UNIVERZA V LJUBLJANI

FAKULTETA ZA
RAČUNALNIŠTVO IN INFORMATIKO

Rok Horjak

Omarica za DVD-je z mikrokontrolerjem PIC

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: izr. prof. dr. Branko Šter

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00525/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ROK HORJAK**

Naslov: **OMARICA ZA DVD-JE Z MIKROKONTROLERJEM PIC**
DVD CABINET WITH PIC MICROCONTROLLER

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Izdelajte omarico za DVD-je s pomočjo mikrokontrolerja PIC16F84. Opišite mikrokontroler PIC16F84 ter druge uporabljene komponente, kot so koračni motor, DC motor in matrična tipkovnica. Opišite tudi razvojno okolje MPLAB. Predstavite programske rešitve posameznih nalog.

Mentor:

prof. dr. Branko Šter

Fakulteta za računalništvo in informatiko
UNIVERZA V LJUBLJANI
2

Dekan:

prof. dr. Franc Solina

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Rok Horjak,

z vpisno številko 63040215,

sem avtor diplomskega dela z naslovom:

Omarica za DVD-je z mikrokontrolerjem PIC

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom (naziv, ime in priimek)

  izr. prof. dr. Branko Šter

  in somentorstvom (naziv, ime in priimek)

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 5.6.2010 Podpis avtorja/-ice: _____

ZAHVALA

Zahvaljujem se svojemu mentorju Branku Šteru za strokovno vodenje, odzivnost, trud in čas, katerega je vložil v mojo diplomsko nalogo, in mi z vsestransko podporo pomagal pri pravočasnem dokončanju le-te.

Največja zahvala pa gre mami Mariji Horjak in sestri Lei Horjak za vso pomoč, podporo in skrb, katero sem bil deležen skozi vsa ta leta. Vajina opora, potrpežljivost in spodbuda, ki sta mi jo nudili, je razlog, da sem lahko dosegel svoje cilje.

Posebna zahvala gre tudi Petru Mlakarju, ki mi je pri izdelavi te diplomske naloge in na splošno skozi ves študij na takšen ali drugačen način priskočil na pomoč, ko sem le-to potreboval.

Iskreno se vam vsem zahvaljujem!

KAZALO VSEBINE

SEZNAM UPORABLJENIH KRATIC

CPU	ang. Central Processing Unit – Centralna procesna enota
DC	ang. Direct Current – Enosmerni tok
DVD	ang. Digital Versatile Disc – Mnogonamenski digitalni disk
EEPROM	ang. Electrically Erasable Programmable Read Only Memory – Električno izbrisljiv programirljiv bralni pomnilnik
EPROM	ang. Erasable Programmable Read Only Memory – Izbrisljiv programirljiv bralni pomnilnik
GPIO	ang. General Purpose Input/Output – Splošno nameski vhodi/izhodi
GPR	ang. General Purpose Register – Splošno namenski register
HB	ang. Hybrid - Hibrid
IDE	ang. Integrated Development Environment – Integrirano razvojno okolje
ISR	ang. Interrupt Service Routine – Prekinitveno strežni program
LED	ang. Light Emitting Diode – Svetleča dioda
PIC	ang. Programmable Interface Controller – Programirljiv vmesniški nadzornik
PM	ang. Permanent Magnet – Stalni magneti
RAM	ang. Random Access Memory – Pomnilnik s statičnom časom dostopa
RBIF	ang. Registry B Interrupt Flag – Prekinitvena zastavica registra B
ROM	ang. Read Only Memory – Bralni pomnilnik
SFR	ang. Special Function Register – Register za posebne funkcije
V/I naprave	Vhodno/Izhodne naprave
VR	ang. Variable Reluctance – Spremenljivo magnetno nasprotovanje

POVZETEK

Velikokrat se jezimo nad problemi, katere lahko rešimo. K sreči živimo v času, kjer nam je tehnologija, s katero lahko rešimo te probleme, splošno na voljo. Te rešitve lahko kupimo ali pa v primeru, da jih še ni na trgu oz. da so predrage, rešimo sami. Slednje sem storil tudi sam, ko sem naletel na problem velike količine DVD-jev (ang. Digital Versatile Disc), brez primernega sistema za hitro in učinkovito iskanje le-teh. Odločil sem se, da naredim omarico, ki nam glede na vneseno številko preko matrične tipkovnice in s pomočjo koračnega motorja ter DC (ang. Direct Current) motorja poišče in izstavi DVD iz omarice.

Opisal sem vsako od ključnih komponent, saj si brez osnovnega predznanja in razumevanja, ne moremo predstavljati, po kakšnem principu deluje celoten sistem. Na začetku sem opisal osnove mikrokontrolerjev, zatem pa še bolj podrobno mikrokontroler PIC16F84 (ang. Programmable Interface Controller), katerega sem uporabil pri izdelavi omarice. Pri izdelavi sem uporabil dva mikrokontrolerja zaradi večje zanesljivosti ter same demonstracije prenosa podatkov med dvema mikrokontrolerjema z preprostim protokolom. V nadaljevanju sem opisal delovanje koračnega motorja, katerega sem uporabil za premikanje mehanizma za iskanje DVD-jev, delovanje DC motorja, za izmet le-teh in matrično tipkovnico, za vnos zaporednega števila želenega DVD-ja.

V nadaljevanju sem opisal še program MPLAB, katerega sem uporabil za pisanje programa za nadzor naprav v omarici. Opisal sem tudi posamezne naloge obeh mikrokontrolerjev ter programske rešitve posameznih nalog.

Ključne besede: PIC mikrokontroler, koračni motor, matrična tipkovnica, MPLAB, zbirni programski jezik.

ABSTRACT

We are often bothered by solvable problems. Luckily we live in a time, when technology, that can be used to solve these problems, is readily available. These solutions can be purchased, or in case of them not being widely available or too expensive, solved and made by ourselves. I chose to do the latter, when I faced a problem of large quantities of DVDs, without an appropriate system for a fast and efficient way to search for them. I decided to make a cabinet, that would give us the correct DVD, according to the number received via a matrix keypad while using a stepper and a DC motor for the delivery.

I described each of the key components, because without the proper knowledge and understanding of mentioned components, one cannot understand the system as a whole. At first I described microcontrollers in general and then the PIC16F84 microcontroller, which I used in the development of this cabinet, in more detail. I used two microcontrollers for greater reliability and to demonstrate a simple and effective data transfer between two microcontrollers with a simple protocol. Following that I described the operation of a stepper motor, which I used to move the mechanism for finding DVDs, the operation of a DC motor, for handing over the DVD and the operation of a matrix keypad to enter the number for the desired DVD.

I also described the program MPLAB, which I used to write a program to control the devices in the cabinet. In addition the specific tasks of both microcontrollers and software solutions to various tasks were also described.

Keywords: PIC microcontroller, stepper motor, matrix keypad, MPLAB, assembler programming language.

1. UVOD

Že pri izbiri primerne teme za svojo diplomsko delo sem si zadal, da se bo vse vrtelo okoli nekaj otipljivega in v vsakdanjem življenju uporabnega. S tem ciljem v mislih sem pogledal vsakdanje probleme in nevšečnosti, s katerimi se srečujem in katerih rešitev bi bila primerna za diplomsko delo. Kmalu mi je pogled obstal na precejšnem kupu DVD-jev, kar mi je dalo idejo za omaro z avtomatskim iskanjem in izdajo DVD plošč, s katero bi se mi skrajšal čas iskanja pravilnega DVD-ja.

Problemov, s katerimi se soočimo že od samega začetka, je seveda precej. Od vprašanja kako nam omarica poda DVD, preko tega kako sprejme zahtevo in najde iskani DVD, do izbire komponent in sestavo le-teh. Da bi se izognil problemu zanesljivosti sem uporabil dva mikrokontrolerja katera medsebojno komunicirata preko preprostega protokola.

Prvi del diplomske naloge je teoretične narave, v katerem se bralec do dobrega seznani z mikrokontrolerji, bolj natančno z mikrokontrolerjem PIC16F84. V tem delu opišem najpomembnejše dele mikrokontrolerja ter interakcijo med njimi.

V drugem delu se bralec seznani z osnovo delovanja vseh ključnih komponent, kot so koračni motor, DC motor in matrična tipkovnica, o njihovih prednostih in slabostih, ter o delovanju komponent v sistemu, saj je to izjemnega pomena za razumevanje delovanja celotnega sistema.

Tretji del opisuje program MPLAB, katerega sem uporabil za programiranje, delovanje in naloge posameznega mikrokontrolerja, probleme, na katere je potrebno paziti, z njihovimi rešitvami, ter protokol za komunikacijo med obema mikrokontrolerjema.

Cilj diplomske naloge je pridobljeno znanje v času študija strniti v praktični projekt, kateri bo zanimiv in primeren za vse ostale, ki jih to področje zanima, in bralca seznaniti s potrebnimi komponentami in težavami, na katere bi lahko naletel pri podobnih projektih. Bralec bi po končanem branju moral biti seznanjen z osnovnimi postopki programiranja PIC mikrokontrolerjev ter s prednostmi in omejitvami, na katere naletimo pri uporabi mikrokontrolerjev in komponent, ki spadajo zraven.

2. MIKROKONTROLER

2.1. Splošno

Mikrokontroler je integrirano vezje, katerega lahko razumemo kot svoje vrste zmanjšan računalnik, ki ga najdemo v številnih vsakodnevnih napravah. Nekaj primerov vsakodnevnih naprav, ki vsebujejo mikrokontroler, so budilka, mikrovalovna peč, mobilni telefon ter mnogi drugi. Ko se zbudimo in izklopimo budilko, si segrejemo zajtrk v mikrovalovki ali se pogovarjamo s svojimi bližnjimi po telefonu, v vseh teh in mnogo drugih primerih uporabljamo oz. koristimo mikrokontrolerje.

V osnovi vsak mikrokontroler vsebuje elemente, ki so centralna procesna enota, pomnilnik, ki je lahko ROM (ang. Read Only Memory), EPROM (ang. Erasable Programmable Read Only Memory), EEPROM (ang. Electrically Erasable Programmable Read Only Memory) ali Flash za shranjevanje programa ter programirljive V/I (Vhodno/Izhodne) zatiče. Program v mikrokontrolerju ne sme biti preobširen, saj moramo biti pozorni, da ne presežemo pomnilnika, katerega uporablja mikrokontroler, ker bi dodatni zunanji pomnilnik znatno povečal finančni vložek, otežil programiranje in v veliko primerih pomenil počasnejše izvajanje programa.

Ker se morajo mikrokontrolerji odzivati na zunanje dogodke v realnem času, je lahko potrebno, da se izvede prekinitev trenutnega zaporedja ukazov in se začne izvajati ISR (ang. Interrupt Service Routine). V ISR se bo izvedelo zaporedje potrebnih ukazov, po katerem se bo prekinjeni program ponovno pričel izvajati.

Zelo uporabna opcija mikrokontrolerja je tudi, da vsebuje zatiče GPIO (ang. General Purpose Input/Output), katere se programsko nastavi, da so vhodni ali izhodni. Vhodne zatiče se načeloma uporablja za branje iz senzorjev ali drugih zunanjih signalov. Izhodne zatiče pa lahko uporabimo za prižiganje LED (ang. Light Emitting Diode) diode ali vklapljaje motorjev in drugih zunanjih naprav.

Mikrokontrolerji so bili zelo popularni že v 70ih letih, ko so bili prvič predstavljeni, saj z njihovo uporabo občutno zmanjšamo število potrebnih elementov ter s tem tudi velikost ter ceno celotnega vezja. S predstavitvijo pomnilnika EEPROM, v začetku 90ih let, je brisanje pomnilnika na mikrokontrolerjih postalo preprostejše ter s tem poenostavilo in pohitrilo izdelavo prototipov. Zaradi padanja cen mikrokontrolerjev in preostalih komponent ter preprostosti

programiranja, o katerem bomo govorili v 4. poglavju, se mikrokontrolerji dandanes uporabljajo v precejšnem številu, tudi v učnih ustanovah ali od osebe, ki se s tem ukvarjajo v prostem času.

2.2. Mikrokontroler PIC16F84

PIC16F84 spada v skupino 8-bitnih mikrokontrolerjev, ki imajo visoko zmogljivost in relativno nizko ceno. Proizvaja jih podjetje Microchip Technology Inc. iz Združenih držav Amerike, ki so leta 1993, z uvedbo EEPROM v mikrokontrolerjih in drugih izvirnih rešitev, zagotovili in prevzeli večinski svetovni tržni delež. Zaradi vsestranske uporabnosti in nizke cene so le-ti postali zelo popularni. K popularnosti in splošni dostopnosti pripomoreta tudi dejstvi, da to podjetje vsakega uporabnika oskrbi z vso potrebno literaturo, ter da je internetna skupnost na tem področju precej močna.

2.2.1.Pomnilnik

PIC16F84 je 8-bitni mikrokontroler z vgrajenim flash programskim pomnilnikom velikosti 1024 x 14 bitov. Ker je vsak ukaz dolg 14 bitov in je prvih pet lokacij zasedenih, to pomeni, da vanj lahko shranimo 1020 ukazov oz. 1020 besed programske kode, ki jo izvaja mikrokontroler. Vgrajen ima tudi RAM (ang. Random Access Memory) velikosti 64 x 8 bitov, katerega uporabljamo za shranjevanje podatkov o zatičih, skladu, števcih in zastavicah, ter EEPROM velikosti 64 x 8 bitov za shranjevanje vektorjev in vrednosti za izvajanje programa. [8]

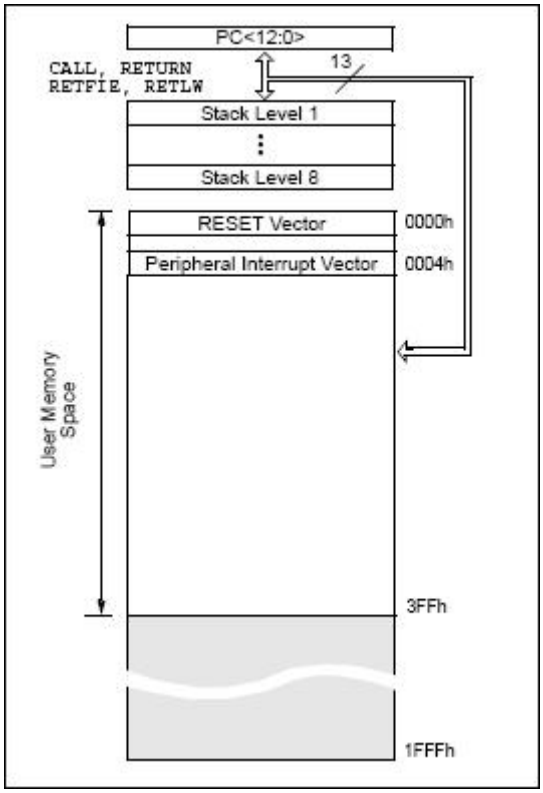
Ta mikrokontroler ima 18 zatičev; od tega jih je 13 V/I, kateri so individualno uporabniško nastavljivi. Nekateri od teh zatičev so multipleksirani z drugimi funkcijami; kot so na primer zunanja prekinitev, prekinitev ob spremembi na PORTB in vnos ure Timer0. Podrobnejši opis zatičev je v tabeli 1.

Pri PIC16F84 imamo programski pomnilnik in podatkovni pomnilnik.

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Tabela 1: Opis zatičev na PIC16F84. [8]

Programski pomnilnik je sestavljen iz 1024 lokacij (od 0x000 do 0x3FF), pri čemer je prvih pet lokacij rezerviranih. Posebno vlogo imata prva in peta lokacija, saj je prva rezervirana za reset, peta pa za prekinitveni vektor. Vsak ukaz skupaj s parametri v PIC16F84 zavzame natanko eno lokacijo dolžine 14 bitov. Podrobnejša predstavitev programskega pomnilnika je na sliki 1.



Slika 1: PIC16F84 programski pomnilnik. [8]

Podatkovni pomnilnik lahko razdelimo na SFR (ang. Special Function Register) in GPR (ang. General Purpose Register). GPR je širok 8 bitov in ga delimo na dve banki, ki se med seboj zrcalita tako, da se pri vpisu podatka na lokacijo v banki 0, pojavi podatek tudi v banki 1. Uporablja se za shranjevanje splošnih podatkov, katere potrebujemo. SFR uporabljajo CPU (ang. Central Processing Unit) in periferne funkcije za nadzor delovanja naprave. SFR zato lahko razdelimo na dva dela; prvega uporablja izključno CPU drugega pa periferne funkcije. [4]

2.2.2. V/I zatiči

V/I zatiči so programsko nastavljivi na vhodni ali izhodni. Nekateri V/I zatiči so multipleksirani za periferne funkcije. Načeloma v primeru, da zatič uporablja periferna funkcija, ne more biti istočasno tudi uporabljen kot V/I zatič. Vse V/I zatiče delimo v dve skupini: PORTA in PORTB.

PORTA ima pet zatičev (od RA0 do RA4). Register, s katerim upravljamo PORTA, je velik 8 bitov in se imenuje TRISA. Če nastavimo bit v TRISA na logično 1, bo ustrezen zatič postal vhod, v nasprotnem primeru, če ga postavimo na logično 0, postane ustrezen zatič izhod. Na ta način, če preberemo vrednost v registru TRISA, lahko ugotovimo, kakšne so nastavitve zatičev skupine PORTA. Zatič RA4/TOCKI je edini v PORTA, ki je multipleksiran z vhodno uro Timer0. Slika 2 prikazuje preprost primer inicializacije zatičev v PORTA.

PORTB ima 8 zatičev (od RB0 do RB7). Kot PORTA ima tudi PORTB register velikosti 8 bitov za upravljanje zatičev, vendar se ta imenuje TRISB. Ravno tako v TRISB nastavljen bit na logično 1, povzroči postavitev ustreznega zatiča na vhod in nastavitev logične 0 na izhod. Če nas zanima nastavitev zatičev v PORTB, preberemo vrednost registra TRISB. Obstajajo štirje zatiči v PORTB; RB4, RB5, RB6 in RB7, kateri imajo še dodatno funkcijo prekinitev-ob-spremembi. To prekinitev lahko sprožijo le zatiči, ki so nastavljeni kot vhod. Trenutni vhodi na teh zatičih se primerjajo z nazadnje prebranimi vhodi. V primeru spremembe kateregakoli od teh zatičev se zastavica RBIF (ang. Registry B Interrupt Flag) postavi na logično 1, kar sproži prekinitev. To prekinitev lahko ustavimo v ISR, s ponovnim branjem omenjenih zatičev oz. s postavitvijo RBIF zastavice na logično 0. Ta vrsta prekinitve se priporoča za uporabo pri prebujanju oz. aktivaciji naprave ob pritisku tipke. Primer nastavitev zatičev RB7, RB6, RB3, RB2 in RB0 kot vhode in RB5, RB4 in RB1 kot izhode vidimo na sliki 3.

```
bcf STATUS, RP0
clrf PORTA          ;postavimo vse vrednosti, v registru PORTA, na 0
bsf STATUS, RP0     ;v registru STATUS postavimo bit RP0 na 1 in s tem izberemo
                    BANK1
movlw 0x0F          ;v register W prenesemo vrednost 0b0000 1111
movwf TRISA         ;v register TRISA prenesemo vsebino registra W
```

Slika 2: Inicializacija zatičev v PORTA.

```
bcf STATUS, RP0
clrf PORTB          ;postavimo vse vrednosti, v registru PORTB, na 0
bsf STATUS, RP0     ;v registru STATUS postavimo bit RP0 na 1 in s tem izberemo
                    BANK1
movlw 0xCD          ;v register W prenesemo vrednost 0b1100 1101
movwf TRISB         ;v register TRISA prenesemo vsebino registra W
```

Slika 3: Inicializacija zatičev v PORTB.

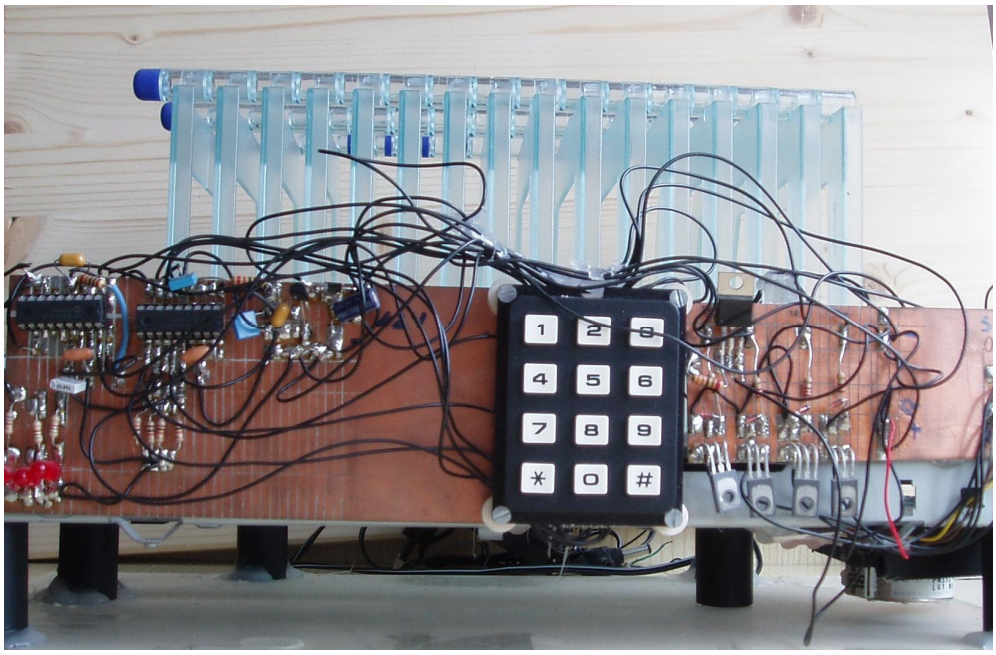
3. VHODNE IN IZHODNE NAPRAVE

Omarica za DVD-je je sestavljena iz treh poglavitnih delov. Iz police za DVD-je, ohišje, po katerem se premika koračni in DC motor ter plošče s celotnim vezjem.

Police za DVD-je nisem izdelal sam, temveč sem jo kupil. Izdelava lastne ni bila potrebna in tudi ni priporočljiva, saj mora biti zelo natančno izdelana z enakomernimi razdaljami med posameznimi DVD-ji.

Zaradi lažje izdelave in bistveno večje natančnosti sem ohišje, po katerem se premikata koračni in DC motor, vzel iz starega tiskalnika. Ohišje je bilo potrebno precejšne obdelave, saj je le-to centralni del omarice, na katero je pritrjeno celotno vezje kot tudi police za DVD-je.

Kot vidimo na sliki 4 je na zadnji strani omarice pritrjeno celotno električno vezje z obema mikrokontrolerjema, s katerima skrbimo za upravljanje koračnega in DC motorja, za preverjanje stanj tipk na matrični tipkovnici, kot tudi vse povezave med vsemi perifernimi komponentami.



Slika 4: Zadnja stran omarice.

3.1. Koračni motor

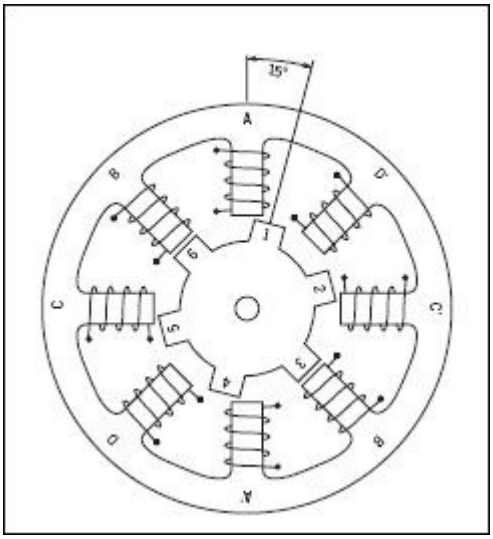
Koračni motor je elektromehanska naprava, ki pretvarja električne impulze v diskretno gibanje. Os koračnega motorja se vrti v diskretnih korakih, ko so električni impulzi poslani v pravilnem vrstnem redu. Z vrstnim redom električnih impulzov nadzorujemo smer vrtenja osi, hitrost

nadzorujemo s frekvenco pošiljanja impulzov ter trajanje vrtenja nadzorujemo z številom vhodnih impulzov.

Koračni motor izberemo, ko potrebujemo dobri nadzor gibanja, kot so rotacijski kot, hitrost vrtenja ter sinhrono delovanje mnogih motorjev hkrati. Zaradi številnih prednosti se koračni motorji pojavljajo v številnih aplikacijah; kot so tiskalniki, trdi diski, DVD predvajalniki in druge natančne naprave.

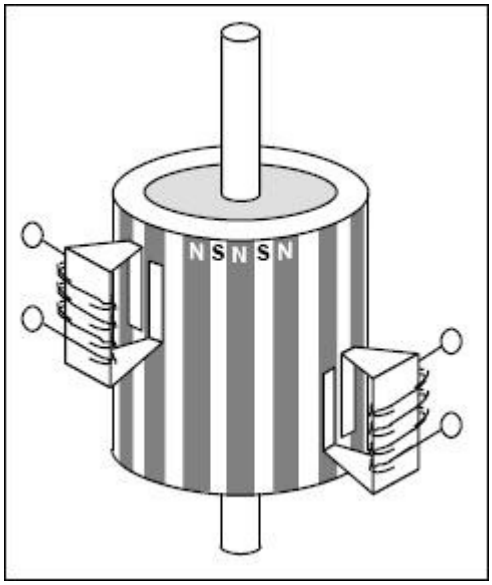
Prednosti koračnega motorja so: zelo dober nadzor rotacijskega kota, natančnost in ponovljivost gibanja, hitra odzivnost na signale, visoka zanesljivost, preprostost upravljanja motorja ter širok razpon hitrosti vrtenja. Pomanjkljivosti pa so: pri neustreznem nadzoru lahko pride do resonance, prav tako je težko nadzorovati motor pri zelo visokih hitrostih.

Poznamo tri vrste koračnih motorjev VR (ang. Variable Reluctance), PM (ang. Permanent Magnet) in HB (ang. Hybrid). VR vrsta koračnega motorja je najdlje v uporabi in tudi najbolj razumljiva. Kot vidimo na sliki 5, ki prikazuje prerez tipičnega VR koračnega motorja, je ta tip sestavljen iz nazobčanega kolesa in statorskega navitja. Ko je statorsko navitje pod napetostjo, poli postanejo namagneteni, kar privlači zob kolesa ter tako povzroči vrtenje.



Slika 5: Prerez VR koračnega motorja. [15]

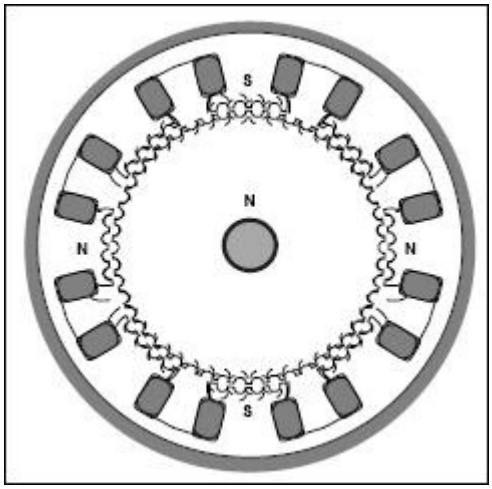
Koračni motor vrste PM je nizkocenovni tip motorja z običajnimi koraki med 7,5 in 15,0 stopinjami. [15] Tip PM ima stalne magnete in nima nazobčanega kolesa kot tip VR. Kot vidimo na sliki 6, ima tip PM vzporedno z rotacijsko osjo polja, katera izmenično polariziramo, kar povzroči povečanje magnetnega toka in s tem vrtenje osi. HB vrsta koračnega motorja je dražja od tipa PM, vendar ima boljše karakteristike kot so npr. hitrost in navor motorja ter da so običajni koraki med 0,9 in 3,6 stopinjami. HB tako združuje najboljše lastnosti VR in PM tipov



Slika 7: Prerez PM koračnega motorja. [15]

koračnega motorja. HB tip ima, kot prikazuje slika 7, nazobčano kolo kot VR tip koračnega motorja in magnet okoli svoje osi kot tip PM. Zobje na kolesu zagotovijo natančnejšo usmeritev magnetnega toka, zaradi česar imamo boljši nadzor ter večji navor kot pri tipih VR in PM. [15]

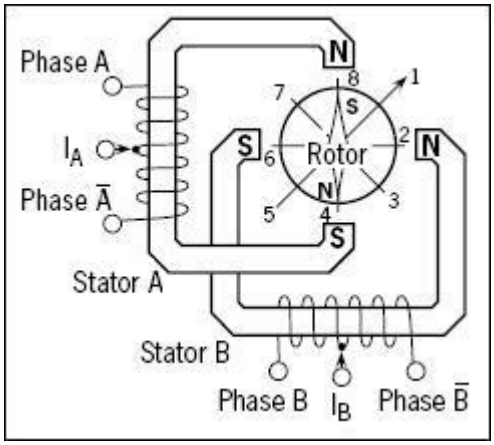
Glede na izvedbo tuljave ločimo koračne motorje še na unipolarne in bipolarne. Koračni motor



Slika 6: Prerez HB koračnega motorja. [15]

naredi korak, ko se spremeni magnetno polje v tuljavi, vendar je razlika med unipolarnimi in bipolarnimi motorji v tem, kako to spremembo dosežemo.

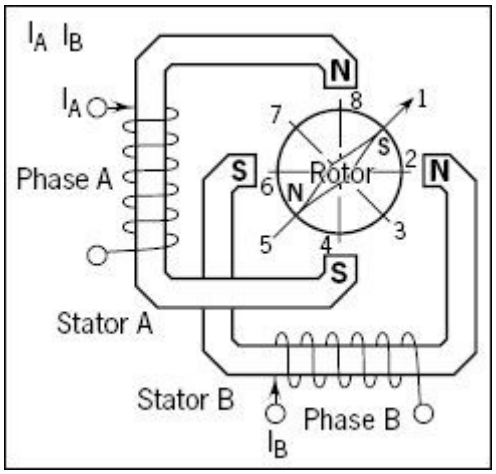
Unipolarni koračni motorji so večinoma povezani, kot je prikazano na sliki 8, s centralno žico v sredini tuljave. Centralna žica je povezana na visoko napetost, oba konca tuljave pa na ozemljitev. Za spremembo magnetnega polja v tuljavi spremenimo smer električnega toka,



Slika 8: Povezava Unipolarnega koračnega motorja. [15]

katero dosežemo tako, da zamenjamo konec tuljave. Glavna prednost unipolarnega koračnega motorja je v preprostosti izdelave in posledično v ceni samega motorja. [3]

Bipolarni koračni motor prav tako vsebujejo tuljavo, vendar brez centralne žice, kot lahko vidimo na sliki 9. Tako je spremembo magnetnega polja v tuljavi težje doseči, saj potrebujemo preklopno stikalo, ki izmenično preklaplja smer električnega toka. Prednost bipolarnega koračnega motorja leži v tem, da so tuljave bolje izkoriščene in močnejše od unipolarnih.



Slika 9: Povezava Bipolarnega koračnega motorja. [15]

Koračne motorje lahko krmilimo na več različnih načinov, od tega se najpogosteje uporabljajo polni korak, polovični korak ter valovni način. V tabeli 2 vidimo zaporedje aktivacij tuljav za vsak način posebej. [3]

V načinu polnega koraka sta v vsakem trenutku aktivni dve tuljavi. Tuljave so aktivirane v zaporedju: A¬B → ¬A¬B → ¬AB → AB, pri čemer so položaji osi: 1 → 3 → 5 → 7. Prednost polnega koraka je v velikem navoru, slabost pa je v precejšnem vibriranju.

Valovni način ima isto število korakov kot način polnega koraka. Z razliko od polnega koraka imamo v valovnem načinu aktivirano le eno tuljavo naenkrat in to v zaporedju A → ¬B → ¬A → B s položaji na 8 → 2 → 4 → 6. Zaradi izjemne podobnosti z načinom polnega koraka in manjšim navorom se ta način le redko kdaj uporablja.

Slabost polnega koraka in valovnega načina odpravimo z uporabo načina polovičnega koraka. Način polovičnega koraka združuje valovni način in polni korak, pri katerem se os ustavi na

	Polni korak				Valovni način				Polovični korak			
	A	B	¬A	¬B	A	B	¬A	¬B	A	B	¬A	¬B
1	X	X			X				X	X		
2		X	X			X				X		
3			X	X			X			X	X	
4	X			X				X			X	
5	X	X			X						X	X
6		X	X			X						X
7			X	X			X		X			X
8	X			X				X	X			

Tabela 2: Način vrtenja koračnih motorjev.

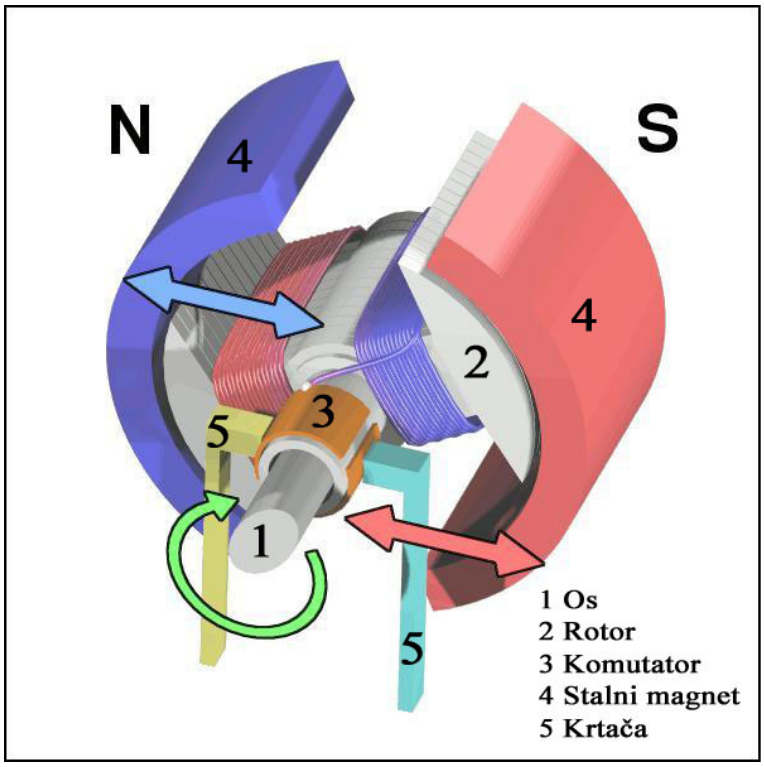
vsakem položaju. Polovični korak ima dvojno število korakov kot valovni način. Z izmenično aktivacijo med dvema tuljavama ter ene tuljave dosežemo, da delamo pol manjše korake kot pri ostalih načinih in s tem zmanjšamo vibriranje motorja in kot vrtenja ter s tem povečamo natančnost motorja.

3.2. DC motor

DC motor je električni motor, ki deluje na enosmernem toku. Vsak električni motor temelji na osnovah elektromagnetizma, tako deluje tudi DC motor. Kot vemo, se magneti z isto polarnostjo

odbijajo in z različno polarnostjo privlačijo. Prav ta princip delovanja uporablja DC motor med stalnim magnetom pritrjenim na stator ter električnim magnetom na rotorjih.

Kot vidimo na sliki 10, DC motor vsebuje šest osnovnih komponent: os, rotor, stator, komutator, stalne magnete in krtačo. Komutator, krtača in električni magneti na rotorju potrebujejo napetost za delovanje. [5] Zaradi iste polarnosti električnega magneta in stalnega magneta se le-ta medsebojno odbijata, kar povzroči vrtenje rotorja. Ko se približamo drugemu stalnemu magnetu, se krtača prestavi na naslednji komutatorjev stik in povzroči aktivacijo naslednjega električnega magneta, kar sproži ponovitev celotnega postopka.



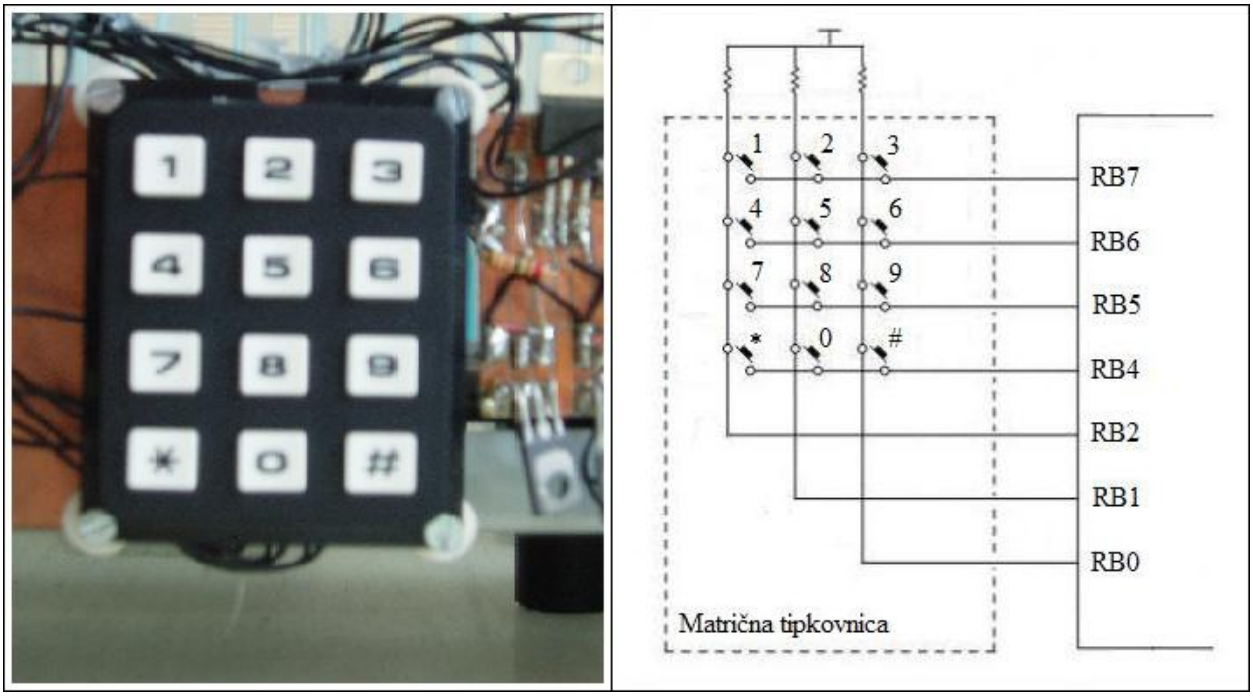
Slika 10: Sestavni deli DC motorja. [11]

Eden od problemov, na katerega naletimo pri DC motorjih, je visoka hitrost vrtenja. Samo nadzorovanje hitrosti vrtenja DC motorja je zahtevno in drago, zato za rešitev tega problema uporabimo kolesa različnih dimenzij, s katerimi lahko poljubno pomanjšamo hitrost končnega vrtenja.

3.3. Matrična tipkovnica

Matrična tipkovnica je serija stikal oz. tipk. V našem primeru sem se odločil za tipkovnico z dvanajstimi tipkami, ki so razdeljene na tri stolpce in štiri vrstice. Za uporabo vseh dvanajstih tipk potrebujemo sedem zatičev na mikrokontrolerju, tri izhode in štiri vhode, kjer izhode

uporabimo za določitev stolpcev in vhode za določitev vrstic. Na sliki 11 vidimo shemo povezave med matrično tipkovnico in mikrokontrolerjem, ter matrično tipkovnico, katero sem uporabil pri izdelavi diplomskega dela.



Slika 11: Shema povezave med matično tipkovnico in mikrokontrolerjem.

Ob pritisku tipke se naredi stik med ustreznim stolpcem in vrstico, kar povzroči, da se postavi vrstica, v kateri je bila pritisnjena tipka, na visoko stanje. Da vemo, katera tipka je bila pritisnjena v določeni vrstici, moramo vedeti, v katerem stolpcu leži, zato nimamo vseh stolpcev istočasno v visokem stanju, vendar jih v določenem vrstnem redu vklapljamo in izklapljamo. Ker v vsakem trenutku vemo, kateri stolpec je aktiven in ob pritisku tipke vemo v kateri vrstici le-ta leži, lahko s tema dvema podatkoma natanko določimo katera tipka je bila pritisnjena[1, 10].

Podrobnejši opis posameznih korakov ter programsko rešitev tega problema si lahko ogledamo v poglavju 4.2.1.

4. NADZORNI PROGRAM

V tem poglavju si bomo pogledali potek programiranja obeh mikrokontrolerjev in možne probleme, na katere lahko naletimo. Vsi PIC mikrokontrolerji podpirajo številna razvojna orodja. Za potrebe tega diplomskega dela se bom omejil le na opis razvojnega orodja MPLAB, katerega sem uporabljal tudi sam.

4.1. MPLAB IDE

MPLAB IDE (ang. Integrated Development Environment) je brezplačni program za programiranje, simuliranje in emuliranje PIC mikrokontrolerjev. MPLAB IDE je 32-bitna aplikacija za operacijski sistem Microsoft Windows v katerem lahko programiramo mikrokontrolerje v zbirnem programskem jeziku ali programskem jeziku C. MPLAB IDE nam tudi dovoljuje uporabo več različnih orodji za razhroščevanje programa, kar nam omogoča enostavno prestavitev iz simuliranja na emuliranje.

4.2. Naloge prvega mikrokontrolerja

Prvi mikrokontroler ima dve glavni nalogi. Prva naloga je, da konstantno preverja vnesene signale matrične tipkovnice, jih pretvori v število ter to število shrani za nadaljnjo uporabo. Druga naloga pa je, da vneseno število enakomerno prenese drugemu mikrokontrolerju v razumljivem formatu oz. po pravilnem protokolu.

4.2.1. Nastavitev in branje matrične tipkovnice

Za preverjanje tipk na matrični tipkovnici potrebujemo 7 zatičev na mikrokontrolerju, od tega bodo trije izhodi, za katere sem izbral od RB0 do RB2 in štirje vhodi, od RB4 do RB7. Zatič RB3 pa je uporabljen za pošiljanje signalov drugemu PIC mikrokontrolerju. Za pravilno delovanje je potrebno zatiče ustrezno nastaviti. Potrebna koda je vidna na sliki 12.

```
clrf PORTB          ;izbrišemo vsebino registra RORTB
bsf STATUS,RP0      ;bit RP0 v registru STATUS postavimo na 1
movlw 0xf0          ;v register W zapišemo 0xf0
movwf TRISB         ;vsebino registra W zapišemo v register TRISB
bcf STATUS,RP0      ;bit RP0 v registru STATUS postavimo na 0
```

Slika 12: Nastavitev V/I zatičev na prvem mikrokontrolerju.

Tipke na matrični tipkovnici beremo v glavni zanki. Kot vidimo na sliki 13, začnemo z aktivacijo tretjega stolpca, ki vsebuje tipke za 3, 6 in 9, kjer preverimo vsako vrstico posebej, nato izklopimo tretji stolpec in aktiviramo drugi stolpec, ki vsebuje 2, 5, 8 in 0, ponovno izklopimo aktivni stolpec in aktiviramo prvi stolpec, za števila 1, 4 in 7. Ko so vsa števila pregledana, se izklopi prvi stolpec in glavna zanka se ponovno prične izvajati od začetka.

```
; Preverjanje pritisnjene tipke
Preveri_3           ;podprogram Preveri_3
    btfsc PORTB,4   ;preverimo zatič RB4 če je pritisnjen
    call Vpisi_3    ;če je pritisnjen kliči podprogram Vpiši_3
    return          ;če ni pritisnjen se vrni nazaj
Preveri_6           ;podprogram Preveri_6
    btfsc PORTB,5   ;preverimo zatič RB5 če je pritisnjen
    call Vpisi_6    ;če je pritisnjen kliči podprogram Vpiši_6
    return          ;če ni pritisnjen se vrni nazaj
Preveri_9           ;podprogram Preveri_9
    btfsc PORTB,6   ;preverimo zatič RB6 če je pritisnjen
    call Vpisi_9    ;če je pritisnjen kliči podprogram Vpiši_9
    return          ;če ni pritisnjen se vrni nazaj
Preveri_2           ;podprogram Preveri_2
    btfsc PORTB,4   ;preverimo zatič RB4 če je pritisnjen
    call Vpisi_2    ;če je pritisnjen kliči podprogram Vpiši_2
    return          ;če ni pritisnjen se vrni nazaj
Preveri_5           ;podprogram Preveri_5
    btfsc PORTB,5   ;preverimo zatič RB5 če je pritisnjen
    call Vpisi_5    ;če je pritisnjen kliči podprogram Vpiši_5
    return          ;če ni pritisnjen se vrni nazaj
Preveri_8           ;podprogram Preveri_8
    btfsc PORTB,6   ;preverimo zatič RB6 če je pritisnjen
    call Vpisi_8    ;če je pritisnjen kliči podprogram Vpiši_8
    return          ;če ni pritisnjen se vrni nazaj
Preveri_0           ;podprogram Preveri_0
    btfsc PORTB,7   ;preverimo zatič RB7 če je pritisnjen
    call Vpisi_0    ;če je pritisnjen kliči podprogram Vpiši_0
    return          ;če ni pritisnjen se vrni nazaj
Preveri_1           ;podprogram Preveri_1
    btfsc PORTB,4   ;preverimo zatič RB4 če je pritisnjen
    call Vpisi_1    ;če je pritisnjen kliči podprogram Vpiši_1
    return          ;če ni pritisnjen se vrni nazaj
Preveri_4           ;podprogram Preveri_4
    btfsc PORTB,5   ;preverimo zatič RB5 če je pritisnjen
    call Vpisi_4    ;če je pritisnjen kliči podprogram Vpiši_4
```

```
        return          ;če ni pritisnjen se vrni nazaj
Preveri_7               ;podprogram Preveri_7
        btfsc PORTB,6   ;preverimo zatič RB6 če je pritisnjen
        call Vpisi_7    ;če je pritisnjen kliči podprogram Vpiši_7
        return          ;če ni pritisnjen se vrni nazaj
Zanka                   ;glavna zanka v kateri preverjamo vsako tipko posebej
        bsf PORTB,2     ;postavimo zatič RB2 na aktivno stanje
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_3  ;kličemo podprogram Preveri_3
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_6  ;kličemo podprogram Preveri_6
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_9  ;kličemo podprogram Preveri_9
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        bcf PORTB,2     ;postavimo zatič RB2 na neaktivno stanje
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        bsf PORTB,1     ;postavimo zatič RB1 na aktivno stanje
        call Preveri_2  ;kličemo podprogram Preveri_2
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_5  ;kličemo podprogram Preveri_5
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_8  ;kličemo podprogram Preveri_8
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_0  ;kličemo podprogram Preveri_0
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        bcf PORTB,1     ;postavimo zatič RB1 na neaktivno stanje
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        bsf PORTB,0     ;postavimo zatič RB0 na aktivno stanje
        call Preveri_1  ;kličemo podprogram Preveri_1
        clrw            ;izbrišemo vsebino registra W
        clrf 0x3C       ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_4  ;kličemo podprogram Preveri_4
        clrw            ;izbrišemo vsebino registra W
```

```
        clrf 0x3C           ;izbrišemo vsebino na lokaciji 0x3C
        call Preveri_7      ;kličemo podprogram Preveri_7
        clrw                ;izbrišemo vsebino registra W
        clrf 0x3C           ;izbrišemo vsebino na lokaciji 0x3C
        bcf PORTB,0         ;postavimo zatič RB0 na neaktivno stanje
        clrw                ;izbrišemo vsebino registra W
        clrf 0x3C           ;izbrišemo vsebino na lokaciji 0x3C
        goto Zanka          ;ponovno začnemo preverjati tipke od začetka
```

Slika 13: Preverjanje tipk na matrični tipkovnici.

4.2.2.Pošiljanje števila

Prenos števila med obema PIC mikrokontrolerjema poteka med zatičema RB3. Ko prvi mikrokontroler sprejeto število pošilja preko zatiča RB3, to stori kot zaporedje aktivnih in neaktivnih signalov. Da zagotovimo drugemu mikrokontrolerju zadosti časa, da zazna aktivno povezavo in da ne prezre neaktivne, potrebujemo zadostno zakasnitev. Potrebna koda za pošiljanje števila ter za zakasnitev je vidna na sliki 14.

```
; Čakalne zanke
Pocakaj_1           ;podprogram Pocakaj_1 ki služi zakasnitvi
        call Polni_2        ;kličemo podprogram Polni_2
        decfsz 0x1C,1       ;pomanjšaj vrednost na lokaciji 0x1C
        goto Pocakaj_2      ;kličemo podprogram Pocakaj_2
        return              ;če je vrednost na lokaciji 0x1C enaka 0 se vrni nazaj
Pocakaj_2           ;podprogram Pocakaj_2 ki služi zakasnitvi
        call Polni_3        ;kličemo podprogram Polni_3
        decfsz 0x2C,1       ;pomanjšaj vrednost na lokaciji 0x2C
        goto Pocakaj_3      ;kličemo podprogram Pocakaj_3
        goto Pocakaj_1      ;kličemo podprogram Pocakaj_1
Pocakaj_3           ;podprogram Pocakaj_3 ki služi zakasnitvi
        decfsz 0x4C,1       ;pomanjšaj vrednost na lokaciji 0x4C
        goto Pocakaj_3      ;kličemo podprogram Pocakaj_3
        goto Pocakaj_2      ;kličemo podprogram Pocakaj_2
Polni_1             ;podprogram Polni_1
        movlw 0x10          ;vrednost 0x10 vnesi v register W
        movwf 0x1C          ;vrednost iz registra W prenesi na lokacijo 0x1C
        return              ;vrni se nazaj
Polni_2             ;podprogram Polni_2
        movlw 0x10          ;vrednost 0x10 vnesi v register W
        movwf 0x2C          ;vrednost iz registra W prenesi na lokacijo 0x2C
        return              ;vrni se nazaj
Polni_3             ;podprogram Polni_3
```

```
        movlw 0x55          ;vrednost 0x55 vnesi v register W
        movwf 0x4C          ;vrednost iz registra W prenesi na lokacijo 0x4C
        return              ;vrni se nazaj
Utripaj                     ;pošiljanje signalov drugemu mikrokontrolerju
        bsf PORTB,3         ;postavimo zatič RB3 na aktivno stanje
        call Polni_1        ;kličemo podprogram Polni_1
        call Pocakaj_1      ;kličemo podprogram Počakaj_1
        bcf PORTB,3         ;postavimo zatič RB3 na neaktivno stanje
        call Polni_1        ;kličemo podprogram Polni_1
        call Pocakaj_1      ;kličemo podprogram Počakaj_1
        goto Pomanjsaj      ;kličemo podprogram Pomanjsaj
Pomanjsaj                   ;podprogram za manjšanje števila katerega pošiljamo
        decfsz 0x3C,1       ;zmanjša vsebino na lokaciji 0x3C in jo shrani v 0x3C
        goto Utripaj        ;kličemo podprogram Utripaj
        return              ;nadaljuj program od klica podprograma Pomanjšaj
```

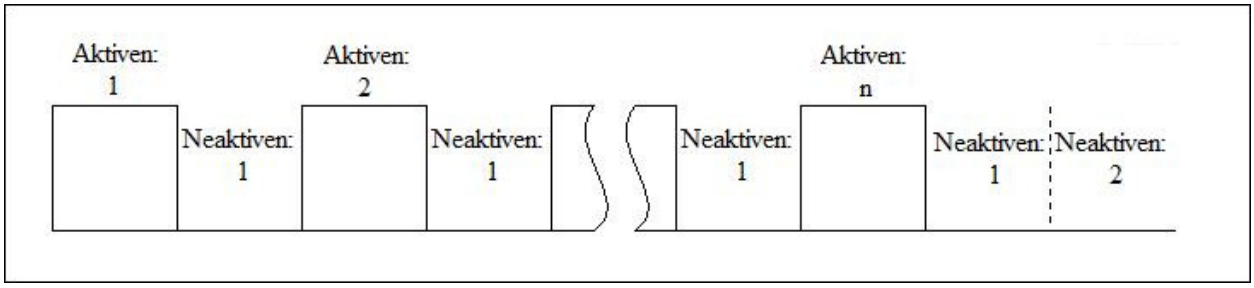
Slika 14: Pošiljanje števila drugemu mikrokontrolerju.

4.3. Naloge drugega mikrokontrolerja

Naloga drugega mikrokontrolerja je, da zazna in prebere število, ki mu ga pošilja prvi mikrokontroler. Glede na sprejeto število premakne koračni motor na želeno mesto, aktivira DC motor, ki iz omarice izvrže DVD ter koračni motor vrne na začetno pozicijo in ponovno začne poslušati vhodne signale.

4.3.1. Sprejemanje števila

Program na drugem mikrokontrolerju skrbi, da za vsako aktivno stanje zatiča RB3 poveča števec za ena. Ta števec nam na koncu pove, koliko je bilo aktivnih linij in s tem, katero število je bilo vneseno v prvem mikrokontrolerju. Da razločimo aktivna stanja med seboj, se mora zatič RB3 v enakih časovnih intervalih postaviti na neaktivno stanje. Za razločevanje premora med dvema aktivnima signaloma in koncem prenosa števila sem uporabil preprost protokol. Kot je vidno na sliki 15 sem se odločil, da dva neaktivna signala zapored pomenita konec prenosa števila. S tem



Slika 15: Zaporedje poslanih signalov.

zagotovimo pravilno štetje aktivnih signalov in nedvoumno oznako konca prenosa. Programska koda za štetje signalov in nadzora nad koncem prenosa ter vse potrebne zakasnitve so vidne na sliki 16.

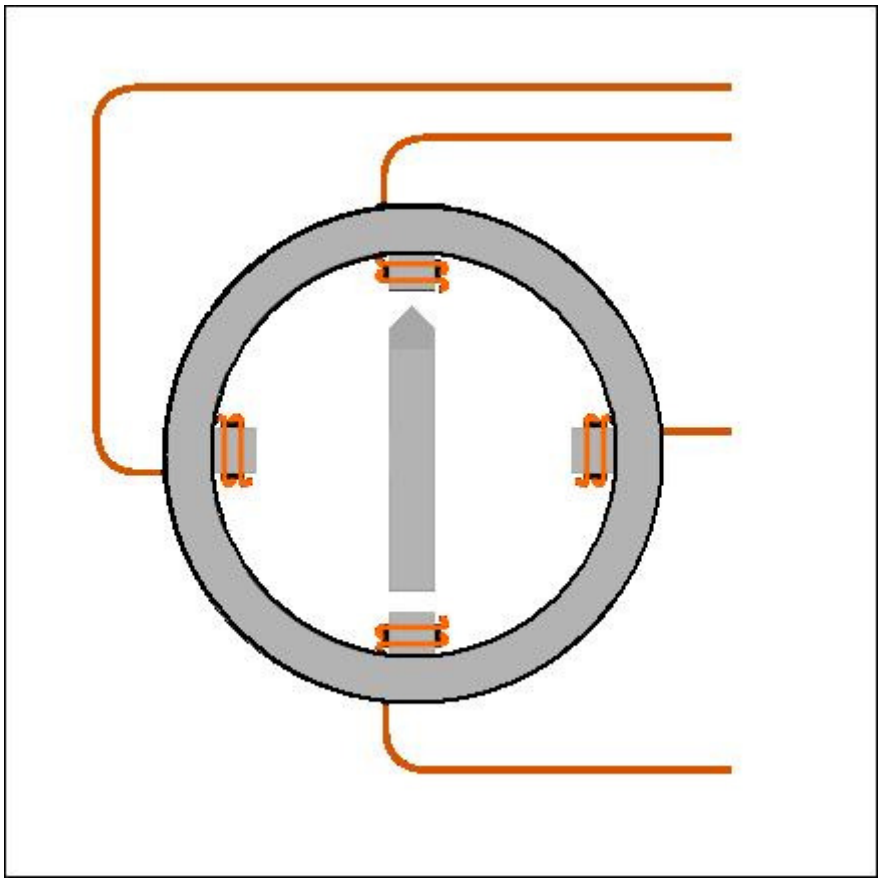
```
; Čakalne zanke
pocakaj1            ;podprogram pocakaj1 ki služi zakasnitvi
        call polni2     ;kličemo podprogram polni2
        decfsz 0x1c,1   ;pomanjšaj vrednost na lokaciji 0x1C
        goto pocakaj2   ;kličemo podprogram pocakaj2
        return          ;če je zakasnitve konec se vrnemo
pocakaj2            ;podprogram pocakaj2 ki služi zakasnitvi
        call polni3     ;kličemo podprogram polni3
        decfsz 0x22,1   ;pomanjšaj vrednost na lokaciji 0x22
        goto pocakaj3   ;kličemo podprogram pocakaj3
        goto pocakaj1   ;kličemo podprogram pocakaj1
pocakaj3            ;podprogram pocakaj3 ki služi zakasnitvi
        decfsz 0x24,1   ;pomanjšaj vrednost na lokaciji 0x24
        goto pocakaj3   ;kličemo podprogram pocakaj3
        goto pocakaj2   ;kličemo podprogram pocakaj2
polni1              ;podprogram polni1
        movlw 0x10      ;vrednost 0x10 vnesi v register W
        movwf 0x1c      ;vrednost iz registra W prenesi na lokacijo 0x1C
        return          ;vrni se nazaj
polni2              ;podprogram polni2
        movlw 0x10      ;vrednost 0x10 vnesi v register W
        movwf 0x22      ;vrednost iz registra W prenesi na lokacijo 0x22
        return          ;vrni se nazaj
polni3              ;podprogram polni3
        movlw 0x55      ;vrednost 0x55 vnesi v register W
        movwf 0x24      ;vrednost iz registra W prenesi na lokacijo 0x24
        return          ;vrni se nazaj

Linija_aktivna      ;če je zatič RB3 aktiven
        incf 0x3c,1     ;povečamo števec aktivnih stanj
        movlw 2         ;ponovno nastavimo števec neaktivnih stanj
        movwf 0x40      ;shranimo ta števec na 0x40
        call polni1     ;kličemo podprogram polni1
        call pocakaj1   ;kličemo podprogram pocakaj1
        goto Zanka      ;se vrnemo v podprogram Zanka
Linija_neaktivna    ;če je zatič RB3 neaktiven
        call polni1     ;kličemo podprogram polni1
        call pocakaj1   ;kličemo podprogram pocakaj1
        decfsz 0x40,1   ;zmanjšamo vrednost števca neaktivnih stanj
```

```
        goto Zanka          ;ko je števec različen od 0 kličemo Zanka
        goto Preveri_0      ;ko je števec 0 kličemo podprogram Preveri_0
Zanka                       ;glavna zanka za preverjanje prejetega števila
        btfsc PORTB,3       ;preverimo stanje zatiča RB3
        call Linija_aktivna ;če je RB3 logična 1 kličemo Linija_aktivna
        call Linija_neaktivna ;če je RB3 logična 0 kličemo Linija_neaktivna
        goto Zanka          ;ponovno začnemo izvajati Zanaka
```

Slika 16: Štetje signalov in nadzor nad koncem prenosa.

4.3.2.Premikanje koračnega motorja

Sedaj ko vemo, katero število je bilo poslano, moramo premakniti koračni motor na mesto, ki ustreza sprejetemu številu. Razdalja med dvema DVD-jema v omarici znaša 16 mm. Z izračunom ugotovimo, da je potrebno za to razdaljo koračni motor zavrteti 57 krat. Ker je vsak obrat koračnega motorja sestavljen iz štirih ukazov, vidimo, da je potrebno 228 signalov, da dosežemo premik med dvema DVD-jema. Kot vidimo na sliki 17 je koračni motor sestavljen iz



Slika 17: Poenostavljen prerez koračnega motorja.

štirih magnetov, katere zaporedoma polariziramo, tako da so tudi ukazi štirje. Sedaj ko vemo, koliko signalov je potrebnih za en premik, moramo ta postopek ponoviti tolikokrat kot je

vneseno število v prvem mikrokontrolerju. Če smo na primer prejeli število 6, moramo koračnemu motorju poslati 1368 signalov, da prispe na pozicijo šestega DVD-ja.

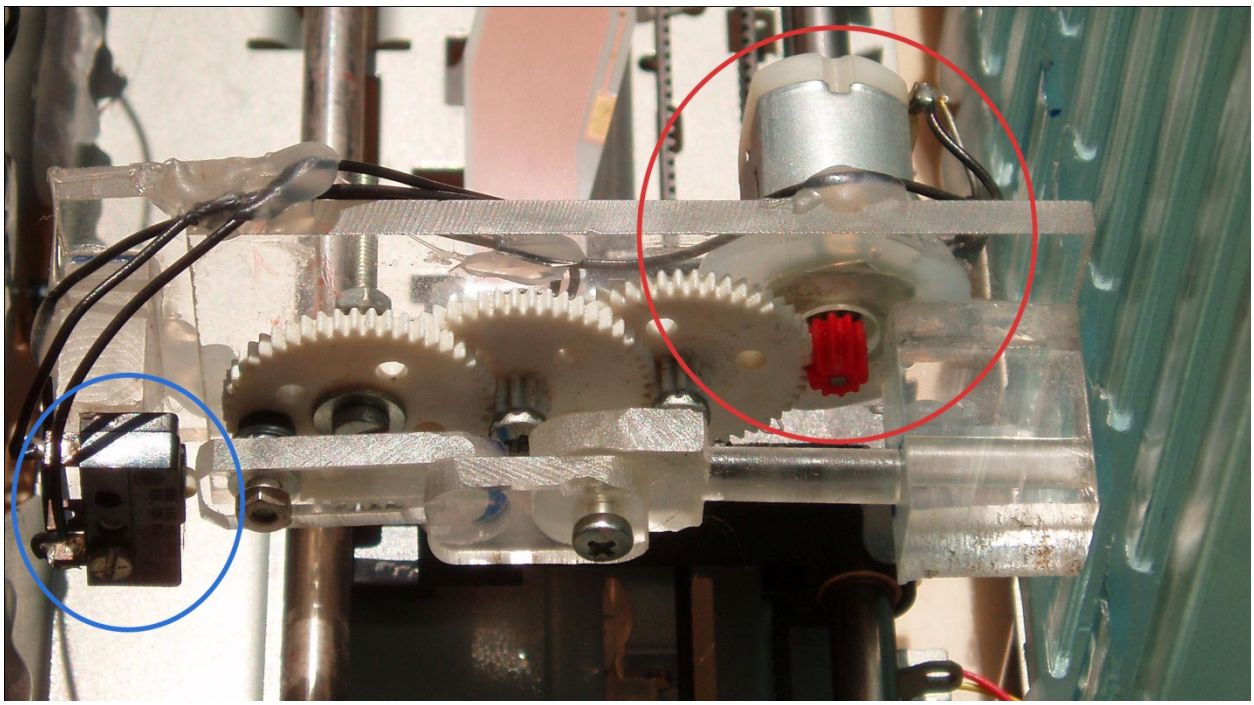
Pri polarizaciji magnetov v koračnem motorju moramo paziti na pravilni vrstni red aktiviranja magnetov, da ne pozabimo aktiviranih magnetov izklopiti ter na zakasnitve med vklopom in izklopom magnetov. Vso potrebno programsko kodo za premik koračnega motorja vidimo na sliki 18.

```
;Vrtenje koračnega motorja naprej
vrti_naprej
        bsf PORTA,0             ;postavimo zatič RA0 na aktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,0             ;postavimo zatič RA0 na neaktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bsf PORTA,1             ;postavimo zatič RA1 na aktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,1             ;postavimo zatič RA1 na neaktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bsf PORTA,2             ;postavimo zatič RA2 na aktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,2             ;postavimo zatič RA2 na neaktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bsf PORTA,3             ;postavimo zatič RA3 na aktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,3             ;postavimo zatič RA3 na neaktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        goto vrti_pom_naprej    ;kličemo podprogram vrti_pom_naprej
;Zanke za premik prvega koračnega motorja glede na ugotovljeno številko
vrti_pom_naprej                 ;podprogram za vrtenje koračnega motorja naprej
        decfsz 0x2e,1           ;pomanjšamo vrednost na lokaciji 0x2e
        goto vrti_naprej        ;kličemo podprogram vrti_naprej
        goto Preveri_1          ;kličemo podprogram Preveri_1
```

Slika 18: Premik koračnega motorja naprej.

4.3.3.Premikanje DC motorja

Ko smo se postavili na pravo mesto, je potrebno DVD potisniti iz omarice. Kot je razvidno iz slike 19, to storimo s pomočjo DC motorja in povezanim mehanizmom. DC motor je povezan s stikalom, katero vzame maso motorju. DC motor poženemo tako, da aktiviramo zatič RB5 na drugem mikrokontrolerju, ki je povezan z maso DC motorja. Ob aktivaciji se sklene električni tok, kar požene DC motor. Zaradi premikanja celotnega mehanizma, se sprosti stikalo, kar zagotovi nemoteno delovanje DC motorja, dokler celotni mehanizem ne pride v začetno stanje in stikalo ponovno pritisne, s čimer pa si odstrani maso in obstane. Dolžina signala zatiča RB5, ki ga pošiljamo na začetku za aktivacijo DC motorja, ne sme biti večja od časa, katerega potrebuje mehanizem, da naredi en obrat, saj bi v nasprotnem primeru mehanizem opravil dva obrata. Kodo za aktivacijo DC motorja in potrebne zakasnitve lahko vidimo na sliki 20.



Slika 19: DC motor (rdeč) z stikalom (moder) in kolesi za zmanjšanje hitrosti.

```
izvrzi_dvd
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bsf PORTB,4             ;postavimo zatič RB4 na aktivno stanje za aktivacijo DC motorja
        call polni1             ;kličemo podprogram polni1
        call pocakaj1           ;kličemo podprogram za zakasnitev, pocakaj1
        call polni1             ;kličemo podprogram polni1
        call pocakaj1           ;kličemo podprogram za zakasnitev, pocakaj1
        call polni1             ;kličemo podprogram polni1
        call pocakaj1           ;kličemo podprogram za zakasnitev, pocakaj1
        bcf PORTB,4             ;postavimo zatič RB4 na neaktivno stanje za izkop DC motorja
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        goto Preveri_2          ;kličemo podprogram za premik koračnega motorja nazaj
```

Slika 20: Aktiviranje DC motorja in zakasnitve.

4.3.4.Pomik na izhodišče

Ko potisnemo DVD iz omarice, se je potrebno vrniti na izhodno lokacijo, da lahko začnemo celotni postopek znova. Program za vrnitev na izhodišče je podoben programu iz poglavja 4.3.2., saj mora biti število zavrtljajev enako, vendar s to razliko, da za premikanje koračnega motorja uporabimo nasprotni vrstni red polariziranja magnetov. Potrebna koda za vrnitev na izhodišče je na sliki 21.

```
;Vrtenje koračnega motorja nazaj
vrti_nazaj
        bsf PORTA,3             ;postavimo zatič RA3 na aktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,3             ;postavimo zatič RA3 na neaktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bsf PORTA,2             ;postavimo zatič RA2 na aktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,2             ;postavimo zatič RA2 na neaktivno stanje
        call polni4             ;kličemo podprogram polni4
        call pocakaj4           ;kličemo podprogram za zakasnitev, pocakaj4
```

```
        bsf PORTA,1          ;postavimo zatič RA1 na aktivno stanje
        call polni4          ;kličemo podprogram polni4
        call pocakaj4        ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,1          ;postavimo zatič RA1 na neaktivno stanje
        call polni4          ;kličemo podprogram polni4
        call pocakaj4        ;kličemo podprogram za zakasnitev, pocakaj4
        bsf PORTA,0          ;postavimo zatič RA0 na aktivno stanje
        call polni4          ;kličemo podprogram polni4
        call pocakaj4        ;kličemo podprogram za zakasnitev, pocakaj4
        bcf PORTA,0          ;postavimo zatič RA0 na neaktivno stanje
        call polni4          ;kličemo podprogram polni4
        call pocakaj4        ;kličemo podprogram za zakasnitev, pocakaj4
        goto vrti_pom_nazaj  ;kličemo podprogram vrti_pom_nazaj
vrti_pom_nazaj               ;podprogram za vrtenje koračnega motorja nazaj
        decfsz 0x2d,1        ;pomanjšamo vrednost na lokaciji 0x2d
        goto vrti_nazaj      ;kličemo podprogram vrti_nazaj
        goto Preveri_2       ;kličemo podprogram Preveri_2
```

Slika 21: Vrtenje koračnega motorja nazaj.

5. SKLEPNE UGOTOVITVE

Diplomsko delo je imelo več ciljev. V prvem delu sem podrobno opisal osnovo mikrokontrolerjev ter njihovo delovanje. Prav tako sem podrobneje opisal delovanje mikrokontrolerja PIC16F84 in njegove pomembnejše dele. Bralcu sem želel razložiti pojem mikrokontroler in ga seznaniti s centralno enoto, katera nadzira in usmerja vse ostale komponente.

V drugem delu diplomske naloge sem opisal koncept delovanja in uporabe ključnih komponent. Opisal sem različne vrste koračnih motorjev z njihovimi prednostmi in slabostmi, pri čemer sem opisal tudi različne vrste delovanja in nadzora le-teh. Prav tako sem opisal delovanje in lastnosti DC motorja ter način nadzora teh motorjev z njihovimi prednostmi in slabostmi. Opisal sem tudi matrično tipkovnico, katero uporabimo za vnos zaporednega števila želenega DVD-ja. Razložil sem način nadzorovanja in delovanja matrične tipkovnice ter koliko in zakaj je potrebnih toliko zatičev na mikrokontrolerju za brezhibno delovanje.

Ko je bralec seznanjen z vsemi ključnimi komponentami omarice, mu v tretjem delu predstavim program MPLAB, v katerem sem napisal oba programa za mikrokontrolerja. Posamezno opišem tudi naloge obeh mikrokontrolerjev, težave s katerimi se srečamo in programske rešitve teh nalog. Dodatno še opišem in razložim kodo, s katero zgoraj naštete komponente nadzorujemo in upravljamo.

Kljub vsem težavam na katere sem naletel pri izdelavi omarice, sem s končnim izdelkom in pridobljenim praktičnim znanjem zelo zadovoljen. Omarica je povsem delujoč zaključen izdelek z možnostjo razširitve oz. nadaljnjega razvoja. Primer razširitve ali nadgradnje bi bilo povečana količina DVD-jev, katere lahko omarica vsebuje z dodajanjem nadstropij ali pa dodati LED zaslon za izpis vnesenega števila.

SEZNAM SLIK

SEZNAM TABEL

SEZNAM LITERATURE IN VIROV

[1] D. Dribin, Keyboard matrix help. Dostopno na: http://www.dribin.org/dave/keyboard/one_html/, Julij 2010

[2] M. Hennessy, PIC programming. Dostopno na: http://www.mhennessy.f9.co.uk/pic/index.htm, Julij 2010

[3] D. W. Jones, Control of Stepping motors. Dostopno na: http://www.cs.uiowa.edu/~jones/step/, Julij 2010

[4] N. Matic, The PIC Microcontroller book. Dostopno na: http://www.ime.eb.br/~pinho/micro/apostila/pic/The_PIC_Microcontroller_Book.pdf, Julij 2010

[5] E. Seale, DC motors, How they work. Dostopno na: http://www.solarbotics.net/starting/200111_dcmotor/200111_dcmotor.html, Julij 2010

[6] Microchip, MPLAB IDE Quick start guide. Dostopno na: http://ww1.microchip.com/downloads/en/DeviceDoc/51281d.pdf, Julij 2010

[7] Microchip, MPLAB Integrated Development Environment. Dostopno na: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en019469&part=SW007002, Julij 2010

[8] Microchip, PIC16F84 Data Sheet. Dostopno na: http://ww1.microchip.com/downloads/en/DeviceDoc/35007b.pdf, Julij 2010

[9] Parallax, What is a microcontroller. Dostopno na: http://www.parallax.com/dl/docs/books/edu/wamv2_2.pdf, Julij 2010

[10] SiliconBlue, Matrix keypad scanner. Dostopno na: http://www.siliconbluetech.com/media/intellectual-property/Matrix_Keypad_Scanner_DE102.pdf, Julij 2010

[11] Brushed DC electric motor. Dostopno na: http://en.wikipedia.org/wiki/Brushed_DC_electric_motor, Julij 2010

[12] DC motor. Dostopno na: http://en.wikipedia.org/wiki/DC_motor, Julij 2010

[13] MPLAB. Dostopno na: http://en.wikipedia.org/wiki/MPLAB, Julij 2010

[14] Stepper motor. Dostopno na: http://en.wikipedia.org/wiki/Stepper_motor, Julij 2010

[15] Stepper motor basics. Dostopno na: http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf, Julij 2010

[16] Talking electronics. Dostopno na: http://talking-electronics.netfirms.com/index.html, Julij 2010

PRILOGA

Program prvega mikrokontrolerja

```
        #include <p16f84.inc>
        cblock 0xc
        endc
        org 0
        goto Glavni
        org 5
Glavni
        clrf STATUS
        clrf PORTB
        bsf STATUS,RP0
        movlw 0xf0
        movwf TRISB
        bcf STATUS,RP0
        clrw
        bsf PORTB,2
Preveri_Enter
        btfsc PORTB,7
        goto Preveri_pom
        goto Preveri_Enter
; Čakalne zanke
Pocakaj_1
        call Polni_2
        decfsz 0x1C,1
        goto Pocakaj_2
        return
Pocakaj_2
        call Polni_3
        decfsz 0x2C,1
        goto Pocakaj_3
        goto Pocakaj_1
Pocakaj_3
        decfsz 0x4C,1
        goto Pocakaj_3
        goto Pocakaj_2
Polni_1
        movlw 0x10
        movwf 0x1C
```

```
        return
Polni_2
        movlw 0x10
        movwf 0x2C
        return
Polni_3
        movlw 0x55
        movwf 0x4C
        return
Utripaj
        bsf PORTB,3
        call Polni_1
        call Pocakaj_1
        bcf PORTB,3
        call Polni_1
        call Pocakaj_1
        goto Pomanjsaj
Pomanjsaj
        decfsz 0x3C,1
        goto Utripaj
        return
; Vpisovanje pritisnjene tipke
Vpisi_3
        movlw 4
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_6
        movlw 7
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_9
        movlw 0xa
        movwf 0x3c
        call Pomanjsaj
        return
Vpisi_2
        movlw 3
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_5
```

```
        movlw 6
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_8
        movlw 9
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_0
        ;movlw 1
        ;movwf 0x3C
        ;call Pomanjsaj
        return
Vpisi_1
        movlw 2
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_4
        movlw 5
        movwf 0x3C
        call Pomanjsaj
        return
Vpisi_7
        movlw 8
        movwf 0x3C
        call Pomanjsaj
        return
; Preverjanje pritisnjene tipke
Preveri_3
        btfsc PORTB,4
        call Vpisi_3
        return
Preveri_6
        btfsc PORTB,5
        call Vpisi_6
        return
Preveri_9
        btfsc PORTB,6
        call Vpisi_9
        return
Preveri_2
```

```
        btfsc PORTB,4
        call Vpisi_2
        return
Preveri_5
        btfsc PORTB,5
        call Vpisi_5
        return
Preveri_8
        btfsc PORTB,6
        call Vpisi_8
        return
Preveri_0
        btfsc PORTB,7
        call Vpisi_0
        return
Preveri_1
        btfsc PORTB,4
        call Vpisi_1
        return
Preveri_4
        btfsc PORTB,5
        call Vpisi_4
        return
Preveri_7
        btfsc PORTB,6
        call Vpisi_7
        return
; Glavna zanka
Preveri_pom
        bcf PORTB,2
        goto Zanka
Zanka
        bsf PORTB,2
        clrw
        clrf 0x3C
        call Preveri_3
        clrw
        clrf 0x3C
        call Preveri_6
        clrw
        clrf 0x3C
        call Preveri_9
        clrw
```

```
clrf 0x3C
bcf PORTB,2
clrw
clrf 0x3C
bsf PORTB,1
call Preveri_2
clrw
clrf 0x3C
call Preveri_5
clrw
clrf 0x3C
call Preveri_8
clrw
clrf 0x3C
call Preveri_0
clrw
clrf 0x3C
bcf PORTB,1
clrw
clrf 0x3C
bsf PORTB,0
call Preveri_1
clrw
clrf 0x3C
call Preveri_4
clrw
clrf 0x3C
call Preveri_7
clrw
clrf 0x3C
bcf PORTB,0
clrw
clrf 0x3C
goto Zanka
end
```

Program drugega mikrokontrolerja

```
#include <p16f84.inc>
cblock 0xc
endc
org 0
goto Glavni
```

```
        org 5
Glavni
        clrf STATUS
        clrf PORTB
        bsf STATUS,RP0
        movlw 0x0f
        movwf TRISB
        movlw 0x00
        movwf TRISA
        bcf STATUS,RP0
        clrw
        movlw 0x2
        movwf 0x40
;Čakalne vrednosti za prenos
        movlw 0x10
        movwf 0x1c
        movlw 0x10
        movwf 0x22
        movlw 0x55
        movwf 0x24
;Čakalne vrednosti za motor
        movlw 0x5
        movwf 0x26
        movlw 0x10
        movwf 0x28
        movlw 0x30
        movwf 0x2a
        movlw 0x01
        movwf 0x2d
        movlw 0x01
        movwf 0x2e
        movlw 0x39
        movwf 0x2f
        movwf 0x30
Preveri_zacetek
        movlw 1
        movwf 0x3c
        btfsc PORTB,3
        goto Zanka
        goto Preveri_zacetek
; Čakalne zanke
pocakaj1
        call polni2
```

```
            decfsz 0x1c,1
            goto pocakaj2
            return
pocakaj2
            call polni3
            decfsz 0x22,1
            goto pocakaj3
            goto pocakaj1
pocakaj3
            decfsz 0x24,1
            goto pocakaj3
            goto pocakaj2
polni1
            movlw 0x10
            movwf 0x1c
            return
polni2
            movlw 0x10
            movwf 0x22
            return
polni3
            movlw 0x55
            movwf 0x24
            return
pocakaj4
            call polni5
            decfsz 0x26,1
            goto pocakaj5
            return
pocakaj5
            call polni6
            decfsz 0x28,1
            goto pocakaj6
            goto pocakaj4
pocakaj6
            decfsz 0x2a,1
            goto pocakaj6
            goto pocakaj5
polni4
            movlw 0x5
            movwf 0x16
            return
polni5
```

```
            movlw 0x10
            movwf 0x28
            return
polni6
            movlw 0x30
            movwf 0x2a
            return
;Vrtenje koračnega motorja naprej
vrti_naprej
            bsf PORTA,0
            call polni4
            call pocakaj4
            bcf PORTA,0
            call polni4
            call pocakaj4
            bsf PORTA,1
            call polni4
            call pocakaj4
            bcf PORTA,1
            call polni4
            call pocakaj4
            bsf PORTA,2
            call polni4
            call pocakaj4
            bcf PORTA,2
            call polni4
            call pocakaj4
            bsf PORTA,3
            call polni4
            call pocakaj4
            bcf PORTA,3
            call polni4
            call pocakaj4
            goto vrti_pom_naprej
;Vrtenje koračnega motorja nazaj
vrti_nazaj
            bsf PORTA,3
            call polni4
            call pocakaj4
            bcf PORTA,3
            call polni4
            call pocakaj4
            bsf PORTA,2
```

```
                call polni4
                call pocakaj4
                bcf PORTA,2
                call polni4
                call pocakaj4
                bsf PORTA,1
                call polni4
                call pocakaj4
                bcf PORTA,1
                call polni4
                call pocakaj4
                bsf PORTA,0
                call polni4
                call pocakaj4
                bcf PORTA,0
                call polni4
                call pocakaj4
                goto vrti_pom_nazaj
;Zanke za premik koračnega motorja
vrti_pom_naprej
                decfsz 0x2e,1
                goto vrti_naprej
                goto Preveri_1
izvrzi_dvd
                call polni4
                call pocakaj4
                call polni4
                call pocakaj4
                bsf PORTB,4
                call polni1
                call pocakaj1
                call polni1
                call pocakaj1
                call polni1
                call pocakaj1
                bcf PORTB,4
                call polni4
                call pocakaj4
                call polni4
                call pocakaj4
                goto Preveri_2
vrti_pom_nazaj
                decfsz 0x2d,1
```

```
                goto vrti_nazaj
                goto Preveri_2
Preveri_0
                movf 0x3c,0
                movwf 0x3e
                movwf 0x3d
                incf 0x3e,1
                incf 0x3d,1
                goto Preveri_1
Preveri_1
                movlw 57
                movwf 0x2e
                decfsz 0x3e,1
                call vrti_naprej
                goto izvrzi_dvd
Preveri_2
                movlw 57
                movwf 0x2d
                decfsz 0x3d,1
                goto vrti_nazaj
                goto Preveri_zacetek
;Preverjanje številke, ki se trenutno pošilja
Linija_aktivna
                incf 0x3c,1
                movlw 2
                movwf 0x40
                call polni1
                call pocakaj1
                goto Zanka
Linija_neaktivna
                call polni1
                call pocakaj1
                decfsz 0x40,1
                goto Zanka
                goto Preveri_0
Zanka
                btfsc PORTB,3
                call Linija_aktivna
                call Linija_neaktivna
                goto Zanka
                end
```