Univerza v Ljubljani
Fakulteta za elektrotehniko

Nejc Gombač

Sistem za avtomatično upravljanje ročnega potenciometra

Diplomsko delo
Mentor: Prof. dr. Dejan Križaj

Ljubljana, 2014

Zahvala

Zahvaljujem se vsem, ki so kakorkoli pripomogli k diplomskemu delu. Še posebej pa bi se rad zahvalil:

- mojemu mentorju doc. dr. Dejan Križaju, univ. dipl. inž. el., ki mi je predlagal idejo za izdelavo tega dela in me vseskozi podpiral ter usmerjal,
- staršem, ki so me moralno in finančno podpirali celotno študijsko obdobje in posebej očetu, ki mi je pomagal izdelati nosilec za koračni motor,
- Drago Tacarju, Zumert Topčagiću in Robi Brajkoviču, ki so mi pomagali pri delu v laboratoriju.

Kazalo

Kazalo slik

Povzetek

V današnjem času je prioriteta meritve opravljati z računalnikom oz. avtomatskim merilnim sistemom, saj lahko izmerjene podatke shranimo, primerjamo z drugimi podatki in med njimi opravljamo različne primerjalne, logične ter računske operacije. To nam omogoča bistveno hitrejše reševanje problemov. Vendar pa smo včasih vseeno prisiljeni uporabljati starejšo opremo, ki ni avtomatizirana in je ne moremo upravljati z računalnikom.

V nalogi sem opisal enega izmed možnih načinov avtomatizacije ročnega potenciometra. Na konkretnem primeru uporabe takšnega sistema – merjenje histerezne krivulje – sem preizkusil, kakšna je funkcionalnost sistema, kakršnega sem si zamislil.

Avtomatizacija sistema je smotrna, če je sistem hitrejši in natančnejši od ročne meritve. Pomemben dejavnik v sodobnem času je tudi cena dodatne opreme za avtomatizacijo. Zato sem poskusil poiskati najenostavnejšo rešitev z dokaj preprosto opremo, tako strojno kot tudi programsko.

Problem sem poskusil rešiti s koračnim motorjem, pritrjenim na nosilec, ki ga postavimo ob napravo, ki jo želimo kontrolirati. Gred motorja sem povezal z gredjo potenciometra. Motorček sem krmilil s kartico Arduino Uno, ki sem jo sprogramiral z nekoliko poenostavljenim programskim jezikom C++ v programskem okolju Arduino.

Ključne besede: avtomatski merilni sistem, ročni potenciometer, histerezna krivulja, hitrost, natančnost, cena, koračni motor, Arduino Uno, C++

Abstract

Nowadays the priority is to take measurements with a computer or automatic measuring system so you can store the measuring data, compare them with other data or perform various comparative, logical and arithmetic operations. This allows us to speed up problem solving significantly. However, sometimes we are forced to use older equipment that is not automated and it does not allow to communicate with computer to store data.

In the task I have described one of the possible ways of automation manual potentiometer. In the particular case – measuring hysteresis curve I have tested what is the functionality of the system I imagined.

Automation system is efficient if the system is faster and more precise than the manual measurement. An important factor in modern times is the price of additional automation equipment. So I tried to find the simplest solution with relatively simple equipment both hardware as well as software.

I tried to solve the problem with a stepper motor attached to a carrier disposed adjacent to the device which we want to control. I connected the motor shaft with the shaft of the potentiometer. The motor is controlled by the Arduino Uno card which is programmed with a simplified programming language C++ in the Arduino programming environment.

Key words: automatic measuring system, manual potentiometer, hysteresis curve, speed, precision, accuracy, price, stepper motor, Arduino Uno, C++

1 Uvod

V diplomski nalogi sem opisal sistem za avtomatizacijo ročnega potenciometra. Sistem sem moral realizirati tako, da sem uporabil primeren koračni motor, ki bi vrtel gumb potenciometra. Vrtenje motorja sem kontroliral z mikrokrmilnikom. Hkrati sem moral omogočiti zajemanje in shranjevanje signalov. Sistem sem realiziral na primeru merjenja histerezne krivulje feromagnetnega jedra. Glavni namen naloge je bil poiskati cenovno ugodno in za izvedbo enostavno rešitev. Poleg tega sem preveril praktičnost tako izvedenega sistema.

Za izvedbo sistema sem raziskal dve možnosti. Izbiral sem med možnostmi krmiljenja koračnega motorja in zajemanjem analognih signalov z večfunkcijskim modulom DAQ, proizvajalca National Instruments – NI 6211, ter odprtokodno platformo Arduino Uno.

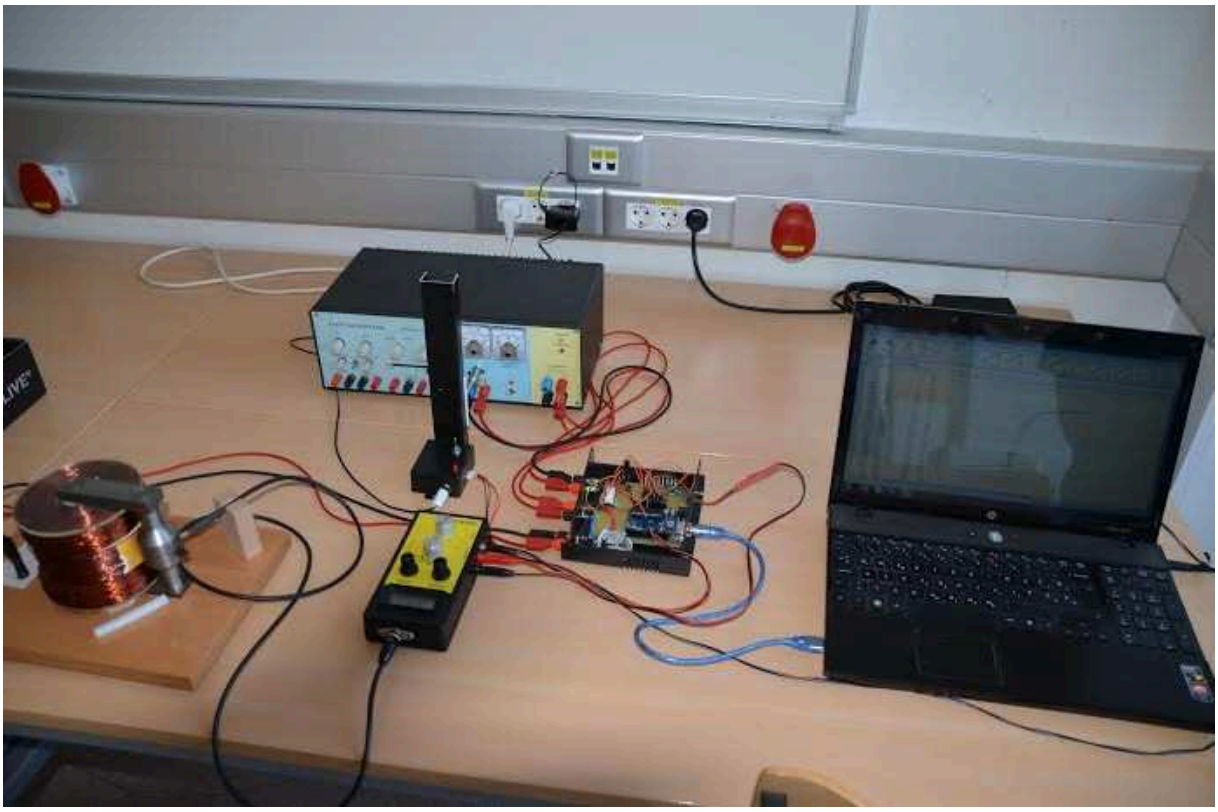
Večfunkcijski modul NI 6211 je kompatibilen s programskima okoljema LabView in MatLab. Ima 16 analognih vhodov, 2 analogna izhoda, 4 digitalne vhode in 4 digitalne izhode. Omogoča merjenje napetosti med -10 V in +10 V preko 16 bitnega A/D pretvornika.

Odprtokodna platforma Arduino Uno ima svoje programsko okolje, ki omogoča programiranje v nekoliko poenostavljenem jeziku C++. Ima 14 digitalnih vhodov ali izhodov in 6 analognih vhodov, ki omogočajo zajemanje napetosti med 0 V in 5 V preko 10 bitnega A/D pretvornika.

Pri izvedbi sistema za merjenje histerezne krivulje sem potreboval 5 digitalnih izhodov. Od tega 4 za krmiljenje koračnega motorja in enega za krmiljenje releja. Potreboval sem še 2 analogna vhoda, na katerih sem meril Hallovo napetost iz teslametra in napetost na merilnem uporu.

Sistem, prikazan na Sliki 1, sem izvedel iz treh glavnih komponent:

- nosilca koračnega motorja s koračnim motorjem in gonilnikom,
- mikrokrmilnika z merilnim vezjem in relejem,
- računalnika za shranjevanje in obdelavo podatkov.



Slika 1: Sistem za avtomatično upravljanje ročnega potenciometra

Obe kartici sta omogočali realizacijo sistema, vendar sem se odločil za odprtokodno platformo Arduino Uno. Sistem sem izvedel s cenejšo opremo, saj me je zanimalo, ali bi bile meritve dovolj natančne. Merilno območje in AD pretvornik sta relativno majhna v primerjavi z veliko dražjim modulom NI 6211. Poleg tega platforma ne omogoča merjenja negativnih vrednosti napetosti, zato sem moral narediti dodatno vezje, s katerim sem to omogočil. Vezje mora dvigniti referenčno napetost iz O V na 2,5 V. Tej vrednosti se prišteva drugi napetostni signal, ki se spreminja med vrednostmi -2 V do 2 V. Pomemben dejavnik pri izbiri kartice je bila tudi programska oprema. S programsko opremo LabWiev nisem imel izkušenj, zato sem se odločil za bolj poznano programiranje v programskem jeziku C++.

2 Koračni motor in gonilnik

Primeren motor za izvedbo avtomatizacije potenciometra mora/ne:

- omogočati enostavno krmiljenje,
- omogočati natančne premike,
- biti nizkonapetostno napajan,
- biti dovolj hiter,
- biti majhen,
- biti lahek,
- potrebuje velikega navora.

Vsem tem zahtevam zadostujejo koračni motorji, zato sem za svoj sistem izbral motor 28BYJ-48 5V (prikazan na Sliki 3). Poleg zgoraj naštetih lastnostih je lahko dostopen na tržišču in cenovno zelo ugoden.

2.1 O koračnih motorjih

Koračni motorji so brezkrtačni enosmerni motorji, ki pretvarjajo električne impulze v premo sorazmerne mehanske premike. Rotor se tako ustrezno premika oziroma zavrti v diskretnih korakih. Njihova največja prednost pred ostalimi motorji je, da zagotavljajo nadzorovano gibanje in pozicioniranje brez komulativne napake. Poleg tega je prednost v enostavnosti uporabe krmilnih vezij. Ta so v večini v integrirani obliki.

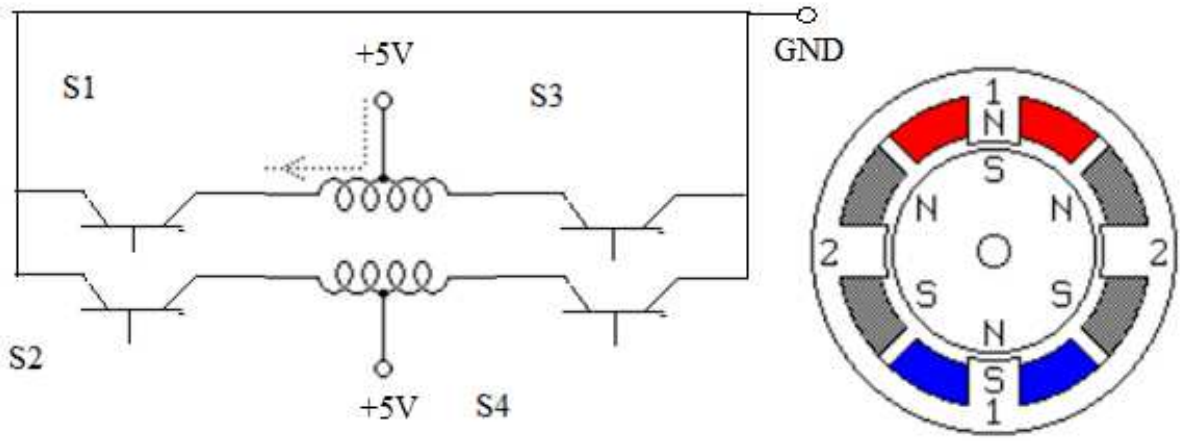
Sistem s koračnim motorjem sestavljajo koračni motor, napajalnik, krmilna logika in močnostni ojačevalnik. Za pravilno delovanje moramo motor pravilno priključiti in ustrezno krmiliti navitja. Enosmerno napetost priključimo na motor preko mreže močnostnih elektronskih stikal, ki jih odpiramo z izvorom krmilnih impulzov (digitalna informacija). To je lahko generator z nastavljivo frekvenco, namensko integrirano vezje ali pa mikroprocesorski sistem. Vsak impulz premakne rotor za en korak. Kot enega koraka je odvisen od tipa motorja in je lahko od 1,8° do 15°. Hitrost zasuka je določena s frekvenco impulzov. Poznamo več metod vzbujanja. Običajna metoda vzbujanja je 4-koračna sekvenca, ostale se uporablja odvisno od izvedbe navitja in zahtevanega

vzorca logičnega zaporedja impulzov, ki krmilijo stikala. Za motor, ki sem ga izbral, je priporočljivo vzbujanje s polovičnim korakom. Na ta način je mogoče vrteti motor v zaporedju polovičnih korakov, kar prispeva k bolj gladkemu vrtenju. Slabost je, da se držalni moment spreminja med močnejšim in slabšim. V nekaterih primerih lahko zaradi tega nastanejo določene težave. Pri izvedbi opisanega sistema teh težav ni bilo, saj je bremenski navor manjši od navora, ki ga proizvede ena sama vzbujena tuljava.

2.2 Unipolarni koračni motor 28BYJ-48 5 V

Delovanje

Unipolarni koračni motor ima na vsakem izmed dveh statorskih navitjih izveden sredinski odcep, ki je spojen na pozitiven priključek napajalnega izvora. Smer toka je odvisna od tega, kateri konec navitja je spojen preko močnostnega stikala (tranzistorja) na negativen potencial napajalnika. V primeru da je vklopljeno stikalo S1 (prikazano na Sliki 2), bo tok stekel skozi levo polovico prvega navitja, kar bo povzročilo magnetni pretok v smeri, kot je prikazano na sliki. Pretok povzroči, da se rotor, ki je trajni magnet, poravna v smeri, prikazani na sliki. Ko izklopimo stikalo S1 in vklopimo S2, bo tok stekel skozi drugo polovico, kar povzroči magnetenje v nasprotni smeri in posledično drugačno poravnavo rotorja. Ob pravilnem preklapljanju stikal se rotor zavrti v eno ali drugo smer. [1]



Slika 2: Shema vezave unipolarnega koračnega motorja s polprevodniškimi stikali

Specifikacije motorja

- Model：28BYJ-48
- Nazivna napetost：5 V DC
- Število faz: 4
- Prestavno mehansko razmerje：1/64
- Kot zasuka：5.625°/64
- DC upornost：50 Ω ± 7 % (25 ℃)
- Delovni moment >34.3 mN.m (120 Hz)
- Držalni moment >34.3 mN.m
- Izolacijska upornost >10M Ω (500 V)
- Teža : 30 g

Slika 3: Unipolarni koračni motor 28BYJ-48

Napajalno napetost motorja je priporočljivo nekoliko zvišati zaradi padca napetosti na gonilniku. Nekoliko se poveča tudi hitrost vrtenja motorja. Največja povišana napetost je 12 V. Zaradi mehanskega prestavnega razmerja se zmanjša kot zasuka gredi. V enem koraku se motor zasuka za 5,625°, gred motorja pa za 5,625°/64 oziroma približno 0,088°. Zaradi tega potrebujemo za celoten obrat motorja približno 4076 korakov. [2]

2.3 Gonilnik

Da lahko motor obratuje, potrebujemo gonilnik oz. tranzistorsko vezje. Uporabil sem gonilnik ULN 2003, ki je prikazan na Sliki 4. Ta omogoča, da lahko enostavno kontroliramo koračni motor iz mikrokrmilnika, kot je Arduino Uno.

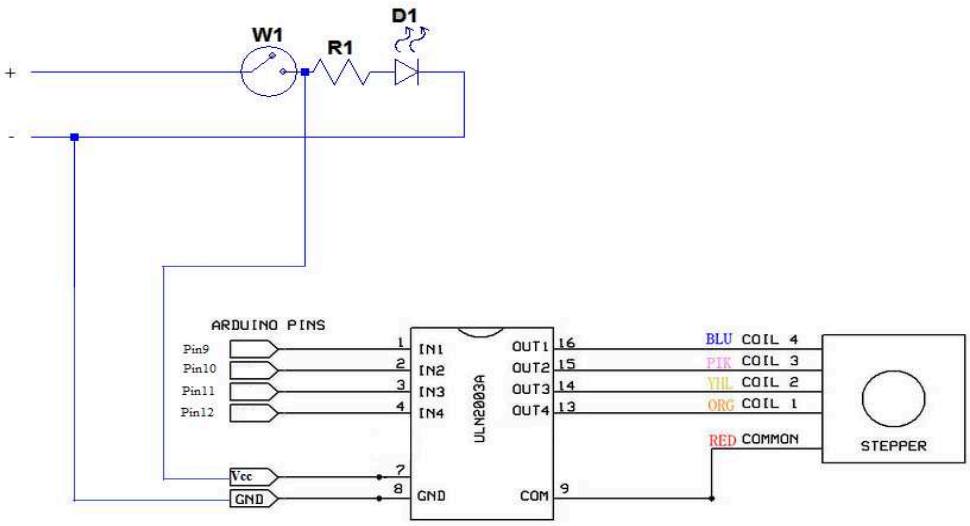
Integrirano vezje ULN 2003 je tranzistorsko polje, sestavljeno iz sedmih parov bipolarnih tranzistorjev, vezanih v Darlington vezavi z odprtimi kolektorskimi priključki in skupnimi emitorji. K vsakemu Darlingtonovemu paru je vezana povratna dioda, kar omogoča preklapljanje induktivnih bremen. Maksimalni izhodni tok je 500 mA na kanal. Ko je vezje vključeno, ima zaradi notranje upornosti padec napetosti 1 V. Poleg čipa so na gonilnem vezju še:

- 5-žični konektor, na katerega priključimo motor,
- 4 led diode, ki signalizirajo, katera tuljava je trenutno napajana,
- 6 vhodnih pinov, od tega 4, ki povezujejo vhode na čip in izhode iz mikrokrmilnika, Vcc za pozitivni priključek napajalne napetosti in GND za negativni priključek napajalne napetosti. [1]

Slika 4: Gonilnik ULN 2003

2.4 Nosilec za koračni motor

Nosilec (na Sliki 6) za koračni motor je sestavljen iz dveh delov. Prvi del je narejen iz ohišja za gonilnik motorja. Hkrati ohišje predstavlja oporo s podlago, zato je postavljeno na kovinsko ploščico, ki poskrbi za obtežitev. Tako se celotni nosilec ne premika, ko motor obratuje. Vezje (prikazano na Sliki 5) je sestavljeno iz napajalnega konektorja enosmerne napetosti, stikala, ki vklopi napajanje, LED diode, ki signalizira, kdaj je napajanje vključeno, ULN2003 gonilnika ter 4-pinskega konektorja, na katerega priključimo digitalne izhode za krmiljenje motorja.



Slika 5: Shema napajalnega vezja z gonilnikom v nosilcu motorja

Drugi del je sestavljen iz profila, pritrjenega na ohišje vezja. Motor je pritrjen na ploščico, ki se pomika po profilu, kar omogoča prilagoditev višine gredi motorja na višino gredi potenciometra, ki ga želimo upravljati. Motor in potenciometer sta povezana s skupno osjo, ki se jo pritrdi na obe gredi s stranskimi vijaki.

Slika 6: Nosilec koračnega motorja

3 Odprtokodna platforma Arduino Uno

3.1 Strojna oprema

Arduino Uno (prikazan na Sliki 7) je odprtokodna platforma, na katero lahko priklopimo različne senzorje in aktuatorje. Platforma ima 14 digitalnih pinov, ki se jih lahko uporablja kot vhode in izhode. Obratujejo na napetosti 5 V in lahko sprejmejo ali oddajo maksimalno 40 mA toka vsak posebej. Nekateri pini lahko izvajajo posebne funkcije. Pin 0 lahko sprejema TTL serijske podatke, pin 1 pa jih oddaja. Oba sta povezana na ATmega16U2 USB-to-TTL Serial čip. 6 pinov (3, 5, 6, 9, 10, 11) se lahko uporablja kot 8-bitne PWM (pulznoširinska modulacija) izhode. Pini 10–13 omogočajo SPI komunikacijo, kar omogoča povezovanje več modulov skupaj. Pri tem je eden od modulov glavni (»single master comunication«). Platforma ima 6 analognih vhodov, ki so povezani na 10-bitni A/D pretvornik. Merjenje poteka med 0 V in 5 V. Če uporabimo AREF pin, lahko zgornjo vrednost spremenimo tako, da na ta pin dovedemo želeno napetost, ki pa mora biti manjša od 5 V. Arduino ima 16 MHz keramični resonator, napajalni priključek, USB konektor, ICSP header ter gumb za resetiranje. Od ostalih platform se Arduino Uno razlikuje po tem, da ima namesto FTDI USB – to – serial čipa čip ATmega 16U2, ki je sprogramiran kot pretvornik iz USB – to – serial. Poleg tega ima tudi avtomatsko varovalko, ki ščiti USB port na računalniku pred kratkimi stiki in prevelikimi tokovi. Čeprav ima večina računalnikov že svojo pretokovno zaščito, predstavlja varovalka dodatno zaščito. V primeru da tok preseže 500 mA, varovalka prekine povezavo, dokler napaka ni odpravljena. Platformo Arduino Uno lahko napajamo preko USB konektorja ali iz zunanjega enosmernega vira (adapter ali baterijsko napajanje). Adapter lahko priključimo preko napajalnega JACK konektorja, medtem ko baterije priklopimo na pina Vin ter GND, ki sta povezana z napajalnim konektorjem. Priporočena napajalna napetost je med 7 V in 12 V. Na plošči sta dva napajalna pina, 5 V in 3,3 V, s katerima lahko napajamo zunanje aktuatorje. Maksimalen izhodni tok je 50 mA. Napajanje plošče preko teh pinov le-to lahko poškoduje in uniči, saj nista povezana z napetostnim regulatorjem. Mikrokrmilnik na plošči programiramo v programskem okolju Arduino z istoimenskim programskim jezikom, ki bazira na jeziku C++.

Platforma Arduino Uno ima nameščen bootloader (računalniški program, ki naloži operacijski sistem po opravljenem samopreverjanju (self-test)), ki omogoča nalaganje programske kode na mikrokrmilnik brez uporabe dodatne hardwarske opreme. Z uporabo ICSP-ja se lahko premosti bootloader na platformi in lahko uporablja zunanjo opremo (npr. nalaganje in preverjanje izvede osebni računalnik). Arduino Uno omogoča avtomatsko resetiranje mikrokrmilnika s programsko opremo, ki se nalaga iz osebnega računalnika, s katerim je platforma povezana. S tem se izognemo fizičnemu pritisku na tipko reset, ki bi jo morali pritisniti vsakič, ko bi želeli naložiti programsko kodo. Mikrokrmilnik se resetira tudi vsakič, ko je opravljena povezava preko USB priključka na osebni računalnik. [4]

Specifikacije

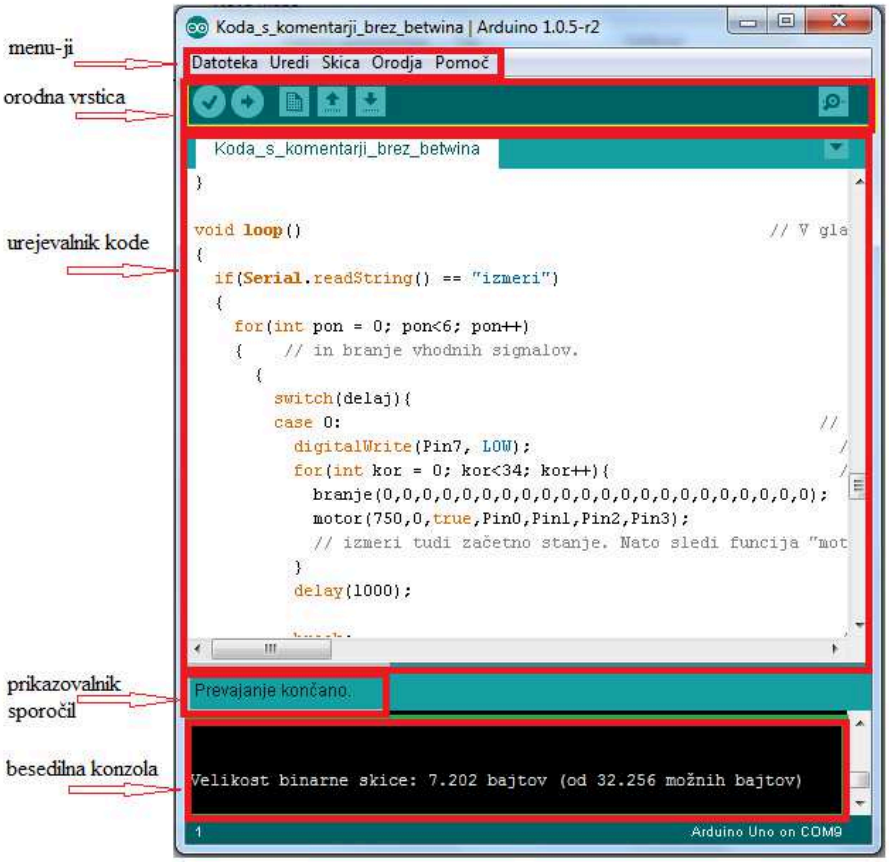
- Mikrokrmilnik ATmega328
- Delovna napetost 5 V
- Napajalna napetost (priporočeno) 7–12 V
- Digitalni I/O pini 14 (6 lahko kot PWM izhodi)
- Število analognih vhodnih pinov 6
- DC tok na I/O pin 40 mA
- DC tok na 3.3 V in 5 V pinu 50 mA
- Flash Memory 32 KB (ATmega328)
- SRAM 2 KB (ATmega328)
- EEPROM 1 KB (ATmega328)
- Hitrost števca 16 MHz

Slika 7: Odprtokodna platforma Arduino Uno

3.2 Arduino programsko okolje

Programsko okolje (prikazano na Sliki 8) omogoča nalaganje kode na strojno opremo ter komunikacijo. Programska koda, ki jo pišemo, se imenuje sketch oziroma »skica«. Programski jezik, ki se uporablja, je nekoliko poenostavljena verzija jezika C++. Nastal je iz jezika Processing, ki je odprtokodni programski jezik, osnovan v jeziku Java. Namenjen je poučevanju osnovnega računalniškega programiranja.

Okolje vsebuje urejevalnik kode, kjer pišemo programsko kodo, orodno vrstico z gumbi za skupne funkcije, vrsto menijev, prikazovalnik sporočil, kjer se zapišejo povratna sporočila o shranjevanju in nalaganju kode, ter besedilno konzolo, v kateri se zapisujejo izhodni podatki programske opreme Arduino. Tu se zapišejo napake, prisotne v kodi, in še nekatera druga sporočila.

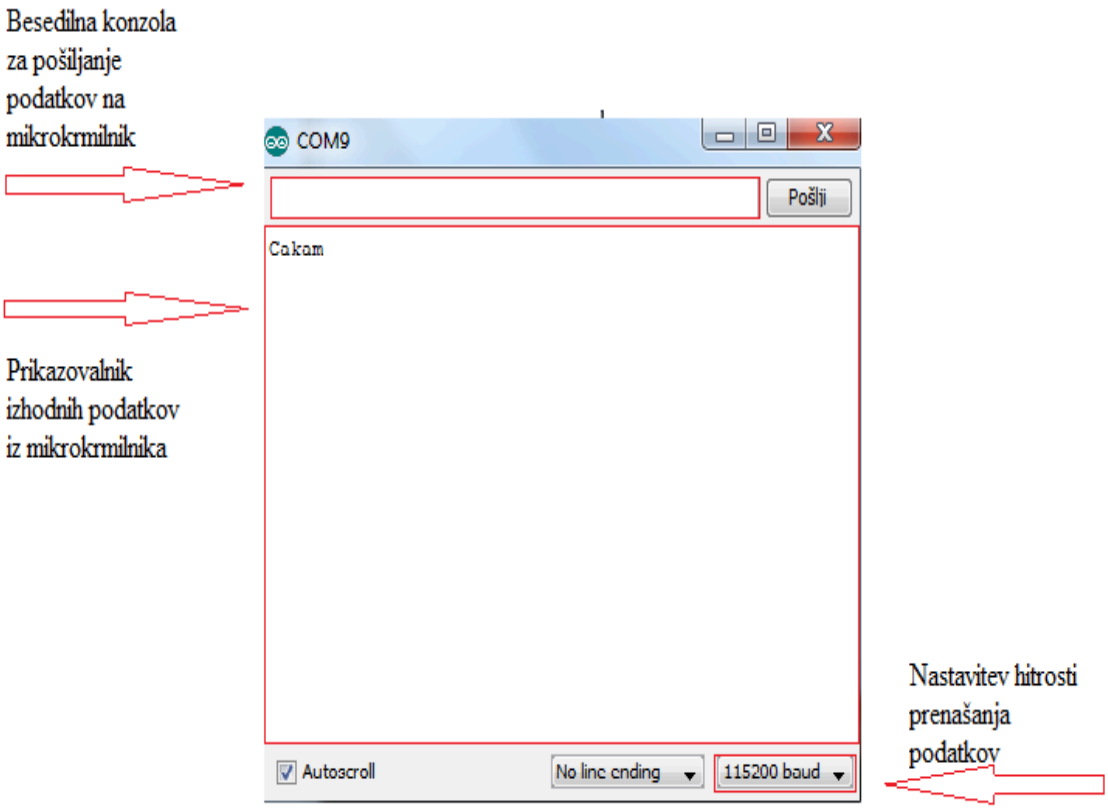


Slika 8: Programsko okolje Arduino

Pred nalaganjem skice na ploščo je potrebno v meniju »Orodja« izbrati njeno ime, npr. Arduino Uno, ter serijski vmesnik, na katerega je priključena npr. COM8. Ko sta

izbrani pravilna povezava in ploščica, lahko naložimo program s pritiskom na gumb »Naloži«. Pri tem se ploščo najprej avtomatsko ponastavi, nato se naloži program.

Okolje omogoča tudi spremljanje podatkov preko serijske komunikacije USB na virtualnem monitorju Serial Monitor (prikazan na Sliki 9). Omogoča tudi pošiljanje podatkov na ploščo. Podatke zapišemo v besedilno konzolo in stisnemo gumb »Pošlji«. Da komunikacija deluje pravilno, moramo nastaviti enako hitrost prenosa v kodi programa oz. v funkciji »setup()« in na zavihku v spodnjem desnem kotu monitorja. [5]



Slika 9: Serial Monitor

4 Merjenje histerezne krivulje

4.1 Magnetne lastnosti snovi

Magnetno polje se nahaja v okolici električnega toka in je posledica gibanja nabojev. Prisotno pa je tudi v okolici trajnih magnetov. Sklepamo, da v trajnem magnetu obstaja električni tok, ki ustvarja to polje. Izkazalo se je, da električne tokove v magnetu ustvarjajo gibajoči elektroni, ki krožijo okrog atomskega jedra. Kroženje (spin) elektrona povzroča magnetni dipolni moment, opisan z enačbo:

$$m = I \cdot A, \tag{4.1}$$

kjer je I tok, ki ga povzroča krožeči elektron, in A površina zanke krožečega elektrona. Vsak elektron v atomu ustvarja svoje magnetno polje, zato imajo vsi atomi določeno magnetno izraženost. Zaradi različnih smeri gibanj posameznih elektronov v atomu ima večina elementov majhno magnetno izraženost. Skupina petih elementov ((Fe) železo, (Ni) nikelj, (Co) kobalt, (Gd) gadolinij, (Dy) disprozij), imenovanih feromagnetiki, ima močno izraženo magnetno polje, saj se njihovi dipolni magnetni momenti ne izničujejo. Tako lahko tvorijo trajne magnete, ki jih predstavimo kot skupek tokovnih zankic (dipolnih momentov), katerih vsota povzroča skupno magnetno polje. Za lažjo obravnavo magnetnih lastnosti modeliramo vpliv vseh tokovnih zankic z vektorjem magnetizacije:

$$\vec{M} = lim_{\Delta V \to 0} \frac{\vec{m}}{\Delta V}, \tag{4.2}$$

kjer $\vec{m}$ predstavlja povprečje dipolnih magnetnih momentov na enoto volumna ΔV. S tem lahko zaključimo, da je magnetno polje izključno posledica gibanja električnih nabojev. Nadaljnje lahko ugotovimo, da sta polja v okolici ravne tuljave in trajnega magneta enaka. Trajni magnet lahko potem predstavimo kot tuljavo z N ovoji in magnetizacijskim tokom I_m:

$$M = \frac{N \cdot I_m}{l}, \tag{4.3}$$

pri čemer je I_m fiktiven tok, ki ga zapišemo le za lažjo predstavo in izračun. Postavitev feromagnetika v tuljavo povzroči povečanje magnetnega polja v tuljavi.

Povečanje je posledica magnetizacije feromagnetika oziroma usmerjanje magnetnih dipolnih momentov v smeri polja, kar zapišemo z enačbo:

$$\frac{B}{\mu} = \frac{N \cdot I}{l} + M, \tag{4.4}$$

kjer je B gostota magnetnega pretoka, N število ovojev tuljave, l srednja dolžina tuljave, I tok skozi tuljavo in M vektor magnetizacije. Po preureditvi enačbe dobimo:

$$\oint H \cdot dl = N \cdot I, \tag{4.5}$$

kjer je vektor H magnetna poljska jakost:

$$H = \frac{B}{\mu} - M \tag{4.6}$$

iz enačbe 4.4

Enačba 4.5 predstavlja Amperov zakon. Zapis Amperovega zakona z vektorjem H je bolj splošen, saj vidimo da je H neodvisen od prisotnosti magnetnih materialov v magnetnem polju. Za povezavo H in M uvedemo magnetno susceptibilnost χ. To je snovna lastnost in predstavlja dovzetnost materiala za magnetenje. Povezavo zapišemo z enačbo:

$$M = \chi \cdot \mathrm{H}, \tag{4.7}$$

kjer je M magnetizacija, H jakost magnetnega polja in χ susceptibilnost. Skupno povezavo med B, H in M predstavlja enačba:

$$B = \mu_0(H + M), \tag{4.8}$$

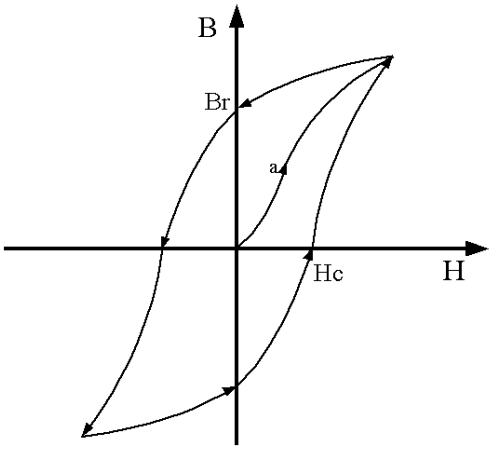
kjer je μ_0 permeabilnost praznega prostora. Enačbo lahko preuredimo z upoštevanjem zveze med H in M. Dobimo:

$$B = \mu_0(1 + \chi) \cdot H = \mu_0 \mu_r H, \tag{4.9}$$

kjer je μ_r relativna permeabilnost, ki je za feromagnetike nelinearna in je funkcija vzbujanja. Poleg tega se pri izklopu vzbujanja spreminja drugače kot pri vklopu. Ta lastnost se imenuje histerezna zanka.

Meritev krivulje magnetenja se opravlja za ugotavljanje magnetnih lastnosti materiala. Pove nam, kako se magnetizacija spreminja z večanjem vzbujalne gostote magnetnega pretoka. Jakost magnetnega polja H prikazuje zunanje vzbujanje.

Gostota magnetnega pretoka B nam prikazuje rezultat magnetenja, ki je v povezavi s H po enačbi 4.9. Tako dobimo B(H) krivuljo prikazano na Sliki 10. [6]



Slika 10: Histerezna zanka

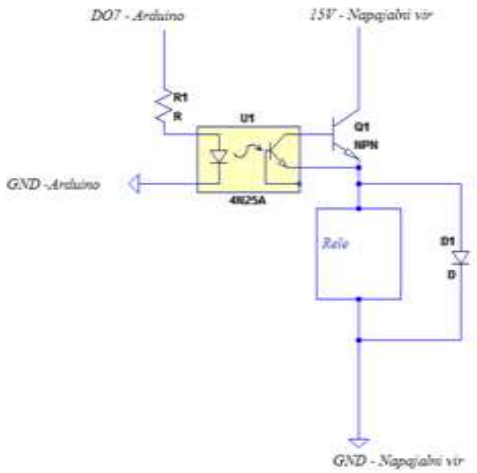
4.2 Postopek meritve

Meritev poteka tako, da postopno zvišujemo tok skozi tuljavo, v kateri je železno (feromagnetno) jedro. S tem se veča vrednost H po enačbi 4.5. Jedro ima zunaj tuljave zračno režo, v katero vstavimo Hallovo sondo, ki je priključena na teslameter. Z njim merimo B. Z večanjem toka se jedro magneti. Krivulja magnetenja strmo narašča le do kolenske vrednosti oz. do magnetnega nasičenja. To pomeni, da je μ_r enaka 1 in povečanje polja s strani feromagnetika ni več mogoče. B se veča le zaradi večanja zunanjega vzbujanja. Dobimo deviško krivuljo magnetenja, ki je na Sliki 10 označena s črko a. Ta krivulja velja le za prvo magnetenje. Pri zmanjševanju vzbujanja se B zmanjšuje počasneje kot pri naraščanju in pri H = 0 B ostane na vrednosti, imenovani remanentna gostota magnetnega polja. Z večanjem vzbujanja v negativno smer B pride na vrednost 0. Točka se imenuje koercitivna jakost polja. Z nadaljnjim večanjem polja v negativni smeri dosežemo nasičenje v le-tej. S ponovnim zmanjšanjem H do 0 in nato večanjem v pozitivno dobimo zaključeno zanko, imenovano histereza.

4.3 Branje meritev

Za izris histerezne zanke pri realizaciji sistema sem moral izmeriti magnetno poljsko jakost H in gostoto magnetnega pretoka B.

V ta namen sem moral železno jedro magnetiti tudi z negativnim tokom. Na tokovnem izvoru sem lahko večal tok le v pozitivno vrednost. Za rešitev tega problema sem uporabil rele, ki je povezan tako, da ob vklopu menja smer toka na svojem izhodu. Rele sem krmilil z digitalnim izhodom na platformi. Izhod lahko odda največji tok 50 mA, kar je bilo za rele, ki za vklop potrebuje minimalen tok 200 mA, premalo. Signal sem ojačil z NPN tranzistorjem, s kolektorjem, priključenim na napetostni izvor 15 V, ter emiterjem na napajanje releja. Krmilni signal iz digitalnega izhoda sem povezal preko optokoplerja na bazo tranzistorja za galvansko ločitev dveh vezij. Vezje je prikazano na Sliki 11.



Slika 11: Shema vezja napajanja in krmiljenja releja

Meritve sem zajemal na analognih vhodih platforme Arduino Uno. Ti omogočajo merjenje napetosti med 0 V in 5 V preko 10 bitnega A/D pretvornika, kar pomeni približno 5 mV natančnost.

Magnetna poljska jakost je proporcionalna el. toku po enačbi 4.5. H lahko izrazimo posredno iz toka, zato sem moral meriti tok skozi tuljavo. S platformo lahko merimo le pozitivne vrednosti napetosti, zato sem tuljavi zaporedno vezal tokovni merilni upor nazivne vrednosti 2 Ω. V vezju sem ga postavil na pozitivni sponki tokovnega vira pred tokovnim relejem, ki zamenja smer toka skozi tuljavo. S tem sem se izognil negativnim vrednostim napetosti, ki bi lahko uničile platformo. Razmerje med tokom in napetostjo je 1:2, kar pomeni da sta 2 V padca napetosti na merilnem uporu enaka 1 A toka skozi tuljavo. Merilni upor sem povezal z analognim vhodom A1.

Za meritve gostote magnetnega pretoka sem uporabil teslameter. Ta deluje na principu Hallovega efekta, ki opisuje delovanje Lorentzove sile na elektrone (ki predstavljajo električni tok) pri prevodniku ali polprevodniku v homogenem magnetnem polju. Lorentzova sila:

$$F = -e \cdot (E + v \times B), \tag{4.10}$$

kjer je F Lorentzova sila, e električni naboj, E električna poljska jakost, v hitrost naboja in B gostota magnetnega pretoka. Sila povzroča odklon gibajočih elektronov prečno na smer gibanja. Odklon ima za posledico pojav napetosti v prečni smeri – Hallova napetost, ter povečanje upornosti v primarni smeri gibanja nosilcev naboja – magnetoupornost. Pojav je bolj izrazit v polprevodnikih, kjer so nosilci naboja tudi vrzeli. Hallovo napetost lahko izračunamo po enačbi:

$$U_{H=\frac{R_X \cdot I_X \cdot B_Z}{d}}, \tag{4.11}$$

kjer je B_Z homogeno magnetno polje, I_X je konstantni tok skozi Hallov element, d debelina elementa ter R_X Hallov koeficient. Hkrati je napetost zelo majhna, zato je v celotni izvedbi teslametra poskrbljeno za napajanje s konstantnim tokom, ojačenje in temperaturno kompenzacijo Hallove napetosti. Hallov senzor omogoča zajemanje enosmernih in izmeničnih magnetnih polj.

Teslameter, ki sem ga uporabil, je imel transverzalno Hallovo sondo. Pri uporabi sem moral merilnik najprej kalibrirati in nastaviti ničlo tako, da sem postavil sondo v celico, kjer ni magnetnega polja. Izhod iz teslametra sem moral nato priključiti na analogni vhod Arduina. Ob menjavi smeri toka skozi tuljavo sem spremeni tudi smer Hallove napetosti na sondi. To bi povzročilo, da bi bila na digitalnem vhodu negativna napetost, kar je nedopustno. V ta namen sem moral realizirati seštevalnik napetosti, ki referenčni napetosti prišteva Hallovo napetost iz teslametra v napetostnem območju 0 V do 5 V.

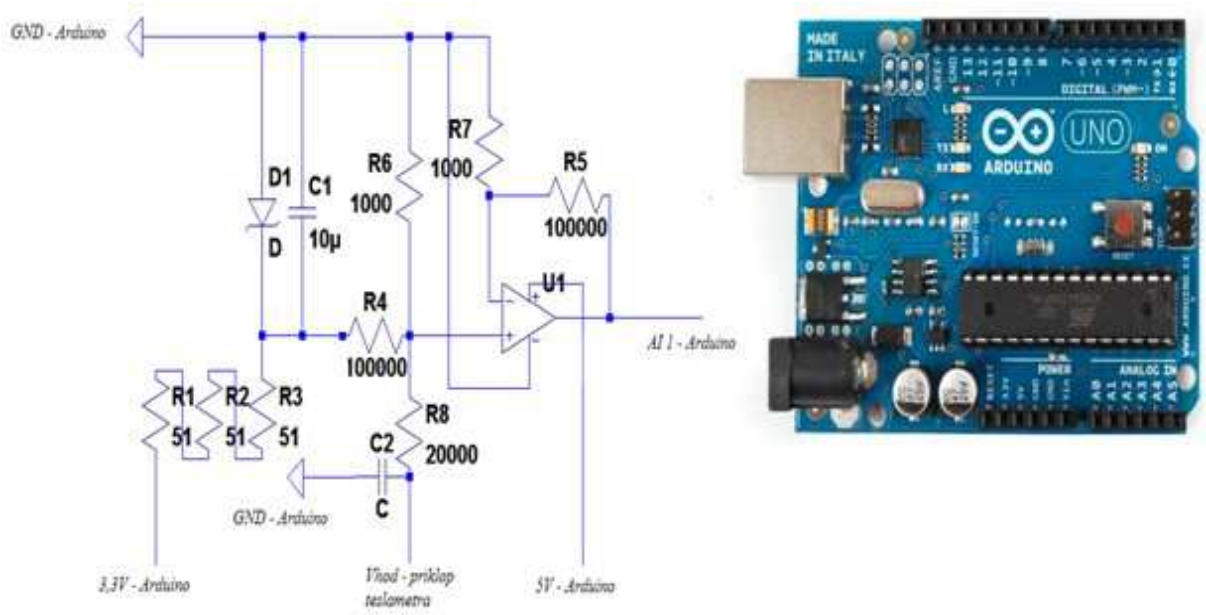
Za seštevalnik napetosti sem uporabil operacijski ojačevalnik LM358, za katerega sem se odločil na podlagi več dejavnikov. Napajanje operacijskega ojačevalnika sem dovedel iz platforme Arduino Uno, zato sem bil pri izvedbi omejen z enostranskim napajanjem in nizkim napajalnim tokom. LM358 ima:

- možnost enostranskega napajanja (single supplay 3 V–32 V), ki omogoča direktno zaznavanje blizu GND, zato omogoča izhodno napetost iz ojačevalnika do 0 V,
- majhno vrednost izgubnega napajalnega toka – 500 µA neodvisna od napajalne napetosti,
- majhen napajalni tok – 1,2 mA.

Izvedel sem neinvertirajoči seštevalnik, brez ojačenja referenčne napetosti in s približno petkratnim ojačenjem Hallove napetosti iz teslametra. [7]

Referenčno napetost sem dovedel iz 3,3 V reguliranega izhoda na platformi. Napetost sem znižal in stabiliziral na 2,5 V. Za stabilizacijo sem uporabil Zenner diodo BZX 2Y4, kateri sem omejil tok s tremi 51 Ω upori. Za večjo stabilizacijo sem Zenner diodi paralelno priključil še elektrolitski kondenzator 10 µF 63 V. Izhod iz operacijskega ojačevalnika sem povezal z analognim vhodom A0 na platformi. Celotno vezje je na Sliki 12.

Celotno vezje z relejem in Arduinom sem postavil v ohišje.



Slika 12: Merilno vezje Hallove napetosti

5 Rezultati meritev

Pri izvedbi meritev histerezne zanke s pomočjo sistema za avtomatično upravljanje ročnega potenciometra sem priključil vse merjene in merilne elemente. To so:

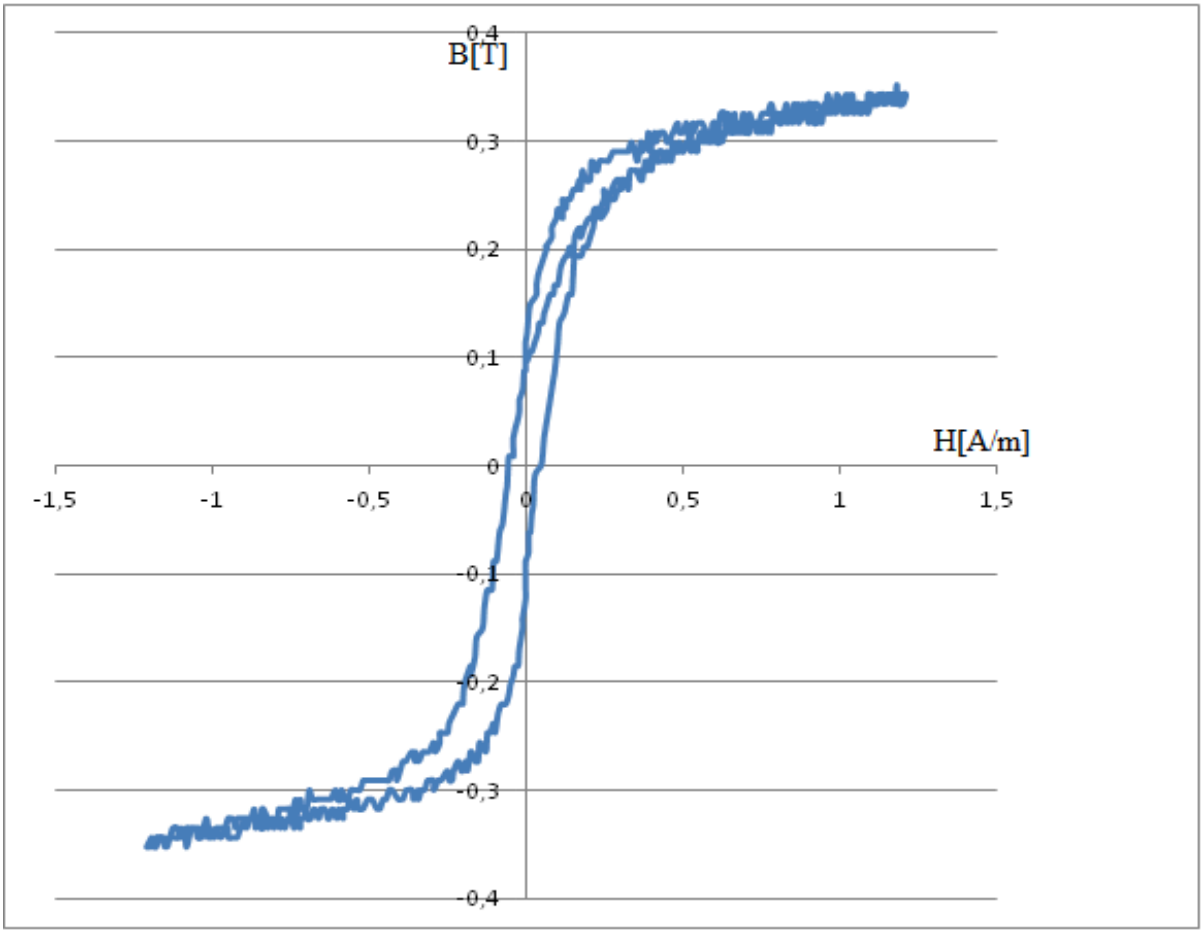
- feromagnetno jedro z navitjem,
- enosmerni vir,
- teslameter s ploščato Hallovo sondo,
- koračni motor,
- mikrokrmilnik za kriljenje motorja in zajem meritev,
- računalnik za zagon meritve ter shranjevanje in obdelavo izmerjenih podatkov.

Ob prvi meritvi sem moral postaviti potenciometer tokovnega izvora skrajno v levo na vrednost 0 in ga povezati s koračnim motorjem. Koračni motor se zavrti za določeno število korakov v desno do vrednosti toka 1,5 A. Nato se vrne za enako število korakov v nasprotno smer, da je nazaj na vrednosti 0. S tem sem se izognil morebitnim poškodbam potenciometra.

Meritev poteka tako, da z računalnikom iz programskega okolja Arduino najprej naložimo programsko kodo na mikrokrmilnik. Ko je koda naložena, se na njem začne izvajati program. Nato s klikom na ikono odpremo SerialMonitor. Odpre se nam okno, kjer se izpisuje napis: »Čakam – Vpiši Start«. Ob vpisu besede »Start« se začne meritev histerezne zanke. V oknu se nam zatem začnejo v dveh stolpcih, ločenih s podpičjem, izpisovati izmerjene vrednosti B in H. Ob koncu meritve se ponovno v oknu začne izpisovati »Čakam – Vpiši Start«. Oba stolpca skopiramo v beležko in shranimo dokument kot datoteko s končnico -.csv. Nato lahko dokument odpremo v excel-u, kjer izrišemo graf.

Preskusil sem več postopkov merjenja tako, da sem spreminjal programsko kodo ter merilno vezje. Pri prvih meritvah sem preskusil postopek, pri katerem se motor vrti brez daljšega ustavljanja. Naraščanje in padanje magnetilnega toka je konstantno od 0 A do 1,3 A. Hkrati z vrtenjem sem poskušal izvesti zajem obeh merjenih veličin. Popolne sinhronizacije nisem mogel izvesti, saj mikrokrmilnik Arduino Uno lahko požene le eno funkcijo v določenem trenutku. Da bi bili obe meritvi zajeti ob enako

odprtem potenciometru tokovnega izvora, sem ukaza branja in izpisa analognih vhodov umestil kar v samo funkcijo vrtenja koračnega motorja pred stavke switch. To je bilo mogoče, saj se motor vrti po majhnih korakih in ne zvezno. Obe merjeni veličini sem zajemal brez programskih in električnih filtrov.

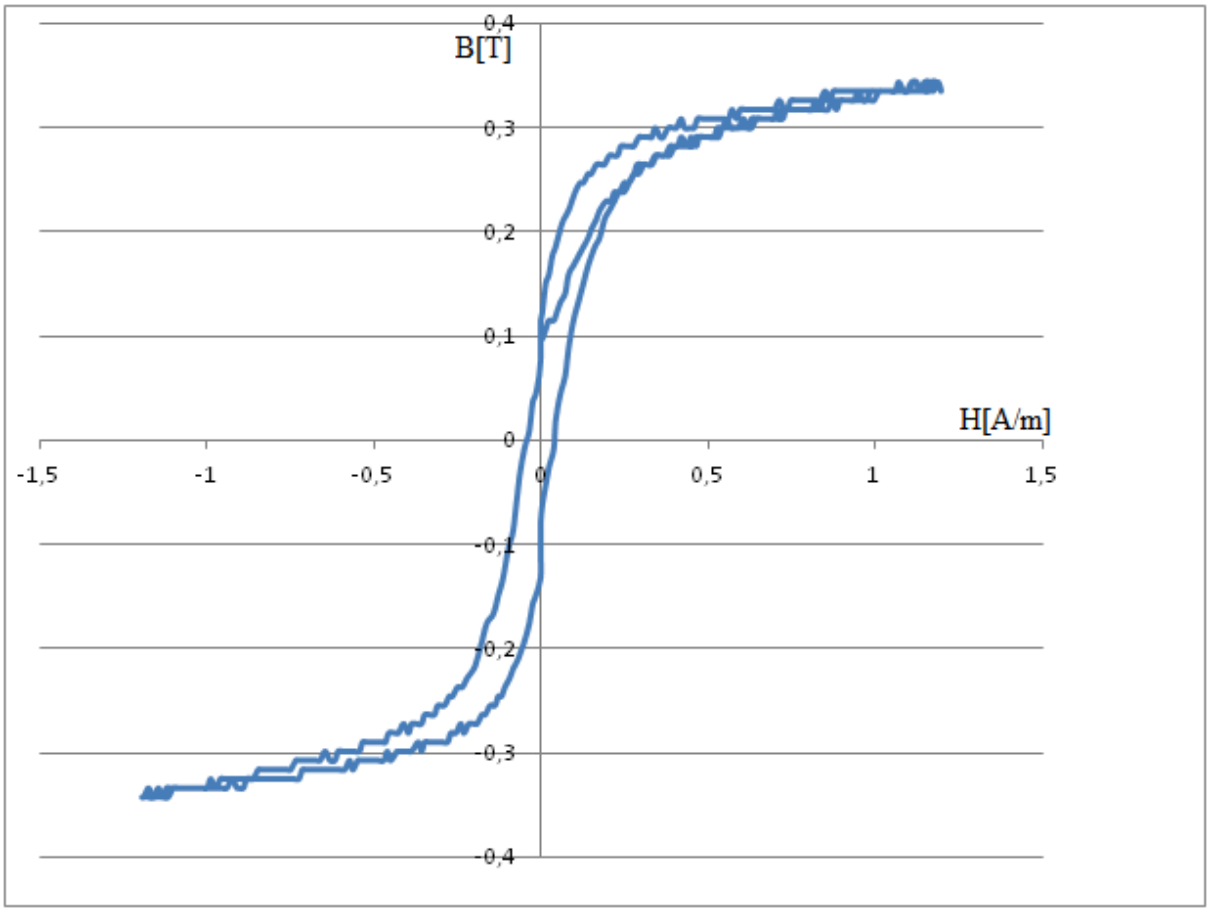


Slika 13: Graf 1. meritve

Rezultat meritve je prikazan na Sliki 13. V primerjavi z rezultati, ki sem jih dobil pri ročni meritvi, je oblika histereze zelo podobna. Obe histerezi sta ozki, kar je značilno za mehke feromagnetike, kot je železo. Največja razlika med meritvama je nekostantnost naraščanja B z naraščanjem H oziroma nazobčanost krivulje pri avtomatizirani meritvi.

Bolj gladko obliko histereze brez navideznega šuma sem poskusil realizirati s spremembo programa tako, da sem najprej opravil meritev in šele nato premik motorja po določeni zakasnitvi. S tem sem se poskušal izogniti dinamičnim razmeram, ki nastanejo ob premiku motorja. Poleg tega sem nazobčanost grafa poskusil izničiti tako, da sem na izhod teslametra dodal RC filter, ki sem ga izvedel s

kondenzatorjem 1 µF 63 V in uporom 1 kΩ. V programu sem zapisal funkcijo branja meritev, ki deluje kot programski filter. Funkcija ob klicu (v trenutku ko motor miruje) opravi deset meritev obeh vhodnih signalov in izračuna njuni povprečni vrednosti.



Slika 14: Graf 2. meritve

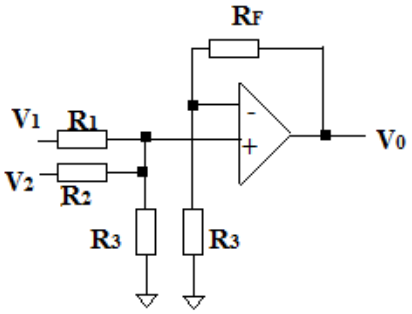
Kljub vsem izboljšavam se meritev ni bistveno izboljšala. Krivulja na Sliki 14 je bila še vedno dokaj nazobčana.

Ob opazovanju velikosti merjenih signalov sem ugotovil, da je velikost Hallove napetosti na teslametru relativno zelo majhna (-350 mV do 350 mV) v primerjavi s celotnim referenčnim območjem 10-bitnega A/D pretvornika (0 V do 5 V). To je pomenilo, da imam izkoriščenih le 15,9 % možnega merilnega območja. Pri 5 V referenčnem merilnem območju in 10-bitnem A/D pretvorniku predstavlja 1 bit 4,89 mV. Razred točnosti, ki je definiran kot odstotek neločljivosti v polnem odklonu merilne naprave, je v tem primeru 0,0978 %. Pri enaki ločljivosti (1 bit 4,89 mV) in 15,9 % izkoriščenem 5 V merilnem območju se razred točnosti poveča na vrednost 0,699 %. Natančnost merjenja se poslabša za več kot 7-krat.

Za povečanje natančnosti merjenja pri isti ločljivosti sem moral zvečati velikost Hallove napetosti. V ta namen sem spremenil merilno vezje (Slika 15) oziroma seštevalnik napetosti. Velikost izhodne napetosti iz seštevalnika je določena z enačbo:

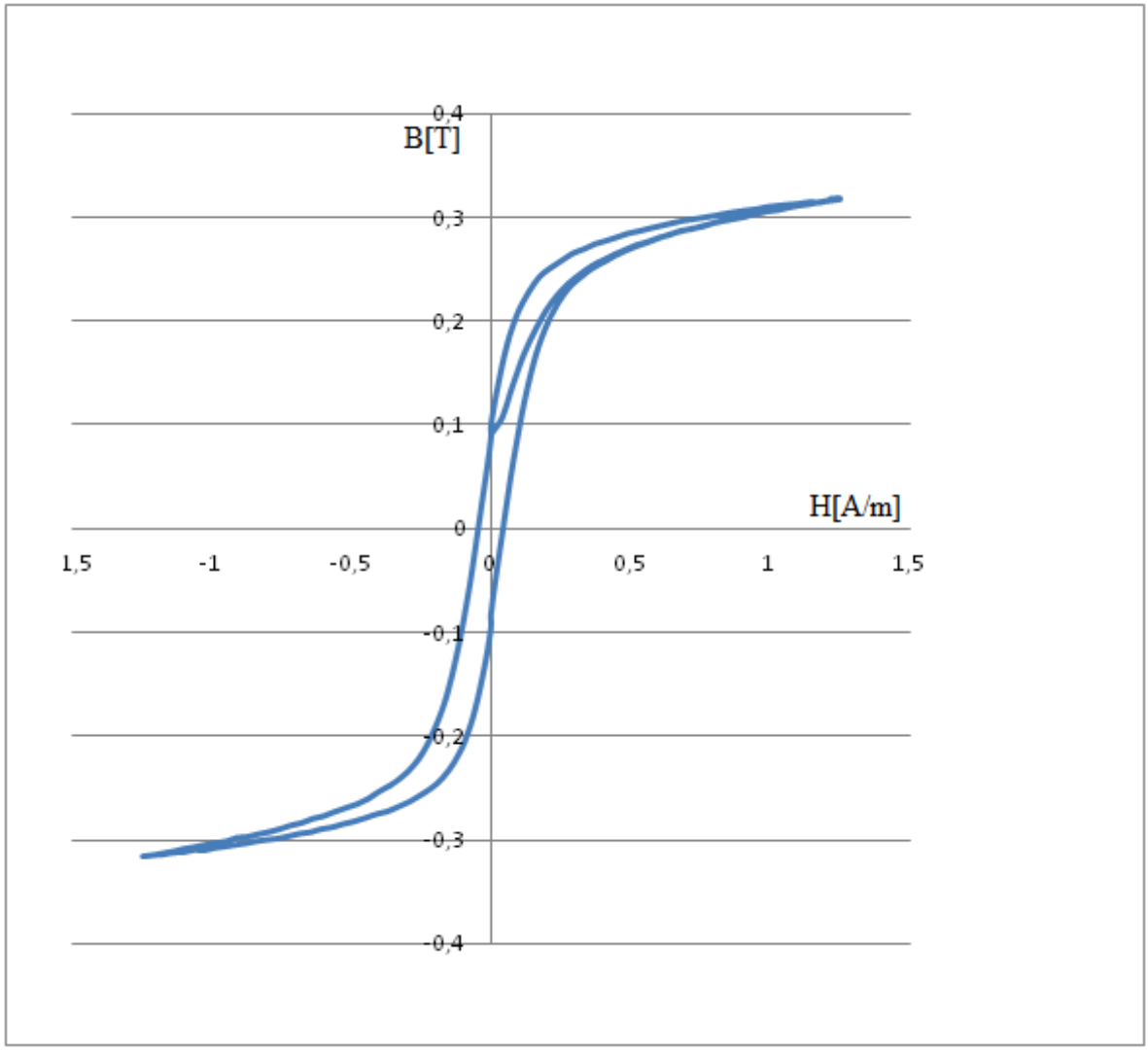
$$V_0 = \frac{(R_3+R_F)\cdot(V_1\cdot R_2+V_2\cdot R_1)}{(R_1\cdot R_2+R_2\cdot R_3+R_1\cdot R_3)}, \quad (5.1)$$

kjer je V_0 izhodna napetost, V_1 konstantna napetost 2,5 V, V_2 Hallova napetost iz teslametra, R_1 ohmski upor 100 kΩ, R_2 ohmski upor 20 kΩ, R_3 ohmski upor 1 kΩ, R_F ohmski upor 100 kΩ.



Slika 15: Shema napetostnega seštevalnika

Pred spremembo sta bila upora R_1 in R_2 enakih vrednosti(100 kΩ). Ko je velikost upora R_F 100 kΩ in uporov R_3 1 kΩ, je izhodna napetost $V_0 \approx 0{,}99 \cdot (V_1 + V_2)$. S spremembo upora R_2 na vrednost 20 kΩ se izhodna napetost spremeni po enačbi $V_0 \approx 0{,}95 \cdot V_1 + 4{,}76 \cdot V_2$. Referenčna napetost se na izhodu nekoliko zniža (pomembno pri izračunu meritev v programu), Hallova napetost pa se ojači za faktor 4,76. V realnosti izmerjene vrednosti nekoliko odstopajo od izračunanih. Referenčna napetost se spusti na 2,2 V, ojačenje Hallove napetosti pa je približno 4,5. Odstopanja so posledica toleranc uporov. Pri isti ločljivosti in sedaj spremenjenem vezju se razred točnosti zmanjša na 0,15 %. Graf 3. meritve je prikazan na Sliki 16.



Slika 16: Graf 3. meritve

Oblika grafa se je po spremembi bistveno izboljšala. Funkcija je gladka in ustreza pričakovanim meritvam.

6 Zaključek

Pri ocenjevanju kakovosti sistema sem se osredotočil na tri področja:

- hitrost merilnega sistema,
- natančnost,
- praktičnost uporabe sistema.

S stališča hitrosti sistema sem pričakoval, da se avtomatizirana meritev opravi hitreje kot pri ročnem postopku merjenja. To sem dosegel, vendar razlika ni zelo velika.

Sistem bi bilo mogoče še nekoliko pospešiti z dvigom napajalne napetosti koračnega motorja s 5 V na 12 V. Dodatno lahko zvečamo hitrost vrtenja motorja tudi programsko, in sicer s spremembo funkcije vrtenja motorja tako, da postopno večamo hitrost ob zagonu od 0 do maksimalnega števila obratov. Iz trenutnih približno 15 obr/min bi z obema spremembama dosegli približno 25 obr/min.

Pri preskušanju sistema sem ugotovil, da se meritev z ustavljanjem motorja pri branju rezultatsko ne razlikuje od meritve, kjer ustavljanja ni. Sistem bi lahko nekoliko pohitril tudi s tem, da bi funkcijo branja vstavil v funkcijo vrtenja motorja in se znebil kratkega ustavljanja ob vsaki meritvi.

Drugo pomembno področje kakovosti sistema je natančnost. S testiranjem sem ugotovil, da je natančnost dokaj visoka kljub relativno majhnemu 10 bitnemu A/D pretvorniku. Za odpravo nekaterih motenj sem sistemu dodal programski in pa RC filter. Dobro stabilizirani (na mV natančno) sta tudi napetosti iz pina 3,3 V in pina 5 V, s katerima sem realiziral merilno vezje.

Pomemben dejavnik s strani uporabnika sistema je praktičnost oz. enostavnost uporabe sistema. V trenutni izvedbi je s tega vidika nekaj pomanjkljivosti. Sistem sem nadzoroval kar s samim programskim orodjem Arduino, in sicer preko serijskega komunikacijskega okna Serial Monitor. To je virtualni ekran, preko katerega poteka komunikacija z odprtokodno platformo Arduino Uno. Tu poženemo sistem z vpisom ukaza »Start« v besedilno konzolo. V prikazovalniku izhodnih podatkov se nam nato izpisujejo izmerjene vrednosti. Slabost sistema je v tem, da moramo po opravljeni meritvi zapisane podatke v oknu označiti in skopirati v prazno beležnico (NotePad). Nato moramo podatke shraniti kot .csv datoteko. To lahko odpremo s programom

Microsoft Excel, kjer se nam rezultati dveh merjenih veličin zapišejo v dveh stolpcih. Tu lahko podatke obdelujemo in izrišemo grafe.

Za lažje upravljanje s sistemom bi lahko v skriptnem programskem jeziku Python sprogramiral grafični vmesnik, preko katerega bi komuniciral s platformo. Aplikacija bi omogočala zagon in zaustavitev sistema ter sprotno izrisovanje 2D grafa.

Sistem, kot je realiziran sedaj, ni dovolj praktičen za širšo uporabo kljub zadovoljivi natančnosti. Menim pa, da bi bil sistem z opisanimi izboljšavami primeren za demonstrativni prikaz manjšega mehatroničnega postroja na laboratorijskih vajah.

7 Viri

[1] O. Jamal, S. Khan, Zainulabideen, »PC Based Wireless Stepper Motor Control« [Online] Dosegljivo: http://www.bth.se/fou/cuppsats.nsf/all/62c86cab2e5fdaa9c1257b7c0064c215/$file/BTH%202013%20Abideen.pdf [Dostopano: 10. 6. 2014]

[2] »28BYJ-48 Stepper Motor with ULN2003 driver and Arduino Uno« [Online] Dosegljivo: http://42bots.com/tutorials/28byj-48-stepper-motor-with-uln2003-driver-and-arduino-uno/ [Dostopano: 10. 6. 2014]

[3] »Arduino« [Online] Dosegljivo: http://arduino.cc/ [Dostopano: 15. 7. 2014]

[4] »Arduino Uno« [Online] Dosegljivo: http://arduino.cc/en/Main/ArduinoBoardUno [Dostopano: 15. 7. 2014]

[5] »Arduino Development Environment“ [Online] Dosegljivo: http://arduino.cc/en/Guide/Environment [Dostopano: 17. 7. 2014]

[6] D.Križaj, »Osnove elektrotehnike II – Magnetostatika« [Online] Dosegljivo: http://lbm.fe.uni-lj.si/oe/OE2/oe2magn_2011.pdf [Dostopano: 21. 7. 2014]

[7] »LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers« [Online] Dosegljivo: http://www.ti.com/lit/ds/symlink/lm158-n.pdf [Dostopano: 21. 7. 2014]

8 Dodatek

8.1 Programska koda

```
int Pin0 = 9;  // 9 - 12 so številke digitalnih izhodov, ki jih potrebujemo za krmiljenje koračnega motorja.
int Pin1 = 10;
int Pin2 = 11;
int Pin3 = 12;

int Pin7 = 7;  // "Pin7" je digitalni izhod, ki ga potrebujemo za krmiljenje releja preko tranzistorja.


int delaj =0; // Spremenljivko "krog" potrebujemo za pogoj v stavku "switch". Začetna vrednost mora biti 0.
int ms = 30;  // Spremenljivka "ms" hrani vrednost časa v milisekundah, ki poteče preden se nadaljuje izvajanje
// funkcije

// Za branje podatkov iz analognih vhodou potrebujemo spremenljivko, kjer se shrani izmerjena
// vrednost. Da je meritev bolj natančna funkcija "branje" opravi 10 meritev iz vsakega vhoda.
// Med posameznimi zapisi meritev je nastavljeno čakanje do naslednje meritve, saj sprememba
// magnetnega pretoka ni skočna.
void branje(int Vr1_vhod0,int Vr2_vhod0,int Vr3_vhod0,int Vr4_vhod0,int Vr5_vhod0,int Vr6_vhod0,int
Vr7_vhod0,int Vr8_vhod0,int Vr9_vhod0,int Vr10_vhod0,int Vr_pov_vhod0,int Vr1_vhod1,int Vr2_vhod1,
int Vr3_vhod1,int Vr4_vhod1,int Vr5_vhod1,int Vr6_vhod1,int Vr7_vhod1,int Vr8_vhod1,int Vr9_vhod1,
int Vr10_vhod1,int Vr_pov_vhod1){

  Vr1_vhod0 = analogRead(A0); // Spremenljivkam pripišemo vrednost meritve s funkcijo "analogRead(št_vhoda)".
  Vr1_vhod1 = analogRead(A1);
  delay(ms);
  Vr2_vhod0 = analogRead(A0);
  Vr2_vhod1 = analogRead(A1);
  delay(ms);
  Vr3_vhod0 = analogRead(A0);
  Vr3_vhod1 = analogRead(A1);
  delay(ms);
  Vr4_vhod0 = analogRead(A0);
  Vr4_vhod1 = analogRead(A1);
  delay(ms);
  Vr5_vhod0 = analogRead(A0);
  Vr5_vhod1 = analogRead(A1);
  delay(ms);
  Vr6_vhod0 = analogRead(A0);
  Vr6_vhod1 = analogRead(A1);
  delay(ms);
  Vr7_vhod0 = analogRead(A0);
  Vr7_vhod1 = analogRead(A1);
  delay(ms);
  Vr8_vhod0 = analogRead(A0);
  Vr8_vhod1 = analogRead(A1);
  Vr9_vhod0 = analogRead(A0);
  Vr9_vhod1 = analogRead(A1);
  delay(ms);
  Vr10_vhod0 = analogRead(A0);
  Vr10_vhod1 = analogRead(A1);

  // V spremenljivki "Vr_pov_vhod0" in "Vr_pov_vhod1" vpišemo povprečni vrednosti obeh analognih vhodov.
  // Funkcija "analogRead()" prebere napetost na analognem vhodu med 0V in 5V. Ta vrednost se pretvori na
  // 10 bitnem AD pretvorniku in funkcija vrne vrednost med 0 in 1023. Da ob deljenju zadržimo vrednost
  // natančno na desetinko, najprej vsoto pomnožimo z 10, nato pa delimo. Pri tem moramo paziti, da vrednost
  // vsote ne preseže vrednosti 32767 kot je območje podatkovnega tipa int.
```

```
  Vr_pov_vhod0 = ((((Vr1_vhod0+Vr2_vhod0+Vr3_vhod0+Vr4_vhod0+Vr5_vhod0)*10/5)
  +((Vr6_vhod0+Vr7_vhod0+Vr8_vhod0+Vr9_vhod0+Vr10_vhod0)*10/5))/2);

  Vr_pov_vhod1 = ((((Vr1_vhod1+Vr2_vhod1+Vr3_vhod1+Vr4_vhod1+Vr5_vhod1)*10/5)
  +((Vr6_vhod1+Vr7_vhod1+Vr8_vhod1+Vr9_vhod1+Vr10_vhod1)*10/5))/2);



  Serial.print(Vr_pov_vhod0);   // Vrednost meritve zapišemo v "serial monitor" s funkcijo "Serial.print()".
  Serial.print(";");            // Ta faunkcija zapiše vrednost spremenljivke "Vr_pov_vhod0" v vrstici.
  Serial.println(Vr_pov_vhod1); // Da lahko kasneje meritve shranimo v excelovi datoteki, moramo med obe vrednosti
                                // postaviti podpičje. Drugo vrednost zapišemo s funcijo "Serial.println()". Ta
}                               // funkcija zapiše vrednost v trenutno vrstico in jo zaključi. Vsi nadaljni
                                // zapisi v "serial monitor" se bodo zapisali v sledečo vrstico, do naslednjega
                                // klica te funcije.

// Da lahko kontroliramo koračni motor, moramo prvilno vklapljati digitalne izhode, ki prižigajo tranzistorje na gonilniku ULN 20003.
// Motor vzbujamo s polovičnim korakom, zato potrebujemo za en obrat motorja osem sekvenc. Sekvence se ponavljajo pri vsakem obratu motorja,
// zato napišemo funkcijo "motor", da zmanjšamo obseg kode.

// Funcija "motor" ne vrne nobene vrednosti, zato je tipa "void". Ima pa 7 parametrov.

// S parameterom "st_korakov" določimo, koliko obratov naredi motor ob klicu funkcije motor. Za točno določitev zasuka moramo upoštevati še
// prestavno razmerje motorja, ki je 1/64. To pomeni, da se mora za en celoten obrat gredi motorja menjati približno 4076 korakov.

// S parametrom "_step" nadzorujemo katera sekvenca se bo izvedla

// S parametrom "dir" določimo v katero smer se bo vrtel motor. V primeru da je "dir = true", se motor zavrti v pozitivno smer.

// Parametri "Pin0" do "Pin3" predstavljajo digitalne izhode od DI9 do DI12.

void motor(int st_korakov, int _step, boolean dir, int Pin0, int Pin1, int Pin2, int Pin3){

  for(int x = 0; x < st_korakov; x++) // S for zanko nadzorujemo število ponovitev osmih sekvenc. Zanka se izvaja dokler je "x" manjši
    // od parametra "st_korakov".
  {

    switch(_step){                  // S stavkom switch nadzorujemo potek sekvenc vklopov in izklopov DI. Parameter _step je ima na
    case 0:                        // začetku vrednost 0. Izvede se primer 0. Vrednosti DI nastavimo s funcijo "digitalWrite",
      digitalWrite(Pin0, LOW);     // v katero zapišemo parameta "Pinx", za določitev številke DI ter "LOW ali HIGH", za določitev
      digitalWrite(Pin1, LOW);     // postavitve DI na brezmapetostno ali napetostno stanje. Stavek "break" prekine izvajanje "switch".
      digitalWrite(Pin2, LOW);     // Nato se preverja pogoj "if(dir)". Če je pogoj "true"
      digitalWrite(Pin3, HIGH);    // se parameter _step poveča za ena, drugače se zmanjša za ena.
      break;                        // Za tem se izvedeta še pogoja "if(_step<0)" in "if(_step>7)". V prvem primeru se parameter "_step"
    case 1:                        // postavi na 7, v drugem pa se postavi na 0, tako da se vrednost "_step" spreminja le v vrednostih
      digitalWrite(Pin0, LOW);     // med 0 in 7.
      digitalWrite(Pin1, LOW);     // Na koncu zakasnimo celotno izvajanje for zanke. S tem kontroliramo hitrost vrtenja motorja. Najmanša
      digitalWrite(Pin2, HIGH);    // vrednost zakasnitve je 1ms. Ta vrednost je določena s fizikalnimi lastnostmi koračnega motorja. Za
      digitalWrite(Pin3, HIGH);    // počasnejše vrtenje motorja nastavimo večjo zakasnitev.
      break;
    case 2:
      digitalWrite(Pin0, LOW);
      digitalWrite(Pin1, LOW);
      digitalWrite(Pin2, HIGH);
      digitalWrite(Pin3, LOW);
      break;
```

```
case 3:
  digitalWrite(Pin0, LOW);
  digitalWrite(Pin1, HIGH);
  digitalWrite(Pin2, HIGH);
  digitalWrite(Pin3, LOW);
  break;
case 4:
  digitalWrite(Pin0, LOW);
  digitalWrite(Pin1, HIGH);
  digitalWrite(Pin2, LOW);
  digitalWrite(Pin3, LOW);
  break;
case 5:
  digitalWrite(Pin0, HIGH);
  digitalWrite(Pin1, HIGH);
  digitalWrite(Pin2, LOW);
  digitalWrite(Pin3, LOW);
  break;
case 6:
  digitalWrite(Pin0, HIGH);
  digitalWrite(Pin1, LOW);
  digitalWrite(Pin2, LOW);
  digitalWrite(Pin3, LOW);
  break;
  case 7:
    digitalWrite(Pin0, HIGH);
    digitalWrite(Pin1, LOW);
    digitalWrite(Pin2, LOW);
    digitalWrite(Pin3, HIGH);
    break;
  default:
    digitalWrite(Pin0, LOW);
    digitalWrite(Pin1, LOW);
    digitalWrite(Pin2, LOW);
    digitalWrite(Pin3, LOW);
    break;
  }
  if(dir){
    _step++;
  }
  else{
    _step--;
  }
  if(_step>7){
    _step=0;
  }
  if(_step<0){
    _step=7;
  }
  delay(1);
}
void loop()                                    // V glavni zanki programa nadzorujemo vrtenje motorja
{                                              // in branje analognih vhodov.
  if(Serial.readString() == "izmeri")          // Uporabimo "switch" stavek, s katerim izbiramo več možnosti.
  {                                            // Parameter "delaj" ima na začetku vrednost 0, zato se izvedejo
    for(int pon = 0; pon<6; pon++)             // stavki v "case 0:".
    {                                          // Rele mora biti v tem primeru sklopljen, zato s funcijo
      {                                        // "digitalWrite()" določimo, da je Pin7 v breznapetostnem stanju.
        switch(delaj){                         // V "for" zanki se najprej izvede funcija "branje" z namenom da se
```

```
      case 0:
        digitalWrite(Pin7, LOW);
        for(int kor = 0; kor<34; kor++){
          branje(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
          motor(750,0,true,Pin0,Pin1,Pin2,Pin3);
        }
        delay(1000);
        break;
      case 1:
        digitalWrite(Pin7, LOW);
        for(int kor = 0; kor<34; kor++){
          branje(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
          motor(750,0,false,Pin0,Pin1,Pin2,Pin3);
        }
        delay(1000);
        break;
      case 2:
  digitalWrite(Pin7, HIGH);
  for(int kor = 0; kor<34; kor++){
    branje(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
    motor(750,0,true,Pin0,Pin1,Pin2,Pin3);
  }
  delay(1000);
  break;
case 3:
  digitalWrite(Pin7, HIGH);
  for(int kor = 0; kor<34; kor++){
    branje(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
    motor(750,0,false,Pin0,Pin1,Pin2,Pin3);
  }
  delay(1000);
  break;
case 4:
  digitalWrite(Pin7, LOW);
  for(int kor = 0; kor<34; kor++){
    branje(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
    motor(750,0,true,Pin0,Pin1,Pin2,Pin3);
  }
  delay(1000);
  break;
 case 5:
   digitalWrite(Pin7, LOW);
   for(int kor = 0; kor<34; kor++){
     motor(750,0,false,Pin0,Pin1,Pin2,Pin3);
   }
   delay(1000);
   break;
 }
          delaj++;
        }
      }
    }
    else
    {
      Serial.print("Cakam");
      delay(5000);
    }
 }
```

```
// izmeri tudi začetno stanje. Nato sledi funcija "motor", ki
// premakne gred motorja za približno 66°. Sledi zakasnitev in izhod
// iz stavka "switch", ki ga izvedemo s stavkom "break".
// Želimo, da se izvede vseh šest primerov stavka "switch" eden za
// drugim, zato na koncu povečamo spremenljiko "delaj" za ena. Vse
// skupaj moramo postaviti še v for zanko, ki se šestkrat izvede.
// V primerih 0-4, spreminjamo samo smer vrtenja motorja s parametroma
// "true" in "false", ter krmilimo tranzistor s postavljanjem pina 7.
// Ostali ukazi so isti.
// V primeru 5 ne potrebujemo branja analognih vhodov, saj želimo
// postaviti potenciometer v začetno stanje. V ta namen izpustimo
// funkcijo "branje".
// Da lahko nadzorujemo kdaj naj se meritev začne vse skupaj postavimo
// v stavek "if" s pogojem "(Serial.readString() == "izmeri")".
// Ta omogoča, da lahko preko "serial monitor" pošljemo ukaz "izmeri"
// na krmilnik, ki požene meritev.
// V nasprotnem primeru se na monitorju vsaki 5s izpiše ukaz "čakam".
```