kroků, který se uloží do proměnné *krokyC*. Zde lze ovlivnit minimální a maximální počet otočení šipky. Hodnota je určena rozsahem generování, který je vždy 0

až zadané číslo mínus jedna. Tato hodnota může být upravena přičtením či vynásobením jiným číslem. Dalšími příkazy jsou nastavení hodnoty *odpocet* na hodnotu celkového počtu kroků a nastavení čekací doby mezi kroky na hodnotu 30ms. Dále se vypočítává počet kroků, který reprezentuje hodnotu úhlu, na kterém se šipka zastaví. Tato hodnota se vypočítá sečtením celkového počtu kroků a předchozího úhlu v krocích. Ze součtu se získá zbytek po celočíselném dělení číslem 48. To je počet kroků pro jednu otáčku šipky o 360°. Výsledné číslo se uloží do proměnné *uhel*. Dalším příkazem se spočítá hodnota výsledného úhlu ve stupních a uloží do proměnné *vysledek*. Následuje cyklus *FOR*, v němž se realizuje krokování. Cyklus se provádí tolikrát, kolik je celkový počet kroků v proměnné *krokyC*. V cyklu se nejprve pošle na výstupní port řídicí datové slovo *DS*. Pak se spustí čekací procedura *delay*. Tato procedura čeká stanovený čas *DY*. Následně dojde pomocí složené podmínky ke změně *DS*, aby mohl být proveden další krok. Dekrementuje se *odpocet*. Další složená podmínka slouží ke změně čekací doby. Tím se docílí toho, že při snižování počtu kroků, které se ještě musí vykonat, se bude krokování zpomalovat. Následuje konec cyklu *FOR*. Rychlost krokování je tedy dána čekací dobou *DY* a dobou, než se vykoná cyklus *FOR*. Následuje vypsání výsledného úhlu na obrazovku a výzva pro uživatele, zda chce pokračovat v losování, nebo program ukončit. Pro ukončení programu musí uživatel zadat Q či q.

3.3.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíšou na obrazovku informace o programu a výzva ke spuštění činnosti programu. Jakmile dá uživatel klávesou pokyn k odstartování losování, roztočí se šipka ukazatele úhlu KRM. Po několika otáčkách se rotace šipky začne zpomalovat až se zastaví na určitém úhlu. Následně je tento úhel vypsán na obrazovku a uživatel je vyzván, zda chce program ukončit, nebo pokračovat v losování. Pokud se uživatel rozhodne pokračovat v losování, program se opakuje od místa losování, ale počáteční výpis se již nezobrazuje. V opačném případě se program ukončí.

3.4 ÚLOHA 4 – HRA

3.4.1 CÍL ÚLOHY

Hlavním cílem je atraktivní formou si vyzkoušet propojení modulu KRM s dalšími částmi MATu, především s pulsery.

3.4.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky.
2. Naprogramujte hru, kde se hráči budou snažit pomocí pulserů strefit co nejpřesněji zadaný úhel. Úhel bude náhodně vygenerován a zadán na obrazovce. Hráči budou pozorovat otáčející se šipku a snažit se strefit zadaný úhel.
3. Strefování se do správného úhlu končí po tom, co oba hráči zmáčknou pulser. Zaznamenává se pouze první stisk pulseru.
4. Zvolte rychlost otáčení tak, aby nebylo moc snadné strefit správný úhel.
5. Program bude komunikovat s uživateli pomocí monitoru.

3.4.3 ZAPOJENÍ

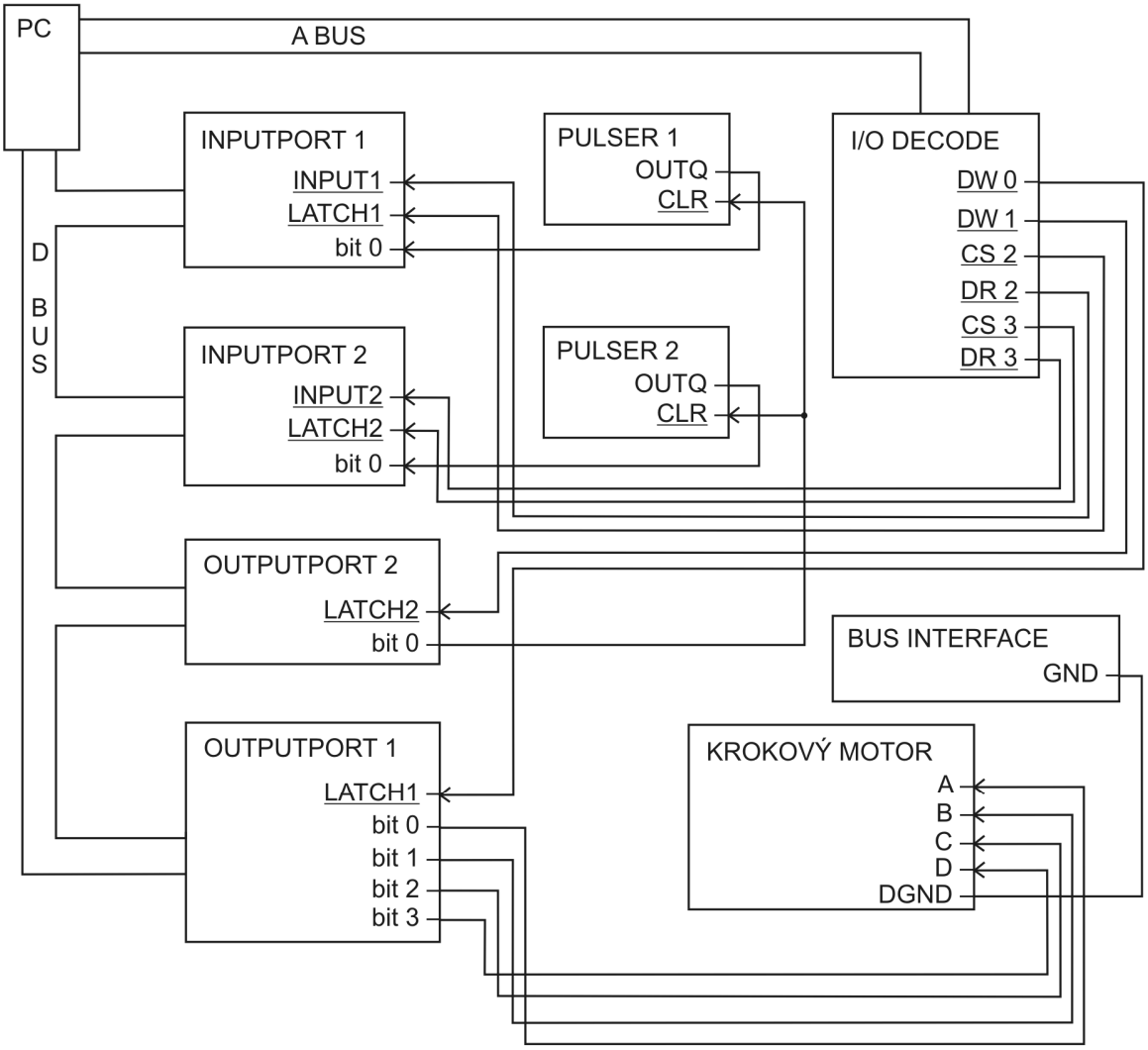
Standardní zapojení krokového motoru s připojením obou vstupních i výstupních portů a dvou pulserů.

Tab. 8 – Tabulka zapojení úlohy 4

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1
I/O DECODE	DW1	→	OUTPUTPORT 2	LATCH1
I/O DECODE	CS2	→	INPUTPORT 1	LATCH1
I/O DECODE	DR2	→	INPUTPORT 1	INPUT1
I/O DECODE	CS3	→	INPUTPORT 2	LATCH2
I/O DECODE	DR3	→	INPUTPORT 2	INPUT2
PULSER1	OUTQ	→	INPUTPORT 1	Bit 0
PULSER2	OUTQ	→	INPUTPORT 2	Bit 0
OUTPUTPORT 2	Bit 0	→	PULSER1	CLR
OUTPUTPORT 2	Bit 0	→	PULSER2	CLR

Negované signály jsou značeny podtržítkem.

3.4.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 20 – Schéma zapojení úlohy 4

3.4.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_4;
uses crt, newdelay;
var I, DS, P1uhel, P2uhel, uhelA, uhelB, odchylkaP1, odchylkaP2 : integer;
    key : char;
    P1, P2 : boolean;

begin
     (*Úvodní výpis programu*)
     writeln;
     writeln('MAT-Pridavny modul Krokovy motor');
     writeln('Uloha 4 – Hra');
     writeln('Pro spusteni stisknete <Enter>');
     readln;

  repeat
     (*Nastavení proměnných po spustění programu*)
     DS:=10;
     uhelB:=0;
```

```
P1:= false;
P2:= false;
(*Reset pulserů*)
port[$301]:=0;
port[$301]:=1;

(*Zadání hledaného úhlu*)
randomize;
uhelA:=(random(48)+1);
writeln('Pozadovany uhel: ', uhelA*7.5:3:1 ,' stisknete <Enter> pro start');
readln;

(*Hlavní tělo programu*)
repeat
      For I:=0 to 47 do
      begin
          (*Realizace krokování*)
           port[$300]:=DS;
           delay(10);

           if DS=10 then DS:=6
           else if DS=6 then DS:=5
           else if DS=5 then DS:=9
           else if DS=9 then DS:=10;

           (*Inkrementace úhlu*)
           inc(uhelB);

           (*Testování stisknutí pulserů*)
           if not(P1) then
           begin
                P1uhel:=port[$302];
           end;

           if not(P2) then
           begin
                P2uhel:=port[$303];
           end;

           (*Uložení hodnoty úhlu po stisknutí pulseru*)
           if (P1uhel=1) and (not(P1)) then
           begin
                P1uhel:=uhelB;
                writeln('Uhel hrace1 je: ', P1uhel*7.5:3:1 );
                P1:=true;
           end;

           if (P2uhel=1) and (not(P2)) then
           begin
                P2uhel:=uhelB;
                writeln('Uhel hrace2 je: ', P2uhel*7.5:3:1 );
                P2:=true;
           end;

           (*Nulování uhlu po jedné otáčce*)
           if uhelB=48 then uhelB:=0;
      end;
until (P1 and P2);

(*Zjištění odchylky*)
odchylkaP1:= abs(uhelA-P1uhel);
if odchylkaP1>24 then odchylkaP1:=abs(odchylkaP1-48);
odchylkaP2:= abs(uhelA-P2uhel);
if odchylkaP2>24 then odchylkaP2:=abs(odchylkaP2-48);

(*Testování, který hráč trefil bližší hodnotu, vyhlášení vítěze*)
if odchylkaP1 = odchylkaP2 then
```

```
     begin
          writeln('Remiza - oba hraci stikli zaroven. Jejich odchylka je:
             ',odchylkaP1*7.5:3:1,' stupnu');
     end
     else
     begin
          if abs(uhelA-P1uhel) < abs(uhelA-P2uhel) then
               writeln('Vytezem je hrac 1. s odchylkou: ',odchylkaP1*7.5:3:1,'
             stupnu')
          else
               writeln('Vytezem je hrac 2. s odchylkou: ',odchylkaP2*7.5:3:1,'
             stupnu');
     end;

     (*Opakování či ukončení programu*)
     repeat
           write('Chcete hrat znovu Y/N: ');
           readln(key);
     until (key='n') or (key='N') or (key='y') or (key='Y');
     writeln;

  until (key='n') or (key='N');
end.
```

3.4.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. První proměnná *I* slouží k inkrementaci cyklu *FOR*. *DS* je řídicí datové slovo sloužící k ovládání KRM modulu. Další proměnné jsou *P1uhel* a *P2uhel*. Tyto proměnné obsahují hodnoty úhlů zaznamenané při stisku pulserů. Proměnná *uhelA* udává hodnotu úhlu, kterou mají hráči strefit a *uhelB* slouží k zaznamenání právě zobrazeného úhlu. Obě hodnoty jsou udávány v krocích. Proměnné *odchylkaP1* a *odchylkaP2* udávají rozdíl úhlu, jenž měli hráči strefit, a úhlu, který skutečně strefili. Proměnná *key* je typu char a slouží k ukončení programu. Poslední proměnné jsou *P1* a *P2* typu boolean, které slouží k identifikaci stisku pulseru.

Na začátku samotného programu jsou funkce, které vypíší na obrazovku základní informace o programu a informaci o tom, že program čeká na spuštění klávesou Enter. Po spuštění se nejprve provede základní nastavení proměnných. *DS* se nastaví na hodnotu 10, což je hodnota prvního kroku ve směru hodinových ručiček. Proměnná *uhelB* se nastaví na 0, což je počáteční pozice šipky ukazatele. Proměnné *P1* a *P2*, které určující stisknutí pulserů, se nastaví na false. Dále dojde k připravení nebo-li resetování pulserů. Pak následuje generování úhlu pomocí funkce *Random*. Následuje hlavní část programu. Ta se bude opakovat, pomocí cyklu *Repeat – Until*, dokud nebudou stisknuty oba pulsery. Do tohoto cyklu je vnořen ještě cyklus *FOR*, který slouží k tomu, aby se provedla vždy celá

jedna otáčka a šipka končila na nule. V tomto cyklu se provede jeden krok, změní se řídicí datové slovo *DS* a inkrementuje se proměnná *uhelB*. Tím se zajistí, že v *uhelB* bude stejný úhel, který ukazuje šipka. Dále se provádí testování pulserů. Pokud nebyl pulser stisknut, načítá se hodnota, kterou vrací. Pokud již stisknut byl, žádná hodnota se nenačítá. Následuje vyhodnocení stisku. Pokud pulser vrátil hodnotu jedna, znamená to, že pulser byl právě stisknut. Tuto hodnotu lze brát jako logickou proměnnou, která se zde testuje. Tudíž pokud ke stisku došlo, hodnota *P1uhel* či *P2uhel* bude jedna. Stisknutím pulseru se spustí načtení právě zobrazeného úhlu, informativní výpis na obrazovku a zakáže se další testování stisknutí. Navíc se hodnota načtená z pulseru přepíše hodnotou zobrazeného úhlu. Po stisknutí obou pulserů a dokončení cyklu *FOR* dojde k ukončení cyklu *Repeat – Until*. Dále se vypočítají odchylky obou hráčů. Pak je na řadě testování. Testuje se, který hráč byl blíže zadanému úhlu a ten je vyhlášen za vítěze. Spolu s informacemi o vítězství se vypíše odchylka, s jakou se vyhrávající hráč strefil. Poslední částí programu je ukončení. Zde je uživatel dotázán zda chce hrát znovu či program ukončit. Pokud se rozhodne hrát znovu, program se opakuje od místa nastavení proměnných. V opačném případě program končí.

3.4.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíší na obrazovku informace o programu a výzva ke spuštění činnosti programu. Jakmile uživatel stiskne klávesu enter, dojde k vygenerování hledaného úhlu, který se vypíše na obrazovku a program vyzve uživatele k odstartování samotné hry. Po odstartování se roztočí šipka ukazatele a čeká se na stisk pulserů. Jakmile je stisknut jeden z pulserů, zaznamená se hodnota úhlu, kterou právě ukazuje šipka, a tato hodnota je vypsána na monitor spolu s informací o tom, který hráč tlačítko stiskl a jaký úhel strefil. Po tom, co stisknou tlačítko oba hráči, se vyhodnotí, který hráč byl s trefenou hodnotou blíže hodnotě zadané. Na monitor se vypíše, kdo z hráčů vyhrál a jak přesně se strefil. Následně je uživatel dotázán, zda chce v programu pokračovat, nebo zda ho chce ukončit. Pokud se rozhodne pokračovat, program se opakuje od místa vygenerování hledaného úhlu. V opačném případě se program ukončí.

3.5 ÚLOHA 5 – ŘÍZENÍ S POLOVIČNÍM KROKEM

3.5.1 CÍL ÚLOHY

Hlavním cílem úlohy je vyzkoušet si řízení KRM s polovičním krokem.

3.5.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky
2. Vytvořte program, který bude realizovat otáčení KRM. Směr a úhel otáčení zadá uživatel.
3. Použijte krokovací sekvenci, která umožní dvojnásobný počet kroků, tzv. řízení s polovičním krokem.
4. Program bude provádět základní komunikaci s uživatelem.

3.5.3 ZAPOJENÍ

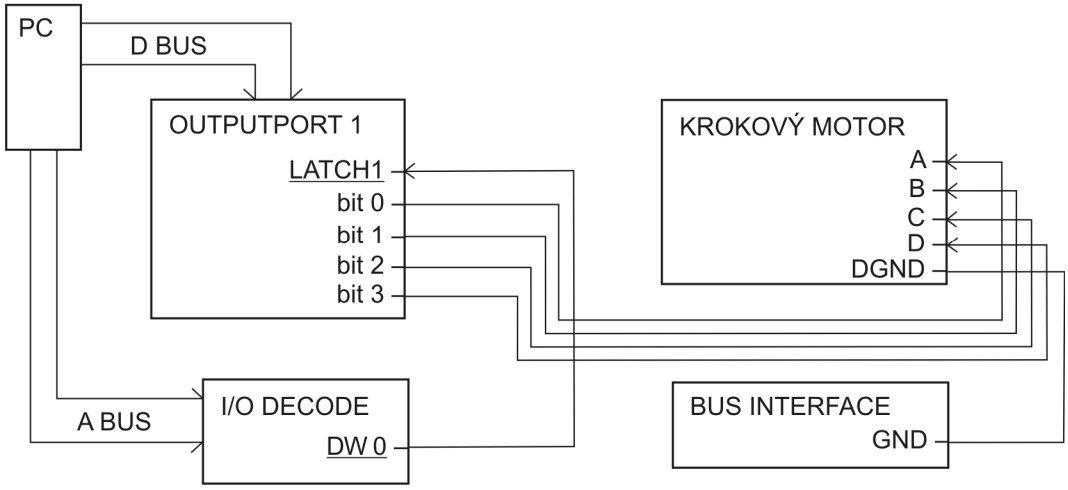
Standardní zapojení krokového motoru.

Tab. 9 – Tabulka zapojení úlohy 5

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1

Negované signály jsou značeny podtržítkem.

3.5.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 21 – Schéma zapojení úlohy 5

3.5.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_5;
uses crt, newdelay;
var DS, krokyC, uhelB : integer;
    smer, konec : char;
    uhel : real;

(*Procedura pro krokování vlevo*)
procedure Vlevo(krokyC:integer);
var I : integer;
begin
     for I:=0 to krokyC-1 do
     begin
         if DS=9 then DS:=13
         else if DS=13 then DS:=5
         else if DS=5 then DS:=7
         else if DS=7 then DS:=6
         else if DS=6 then DS:=14
         else if DS=14 then DS:=10
         else if DS=10 then DS:=11
         else if DS=11 then DS:=9;
         port[$300]:=DS;
         delay(20);
     end;
end;

(*Procedura pro krokování vpravo*)
procedure Vpravo(krokyC:integer);
var I : integer;
begin
     for I:=0 to krokyC-1 do
     begin
         if DS=9 then DS:=11
         else if DS=11 then DS:=10
         else if DS=10 then DS:=14
         else if DS=14 then DS:=6
         else if DS=6 then DS:=7
         else if DS=7 then DS:=5
         else if DS=5 then DS:=13
         else if DS=13 then DS:=9;
         port[$300]:=DS;
         delay(20);
```

```
     end;
end;

begin
     (*Úvodní výpis programu *)
     writeln('MAT-Pridavny modul Krokovy motor');
     writeln('Uloha 5 - Rizeni s polovicnim krokem');
     writeln('Pro spusteni stisknete <Enter>');
     readln;

     (*Nastavení proměnných po spustění programu*)
     DS:=9;
     uhelB:=0;
     repeat
           (*Zadání směru otáčení*)
           repeat
                 write('Zadejte smer otaceni R/L: ');
                 readln(smer);
                 smer:=upcase(smer);
           until (smer='R') or (smer='L');

           (*Zadání úhlu, o který se má ukazatel posunout*)
           write('Zadejte uhel o ktery se ma ukazatel posunout: ');
           readln(uhel);
           uhel:=uhel/3.75;
           krokyC:=round(uhel);

           (*Realizace otáčení. Volání krokovacích procedur*)
           if smer='L' then
           begin
                Vlevo(krokyC);
                uhelB:=uhelB-krokyC;
           end
           else
           begin
              Vpravo(krokyC);
              uhelB:=uhelB+krokyC;
           end;

           (*Výpis úhlu*)
           uhelB:=uhelB mod 96;
           if uhelB<0 then uhelB:=uhelB+96;
           writeln('Zobrazeny uhel je ',(uhelB*3.75):5:2);

           (*ukončení programu*)
           writeln('Hotovo - Pro pokracovani stisnete 2X <Enter> a pro ukonceni
             klavesu Q + <Enter>');
           readln(konec);
           konec:=upcase(konec);
     until konec='Q';
end.
```

3.5.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. *DS* je řídicí datové slovo sloužící k ovládání KRM modulu. Další proměnná je *krokyC*. Do této proměnné se ukládá celkový počet kroků, o který se provede otáčení. Proměnná *smer* je typu char a udává směr otáčení. Další je proměnná *konec* typu char, která slouží k ukončení programu. Poslední deklarovanou

proměnnou je *uhel*, která slouží k zadání úhlu, o který se má KRM pootočit, a k výpočtu počtu kroků.

Další částí programu jsou dvě procedury Vlevo a Vpravo, které slouží k otočení motoru o daný počet kroků na jednu či druhou stranu. Procedury jsou s parametrem *krokyC*, který udává počet kroků, o který se motor pootočí. Otáčení je realizováno pomocí cyklu *FOR*. V jednom průchodu cyklem se provede právě jeden krok, kdy se nejprve vybere datové slovo *DS* pomocí složené podmínky a pak se toto datové slovo pošle na výstupní port. Tímto se realizuje jeden krok. Počet kroků, které se provedou udává právě parametr *krokyC*.

Důležité je, že zde námi realizovaný krok má poloviční velikost. Jedná se o tzv. půlkrok. Tento půlkrok je realizován tak, že do naší původní krokovací sekvence jsou přidána další datová slova, která mají hodnotu logického součtu předchozího a následujícího datového slova. To znamená, že mezi datové slovo 9 a 10 je vložen jejich binární součet, kterým je datové slovo 11. Když se krokovému motoru pošle toto datové slovo, aktivují se dva pólové nástavce vedle sebe a poloha šipky ukazatele bude ve středu mezi těmito dvěma pólovými nástavci. Tímto způsobem se vytvoří sekvenci s dvojnásobným počtem kroků.

Na začátku samotného programu jsou funkce, které vypíší na obrazovku základní informace o programu a žádost o stisknutí klávesy Enter. Po stisku klávesy se nejprve nastaví proměnná *DS* na hodnotu 9, což je hodnota výchozí polohy na nule. Tím se zajistí, že krokovací sekvence vlevo i vpravo bude začínat správně. Následuje žádost o zadání směru otáčení a úhlu, o který se má ukazatel posunout. Zadání směru je ošetřeno tak, aby uživatel mohl zadat pouze požadované hodnoty. U zadávání úhlu je uživateli umožněno zadat jakýkoliv úhel, který je následně zaokrouhlen na nejbližší hodnotu úhlu, který lze pomocí dvojnásobného počtu kroků realizovat. Po zadání vstupních hodnot se realizuje samotné otáčení. Nejprve se pomocí podmínky určí směr otáčení a podle toho se zavolá příslušná procedura pro otáčení vpravo či vlevo. Procedura se volá s parametrem *krokyC*. Po provedení krokování výše zmíněnou procedurou se přejde k poslední části programu. Uživatel dostane nabídku, zda chce v programu pokračovat či nikoliv. Pokud se rozhodne

pokračovat, program se začne opakovat od místa zadávání směru otáčení. Jinak se program ukončí.

3.5.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíšou na obrazovku informace o programu a výzva ke spuštění činnosti programu. Následně je uživatel požádán o zadání směru otáčení a úhlu o který se má šipka ukazatele posunout. Po zadání hodnot odkrokuje ručička ukazatele o zadaný úhel a uživatel je dotázán, zda chce v pokračovat v činnosti programu, nebo zda chce skončit. Pokud se uživatel rozhodne pokračovat, program se opakuje od zadávání směru otáčení.

3.6 ÚLOHA 6 – KOMPAS

3.6.1 CÍL ÚLOHY

Hlavním cílem úlohy je procvičit si pokročilejší práci s řízením KRM.

3.6.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky
2. Naprogramujte kompas. Uživatel zadá jakým směrem se otočí. Šipka ukazatele se přetočí tak, aby v případě, že nula ukazuje směr, který zadal uživatel, šipka ukazovala sever.
3. Kompas bude ukazovat osm světových směrů (S, SV, V, ...)
4. Šipka ukazatele by se měla přesouvat vždy kratší cestou.
5. Program bude komunikovat s uživatelem pomocí monitoru.
6. Bonus: Vytvořte kompas, který bude ukazovat šestnáct světových směrů.

3.6.3 ZAPOJENÍ

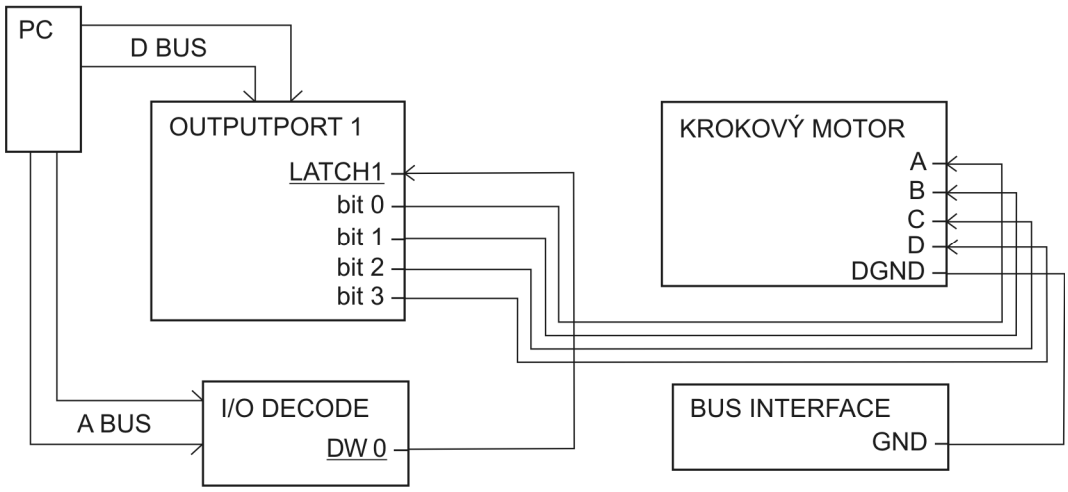
Standardní zapojení krokového motoru.

Tab. 10 – Tabulka zapojení úlohy 6

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1

Negované signály jsou značeny podtržítkem.

3.6.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 22 – Schéma zapojení úlohy 6

3.6.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_6;
uses crt, newdelay;
var I, DS, krokyC : integer;
    uhelA, uhelB, uhelC : real;
    smer : string[3];
    CW : boolean;


(*Funkce, která převede řetězec na velká písmena*)
function UpString(s:string):string;
var i: byte;
begin
     for i:=1 to length(s) do s[i]:=upcase(s[i]);
     UpString:=s;
end;


begin
    (*Úvodní výpis programu *)
```

```
  writeln('MAT-Pridavny modul Krokovy motor');
  writeln('Uloha 6');
  writeln('Pro spusteni stisknete <Enter>');
  readln;

  (*Počáteční nastavení proměnných *)
  DS:=9;
  uhelB:=0;
  writeln('Jste otoceni na sever');

repeat
  (*Výpis směrů otáčení*)
  writeln('Zadejte smer kterym se chcete otocit');
  writeln;
  writeln('Sever - S');
  writeln('Jih - J');
  writeln('Vychod - V');
  writeln('Zapad - Z');
  writeln('Severovychod -SV');
  writeln('Severozapad - SZ');
  writeln('Jihovychod - JV');
  writeln('Jihozapad - JZ');
  writeln('Severseverovychod - SSV');
  writeln('Severoseverozapad - SSZ');
  writeln('Jihojihovychod - JJV');
  writeln('Jihojihozapad - JJZ');
  writeln('Vychodseverovychod - VSV');
  writeln('Vychodjihovychod - VJV');
  writeln('Zapadjihozapad - ZJZ');
  writeln('Zapadseverozapad - ZSZ');
  Writeln('--- pro ukonceni zadejte Q ---');
  readln(smer);

  smer:=UpString(smer);

  (*Zjištění úhlu*)
  if smer='S' then uhelA:=0
  else if smer='SSZ' then uhelA:=22.5
  else if smer='SZ' then uhelA:=45
  else if smer='ZSZ' then uhelA:=67.5
  else if smer='Z' then uhelA:=90
  else if smer='ZJZ' then uhelA:=112.5
  else if smer='JZ' then uhelA:=135
  else if smer='JJZ' then uhelA:=157.5
  else if smer='J' then uhelA:=180
  else if smer='JJV' then uhelA:=202.5
  else if smer='JV' then uhelA:=225
  else if smer='VJV' then uhelA:=247.5
  else if smer='V' then uhelA:=270
  else if smer='VSV' then uhelA:=292.5
  else if smer='SV' then uhelA:=315
  else if smer='SSV' then uhelA:=337.5;

  (*Zjištění směru a velikosti krokování *)
  uhelC:=uhelA-uhelB;
  uhelB:=uhelA;
  if uhelC>0 then
  begin

       if abs(uhelC)<=180  then
       begin
            CW:=true;
            uhelC:=abs(uhelC);
       end
       else
       begin
            CW:=false;
```

```
               uhelC:=360-abs(uhelC);
          end;
     end
     else
     begin
          if abs(uhelC)<=180  then
          begin
               CW:=false;
               uhelC:=abs(uhelC);
          end
          else
          begin
               CW:=true;
               uhelC:=360-abs(uhelC);
          end;
     end;
     krokyC:=trunc(uhelC/7.5);

     (*Realizace krokování *)
     for I:=1 to krokyC do
     begin
          if CW then
             begin
                  if DS=10 then DS:=6
                  else if DS=6 then DS:=5
                  else if DS=5 then DS:=9
                  else if DS=9 then DS:=10;
             end
             else
             begin
                  if DS=10 then DS:=9
                  else if DS=9 then DS:=5
                  else if DS=5 then DS:=6
                  else if DS=6 then DS:=10;
             end;

             port[$300]:=DS;
             delay(20);

          end;
  until(smer='Q');

     (*Ukončení prohramu*)
     writeln('Program ukoncen ... stisknete <Enter>');
     readln;
end.
```

3.6.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. První proměnná *I* slouží pro inkrementaci použitého cyklu *FOR*. *DS* je řídicí datové slovo sloužící k ovládání KRM modulu. Další proměnná je *krokyC*. Do této proměnné se ukládá celkový počet kroků, o který se provede otáčení. Další jsou tři proměnné typu real. Proměnná *uhelA* udává hodnotu úhlu, na kterou se má ukazatel otočit, a *uhelB* udává hodnotu úhlu, na kterou právě ukazuje šipka ukazatele. Proměnná *uhelC* je výslednicí rozdílu předešlých proměnných a udává uhel, o který se má ukazatel otočit ve stupních. Proměnná *smer* je typu string a udává směr otočení kompasu.

Poslední proměnnou je *CW* udávající, zda se bude motor otáčet po směru nebo proti směru hodinových ručiček.

Další částí programu je funkce *UpString*, která slouží pro převedení všech písmen v řetězci na velká písmena. Funkce funguje tak, že převádí jeden znak řetězce po druhém a vrací převedený řetězec.

Na začátku samotného programu jsou funkce, které vypíší na obrazovku základní informace o programu a žádost o stisknutí klávesy Enter. Po stisku klávesy se nejprve nastaví proměnná *DS* na hodnotu 9, což je hodnota výchozí polohy na nule. Tím se zajistí, že krokovací sekvence vlevo i vpravo bude začínat správně. Dále se nastaví hodnota proměnné *uhelB* na 0, jelikož je šipka ukazatele na výchozí pozici. Na obrazovku se uživateli vypíše informace o směru otočení a žádost o zadání směru. Spolu s žádostí se vypíší i směry, které může uživatel zadat. Po zadání směru dojde pomocí složené podmínky k přiřazení správného úhlu, na který se přesune šipka. Před krokováním se musí nejprve zjistit směr, kolik kroků bude potřeba. Pomocí rozdílu proměnných *uhelA* a *uhelB* se zjistí vzdálenost těchto dvou úhlů a uloží se do proměnné *uhelC*. Ze získané vzdálenosti úhlů se určí, kterým směrem a jak daleko se bude krokovat. Velikost krokovaného úhlu se ještě převede na počet kroků a pak se realizuje samotné krokování. Celý program se opakuje od místa zadání směru, dokud není místo směru zadáno písmeno Q. V tomto případě proběhne celý program na prázdno a ukončí se. Před samotným ukončením je uživatel informován o skončení programu a musí jej potvrdit stiskem klávesy enter.

3.6.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíšou na obrazovku informace o programu a výzva ke spuštění činnosti programu. Po stisknutí klávesy enter je uživatel informován o tom, že je otočen na sever a je požádán o zadání směru, kterým se chce otočit, přičemž se na obrazovku vypíší směry, které může uživatel zadat. Pokud uživatel místo zadání směru zadá písmeno Q, program se projede na prázdno a ukončí se. Po zadání směru se ručička ukazatele otočí tak, aby zvolený směr byl reprezentován nulou na ukazateli a šipka mířila na sever. Program se opakuje znovu od místa zadávání směru, dokud není ukončen.

3.7 ÚLOHA 7 – HLEDÁNÍ ÚHLU

3.7.1 CÍL ÚLOHY

Hlavním cílem úlohy je procvičit si práci s řízením KRM. Dílčím cílem je vyzkoušet si možnosti propojení s dalšími částmi MATu.

3.7.2 ZADÁNÍ

1. Zapojte modul KRM dle uvedeného schématu a zapojovací tabulky
2. Vytvořte program, jenž vygeneruje úhel, který se bude uživatel snažit najít. Uživatel hledaný úhel nezná a v hledání mu pomáhá světelná signalizace. Při zadání hodnoty úhlu se nejenom změní signalizace, ale přesune se i šipka.
3. Ze začátku bude vygenerován úhel, na který se šipka ukazatele přesune a hledání bude začínat od něho.
4. Bonus: Po skončení programu dojeďte kurzorem na výchozí nulovou pozici.

3.7.3 ZAPOJENÍ

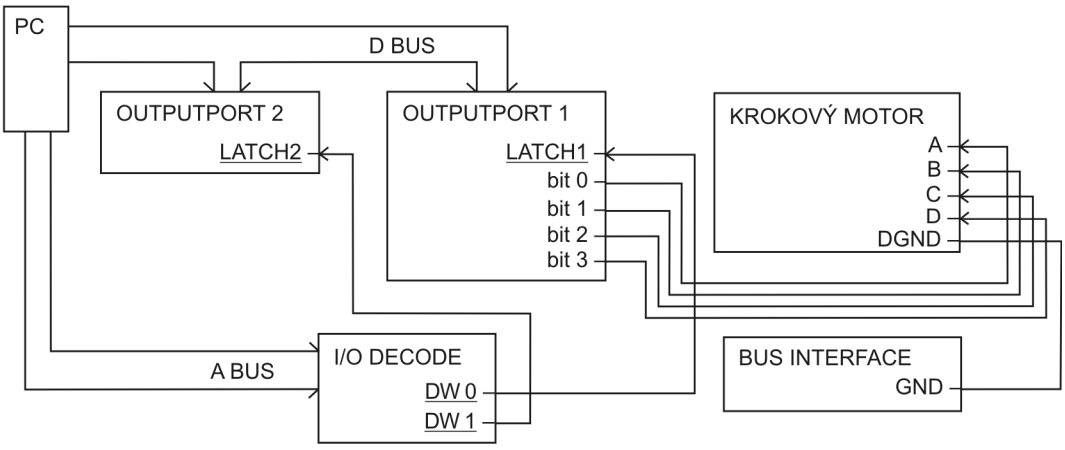
Standardní zapojení krokového motoru s připojením druhého výstupního portu.

Tab. 11 – Tabulka zapojení úlohy 7

Sekce	Signál		Sekce	Signál
KRM	A	→	OUTPUTPORT 1	Bit 0
KRM	B	→	OUTPUTPORT 1	Bit 1
KRM	C	→	OUTPUTPORT 1	Bit 2
KRM	D	→	OUTPUTPORT 1	Bit 3
KRM	DGND	→	BUS INTERFACE	GND
I/O DECODE	DW0	→	OUTPUTPORT 1	LATCH1
I/O DECODE	DW1	→	OUTPUTPORT 2	LATCH2

Negované signály jsou značeny podtržítkem.

3.7.4 BLOKOVÉ SCHÉMA ZAPOJENÍ



Obr. 23 – Schéma zapojení úlohy 7

3.7.5 PŘÍKLAD ZDROJOVÉHO KÓDU

```
Program Uloha_7;
uses crt, newdelay;
var DS, krokyC, uhelA, uhelB : integer;
    smer, konec : char;
    uhel : real;

(*Procedura pro signalizaci směru k hledanému úhlu*)
Procedure Signalizace(uhelA,uhelB : integer);
var uhelC, signal: integer;
begin
     uhelC:=uhelA-uhelB;

     if uhelC=24 then
     begin
          signal:=231;
     end
     else
     begin
          if abs(uhelC)>24 then
          begin
               if uhelC>0 then
                   uhelC:=(48-uhelC)*(-1)
               else
                   uhelC:=48+uhelC;
          end;

          if uhelC>0 then
          begin
               if uhelC>9 then signal:=7
               else if uhelC>4 then signal:=6
               else if uhelC>2 then signal:=4
               else if uhelC<=2 then signal:=8;
          end
          else
          begin
               if uhelC<-9 then signal:=224
               else if uhelC<-4 then signal:=96
               else if uhelC<-2 then signal:=32
               else if uhelC>=-2 then signal:=16;
          end;
     end;
```

```
     port[$301]:=signal;
end;

(*Procedura pro krokování vlevo*)
procedure Vlevo(krokyC:integer);
var I : integer;
begin
     for I:=0 to krokyC-1 do
     begin
         if DS=10 then DS:=9
         else if DS=9 then DS:=5
         else if DS=5 then DS:=6
         else if DS=6 then DS:=10;
         port[$300]:=DS;
         delay(20);
     end;
end;

(*Procedura pro krokování vpravo*)
procedure Vpravo(krokyC:integer);
var I : integer;
begin
     for I:=0 to krokyC-1 do
     begin
         if DS=10 then DS:=6
         else if DS=6 then DS:=5
         else if DS=5 then DS:=9
         else if DS=9 then DS:=10;
         port[$300]:=DS;
         delay(20);
     end;
end;

begin

     (*Úvodní výpis programu *)
     writeln('MAT-Pridavny modul Krokovy motor');
     writeln('Uloha 7 – Hledani uhlu');
     writeln('Pro spusteni stisknete <Enter>...');
     readln;

     (*Počáteční nastavení proměnných *)
     DS:=9;
     konec:=' ';
     (*Pseoudonáhodné generování úhlú *)
     randomize;
     uhelA:=(random(47)+1);
     Writeln('uhel A: ',uhelA*7.5:4:2);
     uhelB:=(random(47)+1);
     Writeln('uhel B: ',uhelB*7.5:4:2);
     (*Posun na počáteční uhel *)
     Vpravo(uhelB);
     (*Počáteční signalizace směru *)
     Signalizace(uhelA,uhelB);

     repeat
           (*Zadání směru otáčení *)
           repeat
                 write('Zadejte smer otaceni R/L: ');
                 readln(smer);
                 smer:=upcase(smer);
           until (smer='R') or (smer='L');

           (*Zadání úhlu otáčení*)
           repeat
                write('Zadejte uhel (0 - 360): ');
                readln(uhel);
```

```
            uhel:=uhel/7.5;
            krokyC:=round(uhel);
        until (krokyC>=0)and (krokyC<=360);

        (*Realizace otáčení*)
        if smer='L' then
        begin
           (*Otáčení vlevo*)
           Vlevo(krokyC);
           (*Úprava hodnoty zobrazovaného úhlu*)
           uhelB:=uhelB-krokyC;
        end
        else
        begin
           (*Otáčení vpravo*)
           Vpravo(krokyC);
           (*Úprava hodnoty zobrazovaného úhlu*)
           uhelB:=uhelB+krokyC;
        end;

        if uhelB<0 then uhelB:=48+uhelB;
        if uhelB>48 then uhelB:=uhelB-48;

        (*Testování úhlu *)
        if uhelA=uhelB then
        begin
            (*Konečný výpis, dokrokování a případné ukončení programu *)
            writeln('Gratuluji. Hledany uhel ',(uhelA*7.5):3:1,' nalezen.');
            port[$301]:=24;
            konec:='Q';
            writeln('Pro ukonceni programu stisknete <Enter>');
            readln;
            Vpravo(48-uhelB);

        end
        else
        begin
            (*Signalizace směru *)
            Signalizace(uhelA,uhelB);
        end;

    until konec='Q';
end.
```

3.7.6 POPIS PROGRAMU

Na začátku programu je tzv. hlavička programu, kde je název programu, připojené knihovny a deklarované proměnné. *DS* je řídicí datové slovo sloužící k ovládání KRM modulu. Další proměnná je *krokyC*. Do této proměnné se ukládá celkový počet kroků, o který se provede otáčení. Proměnná *uhelA* slouží k uchování hodnoty hledaného úhlu v krocích, který se uživatel snaží nalézt. Proměnná *uhelB* obsahuje hodnotu právě zobrazeného úhlu. Proměnná *smer* je typu char a udává směr otáčení. Další je proměnná *konec* typu char, která slouží k ukončení programu. Poslední deklarovanou proměnnou je *uhel*, který slouží k zadání uhlu, o který se má KRM pootočit a k výpočtu počtu kroků.

Za deklarací proměnných následuje procedura Signalizace. Tato procedura má za úkol pomocí výstupního portu 2 signalizovat směr otáčení a přibližnou vzdálenost hledaného úhlu. Parametry procedury jsou *uhelA* a *uhelB*. Tyto hodnoty udávají úhly v krocích. Z těchto dvou úhlů se zjistí jejich vzdálenost při otáčení doprava. Pokud je větší než 24 kroků, signalizace ukazuje směr otáčení doleva. Naopak pokud je vzdálenost menší, signalizuje se směr doprava. Tímto se zaručí signalizace vždy v kratším směru. Pokud jsou úhly vzdálené přesně 24 kroků, nebo-li 180°, je zobrazena speciální signalizace. Pokud jsou úhly vzdáleny 0 kroků, rozsvítí se další speciální světelná signalizace. Vzdálenost je signalizována počtem rozsvícených diod v určeném směru. Určitému rozsahu vzdáleností je vždy přiřazen určitý počet rozsvícených diod.

Další částí programu jsou dvě procedury Vlevo a Vpravo, které slouží k otočení motoru o daný počet kroků na jednu či druhou stranu. Procedury jsou s parametrem *krokyC*, který udává počet kroků, o který se motor pootočí. Otáčení je realizováno pomocí cyklu *FOR*. V jednom průchodu cyklem se provede právě jeden krok, kdy se nejprve vybere datové slovo *DS* pomocí složené podmínky a pak se toto datové slovo pošle na výstupní port. Tímto se realizuje jeden krok. Počet kroků, které se provedou udává právě parametr *krokyC*.

Na začátku samotného programu jsou funkce, které vypíší na obrazovku základní informace o programu a žádost o stisknutí klávesy Enter. Po stisku klávesy nejprve dojde k počátečnímu nastavení proměnných. Proměnná *DS* se nastaví na hodnotu 9, což je hodnota výchozí polohy na nule. Tím se zajistí, že krokovací sekvence vlevo i vpravo bude začínat správně. Dále se nastaví proměnná *konec*. Do této proměnné můžeme vložit prakticky jakýkoliv znak kromě Q. Znak Q se vkládá do proměnné ve chvíli kdy se má program ukončit. Po nastavení proměnných následuje pseudonáhodné generování hledaného a počátečního úhlu. Úhel obsažený v těchto proměnných je udán v krocích, nikoliv ve stupních. Po vygenerování počátečního úhlu *uhelB* se šipka ukazatele přesune na jeho hodnotu pomocí procedury *vpravo* a spustí se procedura *signalizace*. Následuje žádost o zadání směru otáčení a úhlu, o který se má ukazatel posunout. Zadání směru je ošetřeno tak, aby uživatel mohl zadat pouze požadované hodnoty. Stejně tak je tomu u zadávání úhlu. Jelikož by bylo zbytečné zadávat úhel větší než 360°, je uživatel omezen

v zadávání rozsahem 0° až 360°. Následuje realizace otáčení. Pomocí podmínky se rozhodne, na kterou stranu se bude otáčet. V případě, že se hodnota proměnné *smer* rovná L, realizuje se otáčení vlevo pomocí příslušné procedury. Pak se upraví hodnota zobrazeného úhlu v proměnné *uhelB* tím, že se od stávající hodnoty odečte počet kroků, který se právě provedl. Tím se zaručí, že bude v *uhelB* stejná hodnota, jako je hodnota zobrazená. Obdobně se provede otáčení vpravo. Po dokončení otáčení se ještě testuje hodnota uhelB, zda není větší než 48, což odpovídá úhlu 360°, nebo menší než 0. Pokud hodnota není v rozmezí 0° až 360°, upraví se tak, aby odpovídala stejnému úhlu v tomto rozmezí. Poslední částí programu je testování, zda se zobrazený úhel shoduje s úhlem hledaným. Pokud ne, spustí se procedura *Signalizace*, která zobrazí aktuální informace o poloze hledaného úhlu oproti úhlu zobrazenému, a program se opakuje od místa zadávání směru otáčení. Pokud se ale zobrazený a hledaný úhel shodují, zobrazí se na výstupním portu signalizace shody a vypíše se uživateli na monitor informace o nalezení hledaného úhlu. Následně dojde k nastavení proměnné *konec* na hodnotu Q a program informuje uživatele, že čeká na ukončení klávesou enter. Po stisknutí klávesy ještě dojde k dokrokování na výchozí nulovou pozici a program se ukončí.

3.7.7 ČINNOST PROGRAMU PO SPUŠTĚNÍ

Po spuštění se nejprve vypíšou na obrazovku informace o programu a výzva ke spuštění činnosti programu. Následuje generování úhlů a přesun šipky ukazatele na výchozí úhel, přičemž se zobrazí světelná signalizace na výstupním portu 2, která udává směr a naznačuje vzdálenost hledaného úhlu. Následně je uživatel požádán o zadání směru otáčení a úhlu, o který se má šipka ukazatele posunout. Po zadání hodnot odkrokuje ručička ukazatele o zadaný úhel. Pokud se zobrazovaný úhel shoduje s úhlem hledaným, je uživatel o úspěchu hledání informován výpisem na monitor a světelnou signalizací. Následně je uživatel vyzván k ukončení programu klávesou Enter. Pokud se ale úhel neshoduje, světelná signalizace zobrazí nové informace o směru a přibližné vzdálenosti a program se opakuje od místa zadávání směru otáčení. Před ukončením programu šipka ukazatele odkrokuje na počáteční nulovou pozici.

4 ZÁVĚR

Práce se zabývá krokovým motorem, který se nachází na přídavném modulu stavebnice MAT. Začátek práce je věnován stručnému popisu principu řízení krokového motoru, dále je uveden teoretický rozbor modulu KRM. Byla snaha v práci názorně vysvětlit, jakým způsobem tento modul funguje a jakým způsobem lze řídit jeho činnost. Proto je zde zahrnuta i nezbytná teorie krokových motorů.

Ve druhé části práce je prezentována vytvořená sada úloh, po jejímž vypracování by měl každý student zvládat práci s modulem KRM. Úlohy nejsou příliš náročné, ale jsou koncipovány tak, aby se z nich studenti dokázali naučit něco nového. První úloha je věnována základním principům řízení modulu KRM. V této úloze se má student naučit základy ovládání modulu KRM, jako je změna rychlosti či směru otáčení motoru. Další úloha je věnována ukázce praktického využití krokového motoru jako třídičky odpadů. Třetí úlohou je losovací přístroj, který se po roztočení zastaví na náhodném úhlu. Pro vyšší atraktivnost práce s krokovým motorem je zařazena jako čtvrtá úloha hra. Hra spočívá v soupeření postřehu a vnímání dvou hráčů snažících se strefit zadaný úhel při velké rychlosti otáčení. Při tvorbě programu si student procvičí práci s dalšími částmi stavebnice MAT. V dalších úlohách si studenti vyzkouší programově zvětšit počet kroků, či naprogramovat kompas. Poslední úlohou je program, který pomáhá pomocí světelné signalizace hledat neznámý úhel. Byla snaha vytvářet úlohy co nejatraktivnější. Všechny dodané programy jsou funkční, přehledné a doplněné komentáři.

K práci patří také CD, které obsahuje tuto práci v elektronické podobě, veškeré programy a doplňkové materiály. Mezi programy, jenž jsou na CD, patří sada vypracovaných úloh, úvodní program pro úlohu 1 a dva pomocné programy. Mezi doplňkové materiály patří zapojovací schémata a fotografie stavebnice MAT, modulu KRM a externího zdroje pro modul KRM.

Mým osobním prospěchem z této práce je proniknutí do problematiky krokových motorů. Zpracování této práce navíc pozitivně působilo na můj duševní rozvoj a rozvoj technické představivosti.

5 SEZNAM OBRÁZKŮ

6 SEZNAM TABULEK

7 SEZNAM LITERATURY

[1] STANĚK, Filip. *Výukový Materiál o počítačové stavebnici MAT*. 2009. 47 s. Dostupné z: https://portal.zcu.cz/wps/PA_StagPortletsJSR168/KvalifPraceDownloadServlet?typ=1&adipidno=22293. Diplomová práce. Západočeská univerzita v Plzni Fakulta pedagogická Katedra výpočetní a didaktické techniky. Vedoucí práce Ing. Petr Michalík, Ph.D.

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

[3] NEKVASIL, Vladimír. *Ovládání krokových motorů*: *didaktická pomůcka*. 2008. 31 s. Dostupné z: https://support.dce.felk.cvut.cz/mediawiki/images/7/78/Bp_2008_nekvasil_vladimir.pdf. Bakalářská práce. České vysoké učení technické v Praze Fakulta elektrotechnická Katedra řídicí techniky. Vedoucí práce Ing. Jan Havlík.

[4] E&L INSTRUMENTS. *Stepper motor lab manual*. New Haven, Connecticut, 1942, 56 s. 80-01-0474.

[5] UHLÍŘ, Ivan. *Elektrické stroje a pohony*. Vyd. 1. Praha: Vydavatelství ČVUT, 2002, 120 s. ISBN 80-010-2482-2.

[6] SKALICKÝ, Jiří. *Elektrické servopohony*. 1. vyd. Brno: VUT v Brně, 1999, 86 s. ISBN 80-214-1484-7.

[7] MICHALÍK, Petr, Zdeněk ROUB a Václav VRBÍK. *Zpracování diplomové a bakalářské práce na počítači*. 3. vyd. V Plzni: Západočeská univerzita, 2009, 67 s. ISBN 978-80-7043-828-2.

[8] POLÁCH, Eduard. *Programování v jazyku Turbo Pascal* [online]. [cit. 2012-06-06]. Dostupné z: http://home.pf.jcu.cz/~edpo/program/program.html

[9] VRBÍK, Václav. *Programování 1.* 1. vyd. Plzeň: Západočeská univerzita, Pedagogická fakulta, 2000, 177 s. ISBN 80-708-2663-0.

[10] Vlastni poznámky z cvičeni předmětu TCHP2

8 RESUMÉ

This bachelor´s work describes additional module to MAT trainer kit. I took this work as potential material for teaching students or creating electronic educational material. In this work you can find basic principles of stepper motors and description of stepper motor module. In the next part of this work there are seven tasks for stepper motor module. Every task contains instructions and possible solution with detailed description.

PŘÍLOHY

Obsah přiloženého CD

1. Bakalářská práce v elektronické podobě
2. Úloha 1 zadání
3. Sada vypracovaných úloh
4. Zapojovací schémata
5. Pomocné programy
6. Fotografie