Automatizirana šahovska ploča

Stemberger, Eros

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Eros Stemberger

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Prof. dr. sc. Željko Šitum, dipl. ing.

Student:

Eros Stemberger

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se obitelji što mi je pružila neizmjernu podršku i njome mi omogućila bezbrižno školovanje.

Također se zahvaljujem mentoru, prof. dr. sc. Željku Šitumu na pomoći i inspiraciji tijekom izrade ovog rada.

Eros Stemberger

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Eros Stemberger** JMBAG: 0035214728

Naslov rada na hrvatskom jeziku: **Automatizirana šahovska ploča**

Naslov rada na engleskom jeziku: **Automated chess board**

Opis zadatka:

Šah je igra koja zbog svoje društvene integriranosti i rastuće popularnosti može služiti za prezentiranje znanja i inovacija iz suvremenog inženjerstva. Šahovska partija od igrača zahtijeva mnoge vještine koje su korisne i u drugim sferama života kao što su strateško razmišljanje, pamćenje i fokusiranost. U okviru zadatka potrebno je integrirati znanja iz različitih područja elektrostrojarstva s ciljem izrade šahovske ploče koja samostalno igra naspram čovjeka. Danas je računalo neizostavno pomagalo igračima šaha zbog procesiranja nevjerojatnog broja mogućih poteza i pozicija koje se za vrijeme jedne partije mogu pojaviti te su idealni za evaluiranje šahovskih pozicija. Ovakav sustav koji integrira računalo u fizičku šahovsku ploču može utjecati na razvoj čovjekovih kognitivnih sposobnosti, poboljšanja pamćenja, vizualnih sposobnosti i opće inteligencije. Suvremena računalna tehnologija potaknula je novi način igranja šaha i produbila raspravu o umjetnoj inteligenciji i izazovima koje ona donosi društvu u cjelini.

U radu je potrebno:

- projektirati i izraditi automatiziranu šahovsku ploču s mogućnošću dvoosnog pomicanja elektromagneta za prihvat i otpuštanje šahovske figure
- integrirati konstrukcijske, pogonske, upravljačke i senzorske komponente u funkcionalnu cjelinu
- izraditi program za prepoznavanje šahovskih poteza protivničkog igrača te razviti program za automatizirani način igranja šaha
- istražiti tržišne mogućnosti komercijalizacije razvijenog sustava.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
7. ožujka 2024.	9. svibnja 2024.	13. – 17. svibnja 2024.

Zadatak zadao: Predsjednik Povjerenstva:

Prof. dr. sc. Željko Šitum Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS TEHNIČKE DOKUMENTACIJE

202404ASP1	Kućište
202404ASP2	Nosiljka
202404ASP3	Nosač elektromagneta

POPIS OZNAKA

Oznaka	Jedinica	Opis
V_{ref}	V	Referentni napon
I_{naz}	A	Nazivna struja
η	-	Faktor sigurnosti

SAŽETAK

U ovom diplomskom radu prezentirano je projektiranje i izrada automatizirane šahovske ploče koja autonomno igra protiv igrača. Danas sve više igrača igra šah koristeći računalo. Cilj ovakvog sustava je omogućiti prednosti igranja šaha protiv kompjutera na fizičkoj ploči. Uređaj se sastoji od mehanizma za dvoosno pomicanje elektromagneta, sustava za prepoznavanje šahovskih poteza i programa za upravljanje sustavom i odabiranje šahovskih poteza. Uređaj je upravljan Arduino razvojnom pločicom. U radu su također istraženi slični komercijalizirani sustavi.

Ključne riječi: šah, elektromagnet, automatizacija, Arduino

SUMMARY

The topic of this thesis is the designing and making of an automated chess board that autonomously plays against the player. Today more and more players play chess using a computer. The goal of this system is to provide the features of computer chess on a physical board. The device is made up of a mechanism that moves an electromagnet along two axis, a system for detecting the players moves and a program that controls the system and chooses its moves. The device is controlled by an Arduino development board. Furthermore, the chess market which is continuously growing is explored in the thesis.

Key words: chess, electromagnet, automation, Arduino

1. UVOD

U ovom radu prezentirat će se projektiranje i izrada šahovske ploče koja autonomno igra protiv igrača.

Kako bi ploča imitirala pravog igrača mora izvršavati tri glavne funkcije. Treba prepoznavati trenutno stanje ploče, u razumnom vremenu odabrati potez koji poštuje pravila šaha i odigrati taj potez. U sklopu rada prezentirat će se kako je izrađena ploča, kako međusobno funkcioniraju komponente i objasnit će se program koji upravlja šahovskom pločom. Sve komponente sustava detaljno će se opisati, navodeći kako funkcioniraju i kako interaktiraju s ostatkom sustava. Ključne komponente uređaja su mikrokontroler, elektromotori, remen, linearne vodilice, elektromagnet, reed prekidači, multiplekseri. Također, na kraju rada proučit će se slični komercijalizirani sustavi i šahovsko tržište općenito.

1.1. Povijest šaha

Šah ima bogatu povijest koja započinje u 6. stoljeću u Indiji s igrom „chaturanga“. Od nje su se do danas u šahu zadržala pravila da različite figure imaju drugačije mogućnosti kretanja i da se pobjeda ostvaruje osvajanjem jedne najbitnije figure. Iz Indije se šah proširio na Perziju gdje mu je promijenjeno ime u „shatranj“ i tada je krenula norma izvikivanja "Shāh!" kada je kralj napadnut i "Shāh Māt!" („Kralj je mrtav!“ na perzijskom jeziku) kada je igra gotova. U Europu je igra stigla u 9. stoljeću i do kraja tisućljeća se proširila po kontinentu. U Europi su figure promijenile oblike, no funkcije figura većinom su ostale iste. Najveće promjene su napravljene lovcu i kraljici u 15. stoljeću. Lovac je originalno mogao preskočiti figuru, ali se mogao pomaknuti samo za dva polja, a kraljica se mogla kretati samo dijagonalno i samo za jedno polje. Zbog tih pojačanja figura kralj se nalazio u većoj opasnosti, pa je uvedena i rohada kako bi se kralj mogao sakriti u jedan od kuteva i tako se obraniti od iznenadnih napada preko centra ploče. [1]

1.2. Osnovni šahovski principi

Svaka šahovska partija može se podijeliti u 3 dijela: otvaranje, središnjicu i završnicu.

Otvaranje je početni dio igre kojeg obilježuje prvi potez bijelog igrača zbog kojeg bijeli uvijek odlučuje u kojem smjeru će povesti igru. Glavni cilj bijelog u otvaranju je povećati svoju prednost tako da odvede protivnika u poziciju koju poznaje bolje od protivnika, dok crni u ovom stadiju pokušava preoteti tempo igre i neutralizirati prednost bijeloga. Dok je nekima otvaranje najfascinantniji dio igre zbog količine knjiga i teorije općenito koja je razvijena o njima, drugima ta ekstenzivna priprema igrača smeta jer umanjuje utjecaj kreativnost i snalažljivosti tijekom igranja. Također, u otvaranju su razvijene mnoge zamke na koje treba obratiti pozornost, jer na prvi pogled uvijek izgledaju kao previd igrača, a često mogu okončati igru u samo nekoliko poteza. Kontroliranje centra ploče, razvijanje figura na aktivna polja, osiguravanje kralja i stvaranje inicijative glavni su ciljevi igrača u otvaranju igre.

Središnjica počinje nakon što se većina figura razvila, što se generalno desi oko desetog poteza igre. U središnjici igrači žele biti fleksibilni kako bi brzo mogli reagirati na protivnikove greške. Bitno je imati pješačku strukturu koja je solidna i figure koje se međusobno brane. U središnjici dolaze do izražaja razne taktike i kombinacije s pomoću kojih igrač može forsirati povoljne razmjene. Najbitnije je stvoriti konkretan plan kako bi se moglo prijeći u završnicu s prednosti.

Završnica kreće kada je ostalo samo nekoliko figura na ploči. Tada se više igrač ne brine puno za sigurnost kralja, već kralj postaje aktivna figura kojoj je glavni cilj izvesti nekog od pijuna do kraja ploče. U završnici igrač ima limitirane opcije, pa je moguće kalkulirati jako puno poteza unaprijed. Za početnike najbolje je krenuti od učenja završnica kako bi im znanje šaha počivalo na dobrim temeljima.

1.3. Prednosti šaha

Šah je od svojih začetka do današnjih međunarodnih turnira služio kao izazov za ljudski intelekt. Složenost igre je kroz povijest fascinirala učenjake, filozofe i strategе. Sposobnost predviđanja posljedica, osmišljavanja planova više koraka unaprijed i prilagođavanje protivnikovim strategijama ključne su kognitivne sposobnosti koje šah zahtijeva i time ih razvija kod igrača. Zbog toga se smatra jednim od najboljih hobija za um i uvijek se promiče igranje i treniranje šaha u školama i ostalim obrazovnim ustanovama. Njegova popularnost dokaz je trajne privlačnosti intelektualnih izazova i nadmetanja.

Slika 1. Prednosti šaha [2]

1.4. Šahovski programi

Šah je oduvijek predstavljao platformu na kojoj se razvijaju nove tehnologije. Naročito za razvoj računarstva gdje su znanstvenici od začeća te znanosti pokušavali razviti program koji može parirati čovjeku u partiji šaha.

Prvi program probao je napraviti otac umjetne inteligencije Alan Turning 1951. godine, no nije uspio. Međutim manje od pola stoljeća kasnije, 1997. u drugom meču, dodatnim prilagođavanjem nakon poraza, IBM-ov program Deep Blue pobijedio je aktualnog šahovskog prvaka Garija Kasparova (Slika 2.). Ovaj događaj predstavlja prekretnicu u šahu i računarstvu općenito.

Od tada do danas su šahovski programi toliko napredovali da čovjek više ne može zamisliti da pobjedi kompjuter. Za usporedbu, prosječni amaterski šahovski igrač ima rejting od 1500. Magnus Carlsen već je dugo najbolji igrač šaha na svijetu i 2014. godine dosegao je najveći zabilježeni šahovski rejting od 2882. Najbolji kompjuteri danas imaju rejting veći od 3500. Mnogi su mislili da će ljudima šah prestati biti interesantan kada kompjuter počne igrati šah bolje od čovjeka, no to se nije desilo. Kompjuter je postao alat koji pomaže igračima da analiziraju partije, vježbaju taktike i uče otvaranja.

Slika 2. Gari Kasparov protiv Deep blue programa [3]

1.5. Arduino

Arduino je popularna platforma za razvoj ugrađenih sustava koja je zbog politike otvorenog izvornog koda postala ključni alat za veliki broj projekata s elektroničkim komponentama. Arduino je nastao 2003. na institutu „*Interaction Design Institute Ivrea*“ gdje je razvijen kao alat koji je studentima služio za realiziranje svojih ideja. Ima široku primjenu u različitim područjima kao što su automatizacija, robotika, edukacija i internet stvari. Arduino nudi raznovrsne hardverske razvojne pločice koje se koriste kako bi se senzori, aktuatori, motori i ostale elektroničke komponente upravljale i povezale u jednu cjelinu. Ključne karakteristike platforme su jednostavnost uporabe, pristupačna cijena i velika zajednica aktivnih korisnika koji pomažu jedni drugima. U ovom radu se radi jednostavnosti Arduino razvojna pločica ponekad naziva mikrokontroler iako je mikrokontroler samo jedan dio razvojne pločice.

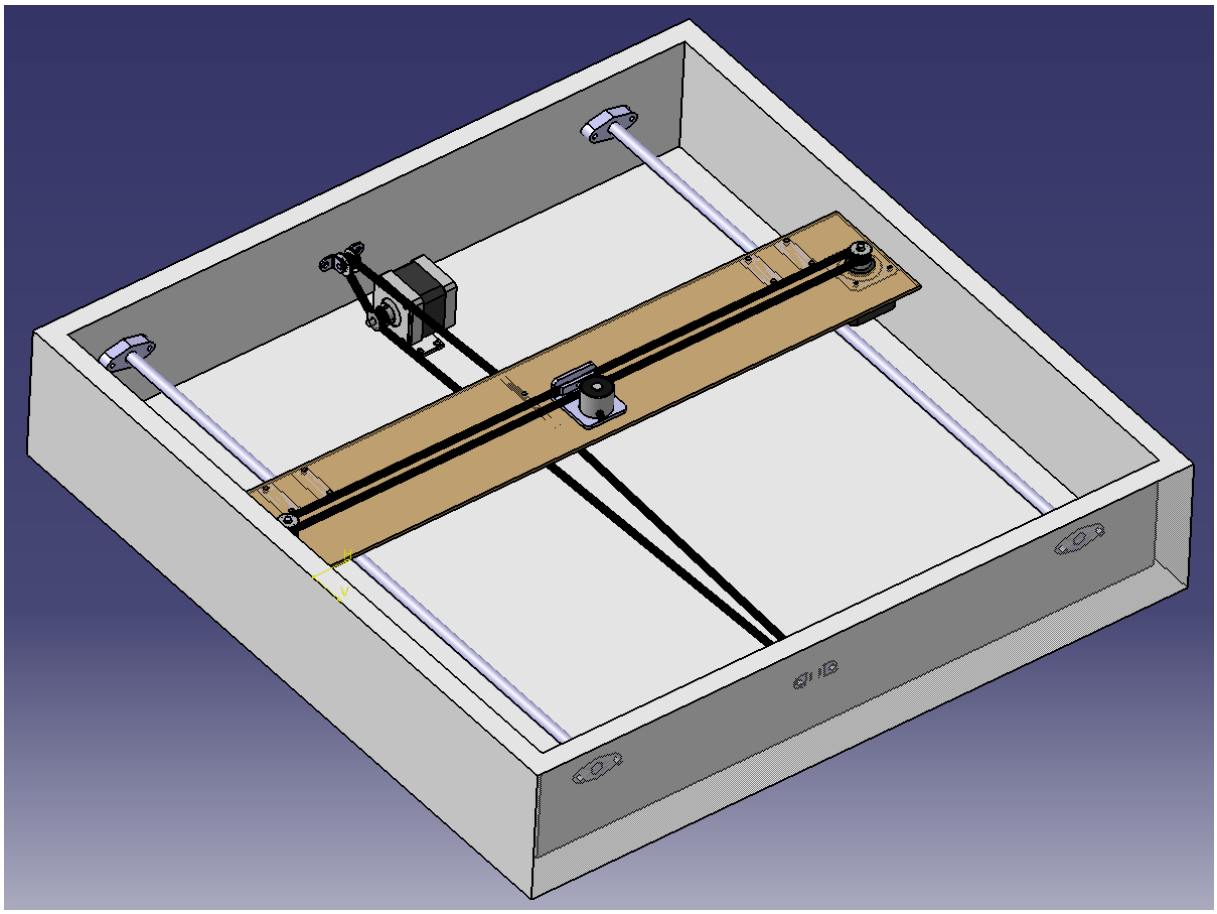
2. PROJEKTIRANJE SUSTAVA

Kutija u kojoj se nalazi mehanizam izrađena je od dasaka smreke debljine 18 mm koje osiguravaju stabilnost konstrukcije. Poklopac, odnosno sama šahovska ploča je od stakla kako bi se vidio mehanizam. Debljina stakla je 5 mm. Dimenzije sustava su 600 x 600 x 125 mm. Šahovska polja su kvadrati s dužinom stranica 5 cm. Kako bi konj mogao prolaziti između figura izabrane su malene figure s maksimalnim promjerom od 2,4 cm. Kako bi elektromagnet mogao pomicati figure izabrane su lagane figure od plastike i u svaku je stavljen permanentni magnet dimenzija Ø10 x 2 mm (Slika 3.).

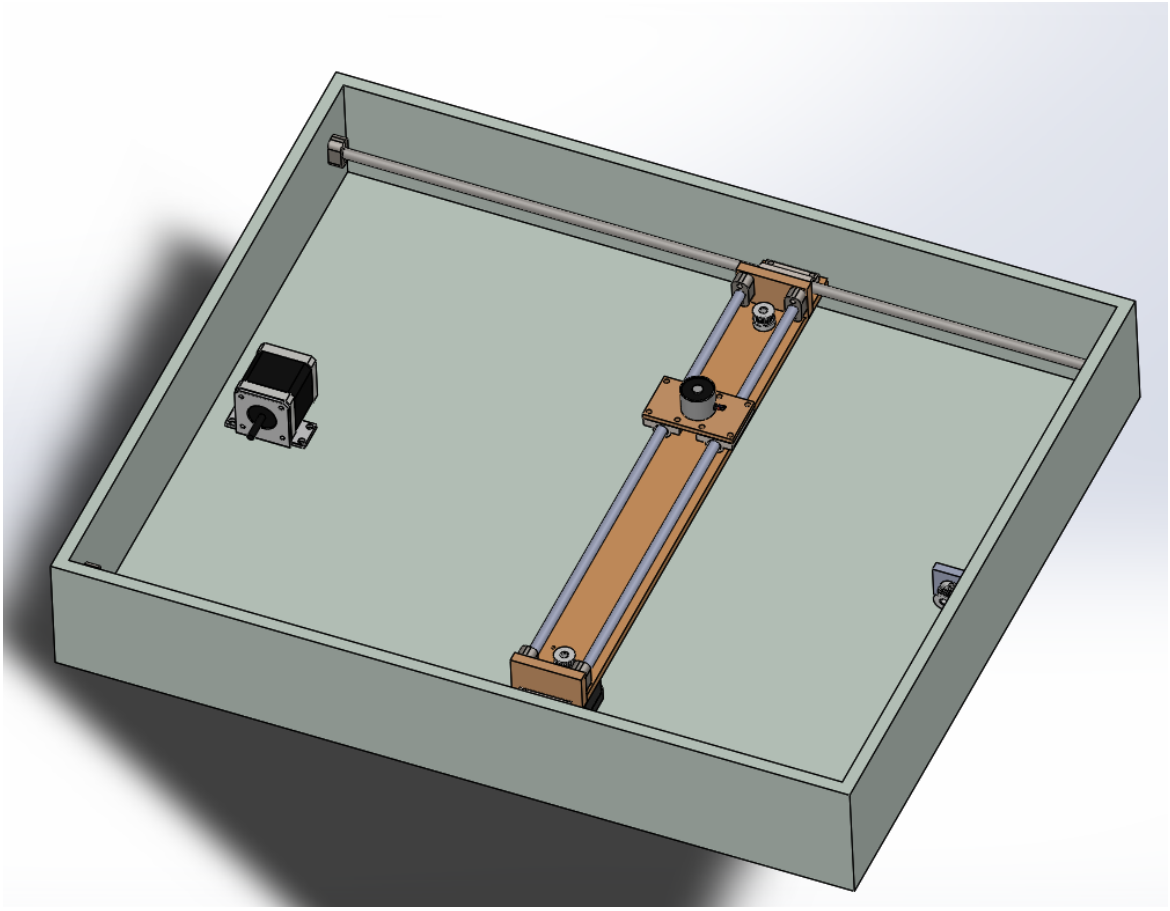
Slika 3. Šahovska figura s magnetom

2.1. Izrada modela

Kako bi se vizualizirao mehanizam za pomicanje elektromagneta izrađen je trodimenzionalni model u CATIA programskom paketu. Po okomitom smjeru od igrača jači elektromotor preko remenice pomiče nosiljku na kojoj se nalaze elektromagnet i drugi elektromotor. Minimalno trenje i ravno gibanje nosiljke osigurano je paralelno postavljenim linearnim vodilicama po kojima nosiljka klizi na linearnim kugličnim ležajevima. Manji elektromotor pričvršćen za nosiljku remenom pokreće plastični nosač na kojem se nalazi elektromagnet.



Slika 4. Model ploče konstruiran u programu CATIA

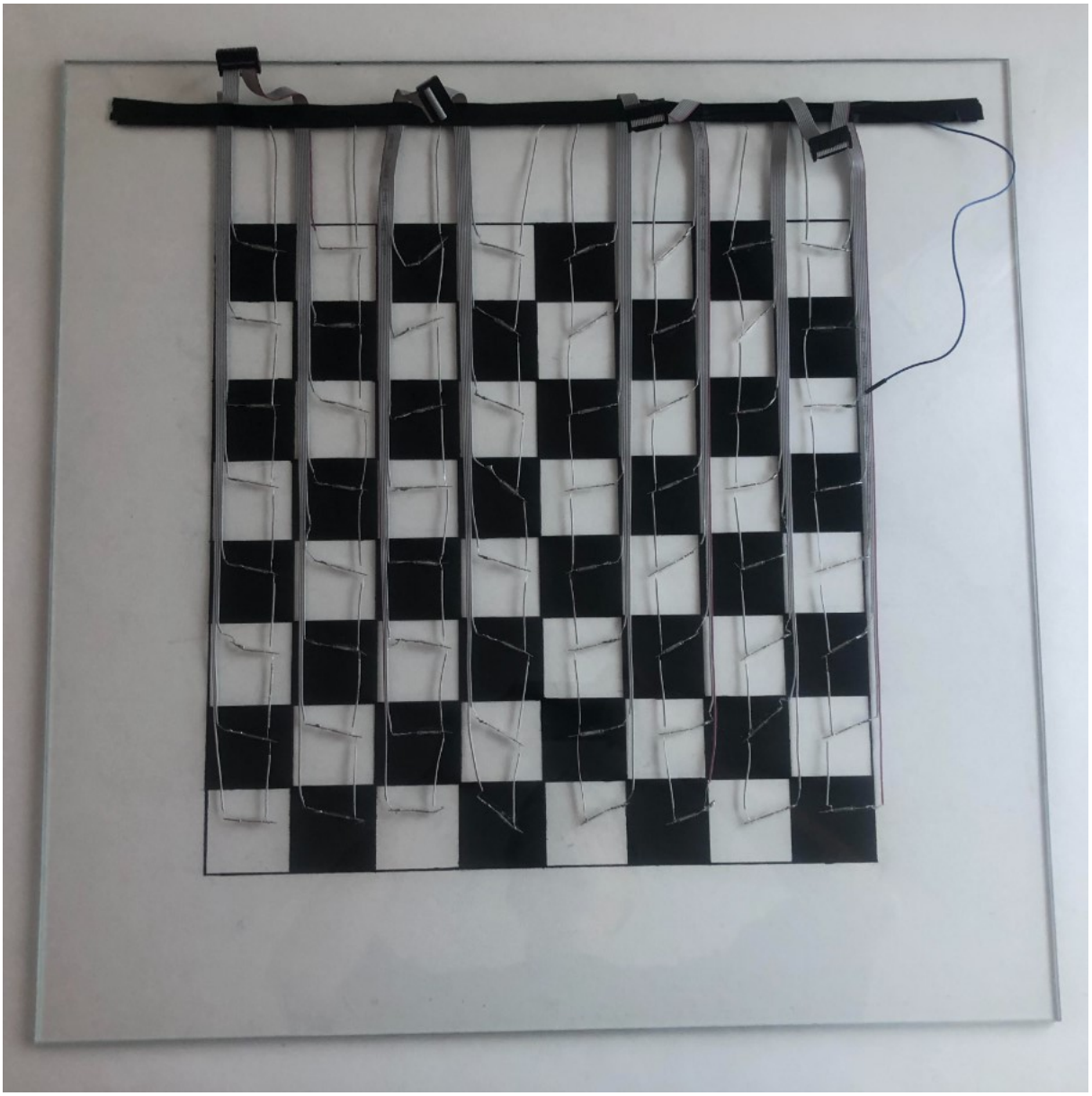


Slika 5. Rana verzija modela

Prva ideja bila je koristiti linearne vodilice za obje osi, no tako bi nosiljka više težila i ostalo bi premalo mjesta za vođenje po jednoj od osi (Slika 5.).

Slika 6. Izrađeni sustav

Na slici 6. prikazan je gotov uređaj. S obje strane ploče ostavljen je prostor za pojedene figure. Šahovska polja nacrtana su permanentnim markerom. Na staklo je povremeno potrebno nanijeti sredstvo za podmazivanje kako bi se smanjilo trenje između figura i stakla.

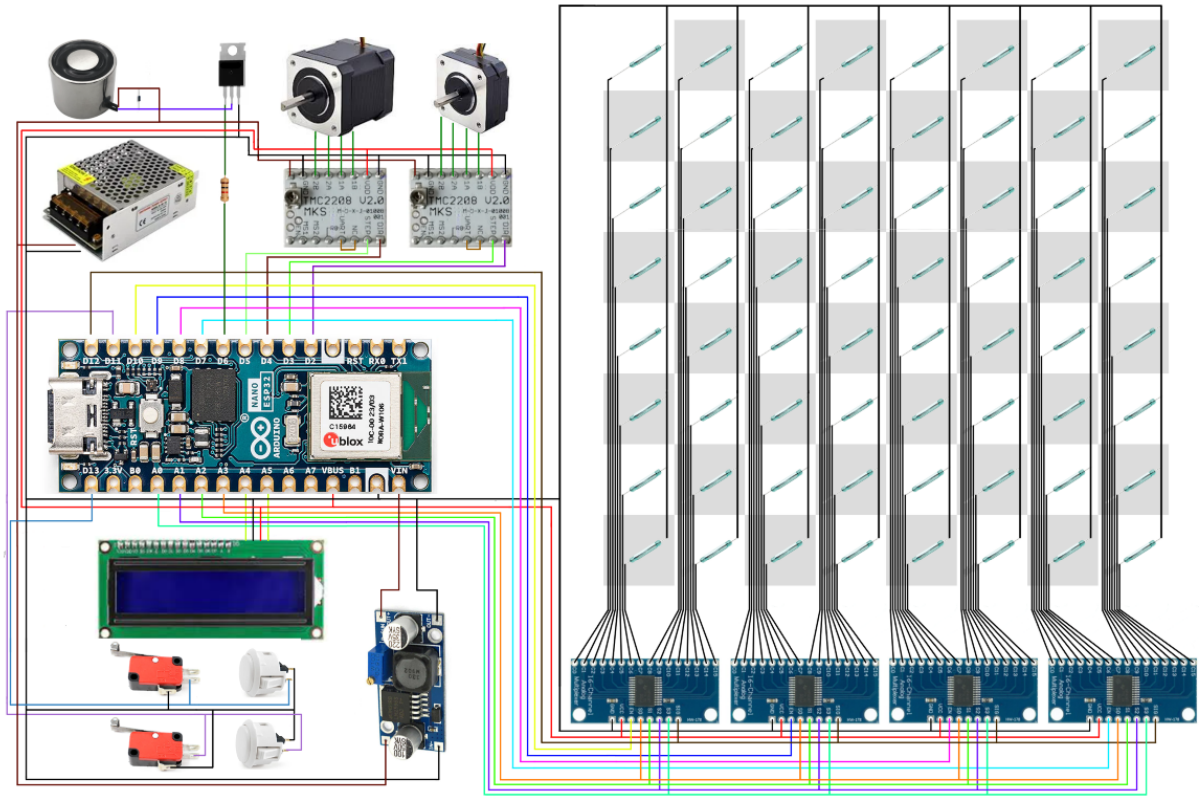


Slika 7. Donja strana šahovske ploče

Na slici 7. prikazana je 8 x 8 matrica reed prekidača postavljena s druge strane stakla. Reed prekidači naizmjenično detektiraju nalazi li se magnet u neposrednoj blizini. Svaki reed prekidač ima zasebnu žicu koja ga paralelno spaja s mikrokontrolerom preko multipleksera. Drugi krajevi prekidača spojeni su zajedno na uzemljenje. Reed prekidači zalijepljeni su za staklo snažnim ljepilom.

2.2. Shema elektroničkih komponenti

U programu Krita napravljena je shema spajanja elektroničkih komponenti (Slika 8.). Sve komponente bit će opisane u sljedećem poglavlju.

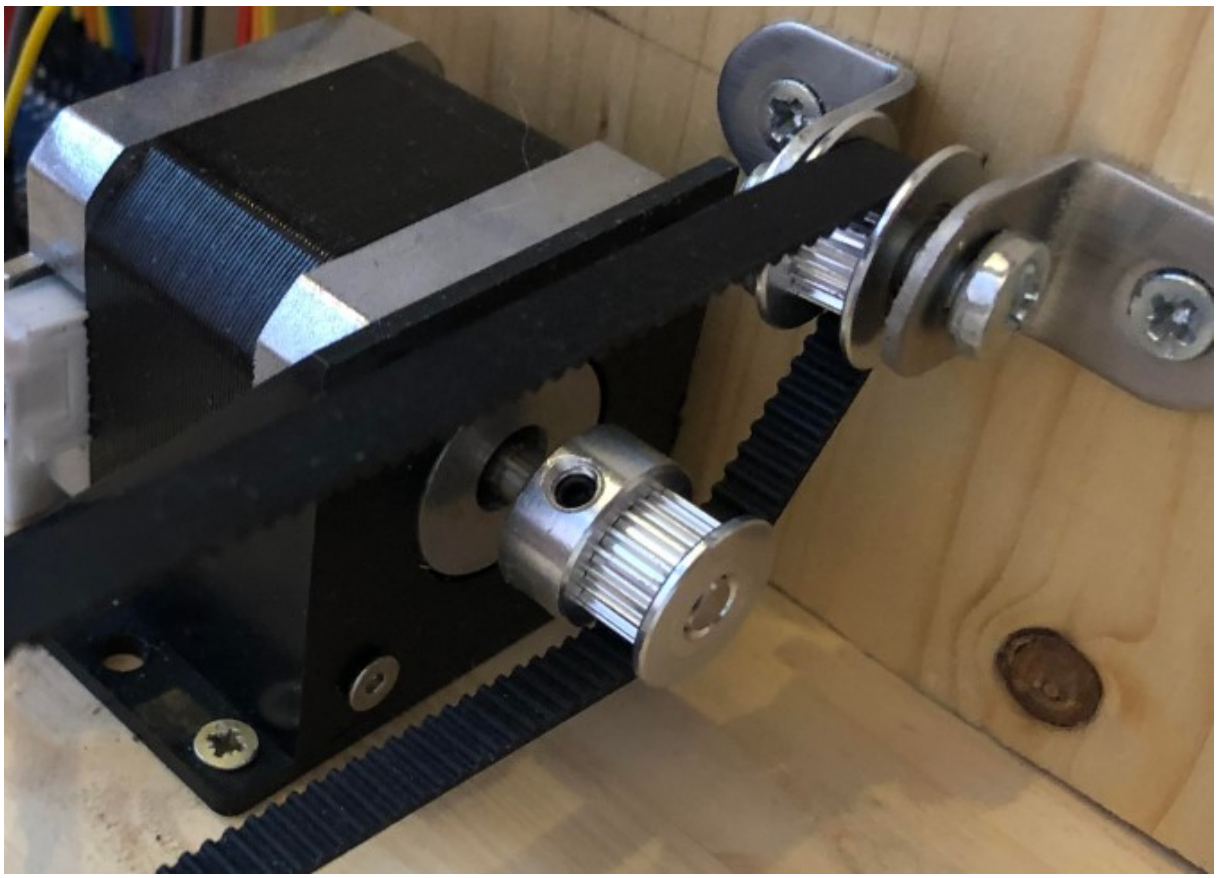


Slika 8. Shema elektronike

3. KOMPONENTE SUSTAVA

3.1. Mehaničke komponente

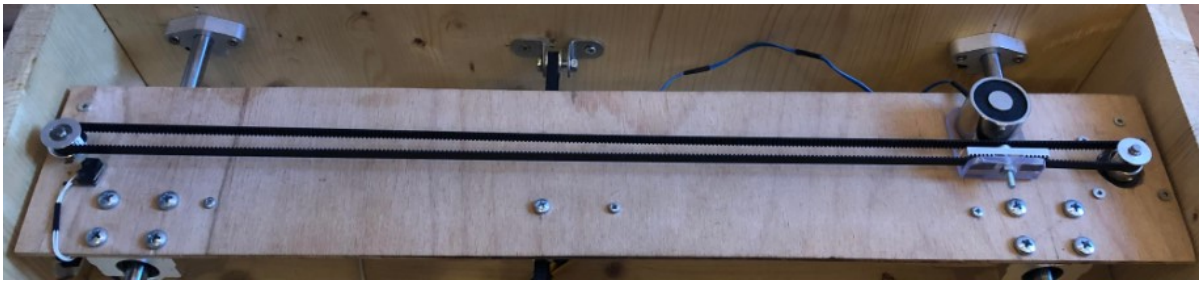
Za prijenos snage od motora na nosiljku korišten je remen GT2 širine 6 mm koji se vrti na GT2 remenicama s 20 zuba i provrtom od 5 mm kako bi bile kompatibilne s osovinom Nema 17 motora (Slika 9.). GT2 remeni imaju korak od 2 mm i zaobljeni profil zuba koji točno sjeda u utor remenice zbog čega nema pomicanja zuba remena u utoru prilikom mijenjanja smjera. Remenice su pričvršćene za kućište koristeći po dva kutna nosača i M5 vijak.



Slika 9. Elektromotor s remenom

Nosiljka je izrađena od šper ploče debljine 3 mm. Široka je 10 cm tako da se na jednom dijelu nosiljke nalaze M5 vijci koji osiguravaju nosiljku za ležajeve, a na drugom dijelu se giba elektromagnet. Elektromagnet je vijkom pričvršćen za plastični nosač. Remen prenosi silu na nosač preko aluminijske kopče koja steže remen vijkom.

Na jednom kraju nosiljke instaliran je elektromotor s remenicom, a na drugom kraju nalazi se M5 vijak koji služi kao osovina oko koje se vrti druga remenica. S donje strane nosiljka je ojačana daščicama kako se ne bi savijala zbog djelovanja sile koja drži remen u napetom stanju.



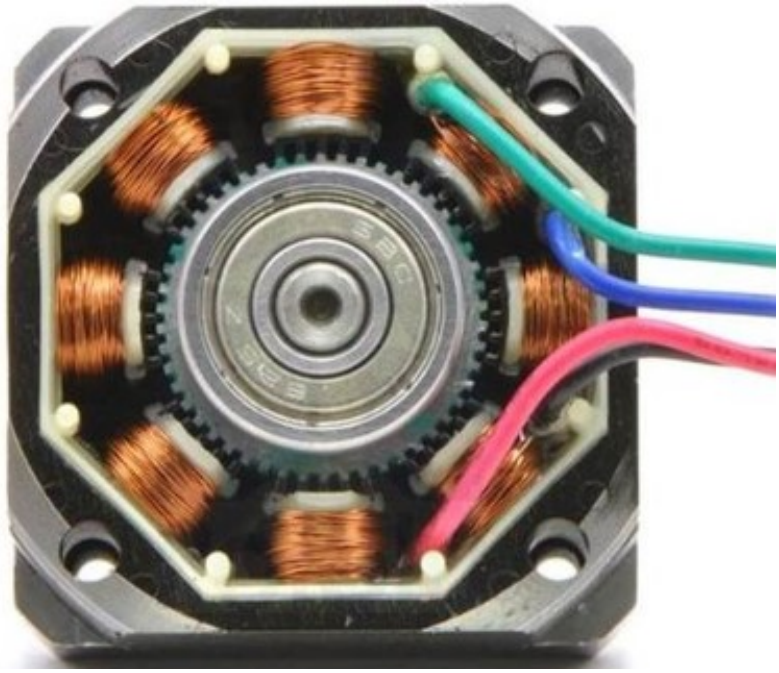
Slika 10. Nosiljka

Linearni ležajevi klize po čeličnim linearnim vodilicama promjera 10 mm. Vodilice za kućište osiguravaju aluminijski nosači vodilica s vijkom za pritezanje.

3.2. Elektroničke komponente

3.2.1. Elektromotori

Odabrani su Nema 17 koračni motori. Koračni motori postali su najpopularniji izbor za Arduino projekte zbog povoljnog omjera cijene i kvalitete. Široko su korišteni za pokretanje CNC strojeva, 3D printera, robota i sličnih automatiziranih uređaja. Oni rade pretvaranjem električnih impulsa u promjenjivo magnetsko polje. Magnetsko polje se stvara između kružno raspoređenih elektromagneta u statoru i permanentnog aksijalnog magneta u rotoru. Promjene u magnetskom polju nastaju aktiviranjem različitih elektromagneta raspoređenih u faze. Nema 17 koračni motori nemaju četkice, stoga je za njihovo upravljanje potreban upravljač motora. Upravljanje drajverom umjesto četkica smanjuje potrebu za održavanjem motora i omogućuje mu duži životni vijek.



Slika 11. Unutrašnjost Nema 17 koračnog motora [4]

Nema 17 označava standardiziranu veličinu i ima karakteristike: [5]

1. Kompaktna veličina: NEMA 17 motori relativno su kompaktni i lagani
2. Precizno pozicioniranje: Motor ima 50 zuba na rotoru. Broj pomaka po okretaju dobije se množenjem broja zuba s brojem mogućih orijentacija magnetskog polja. Motor ima dvije faze, stoga može napraviti 4 različite magnetske orijentacije i time je broj diskretnih pomaka po okretaju ovog motora 200. Nadalje preciznim upravljanjem svaki pomak može se dodatno podijeliti 256 puta čime maksimalna razlučivost iznosi 0,007° po koraku.
3. Kompatibilnost: NEMA 17 motori pridržavaju se standardiziranih dimenzija i obrazaca za montiranje, osiguravajući kompatibilnost s mnogim dodatcima kao što su nosači, spojnice, remenice itd.
4. Upravljanje otvorenom petljom: NEMA 17 motorima se može upravljati bez povratne veze. Ova jednostavnost može smanjiti složenost sustava i troškove.
5. Visoki zakretni moment pri malim brzinama: Ovi motori nude dobre performanse zakretnog momenta, posebno pri nižim brzinama. Ova karakteristika čini ih prikladnima za primjene koje zahtijevaju veliki okretni moment, kao što je pomicanje tereta. Također

zbog ove prednosti često nije potrebno primjenjivati reduktor, što dodatno smanjuje kompleksnost i cijenu sustava.

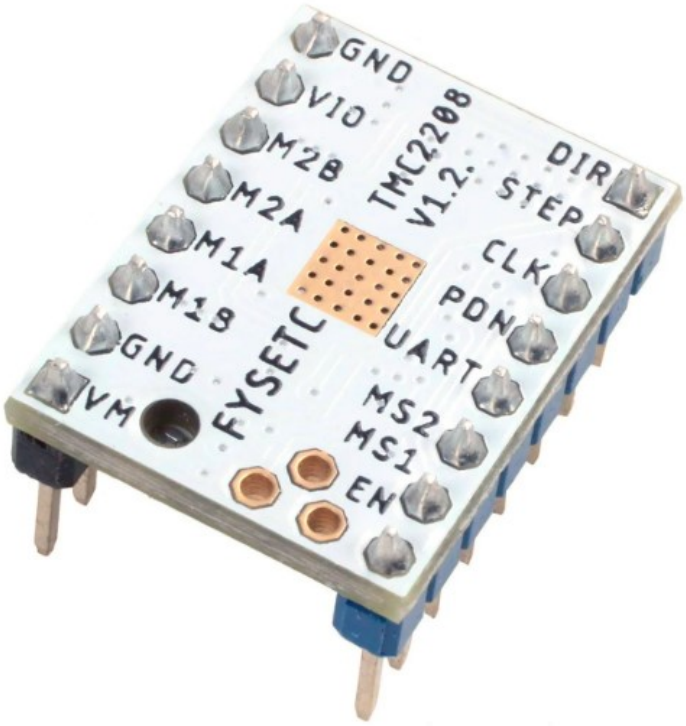
6. Dvosmjerna kontrola: Koračni motori mogu se okretati u oba smjera bez potrebe za dodatnim komponentama što je vrlo korisno kada je potrebno kretanje naprijed-nazad.
7. Moment držanja: I kada se koračni motor ne giba, može držati poziciju s određenim iznosom torzije zbog privlačnosti između permanentnog magneta u rotoru i aktiviranih elektromagneta u statoru. Ova karakteristika prevenira neželjene pomake između operacija.

Specifikacije izabranih elektromotora prikazane su u tablici 1.

Tablica 1. Specifikacije elektromotora [6][7]

	Jači elektromotor	Slabiji elektromotor
Model elektromotora	17CS04A	17HS4023
Kut koraka	1,8°	1,8°
Okretni moment	0,42 Nm	0,14 Nm
Nazivna struja	1,7 A	0,7 A
Broj faza	2	2
Masa motora	0,29 kg	0,14 kg
Promjer osovine	5 mm	5 mm
Gabaritne dimenzije	42 x 42 x 64 mm	42 x 42 x 55,5 mm

Za vođenje motora korišteni su driveri TMC2208. Jedna od najznačajnijih prednosti TMC2208 je njegova tehnologija *StealthChop*, koja omogućuje tihi i glatki rad motora pri malim brzinama. To ga čini idealnim za primjenu u šahovskoj ploči. Također razvija manje topline, može postići maksimalnu razlučivost motora od 51800 koraka po okretaju i može podnijeti široki raspon napona.



Slika 12. Upravljač motora TMC2208

GND pinovi se spajaju na uzemljenje. VIO konekcija je za napajanje logike upravljača, M2B i M2A služe za upravljanje jedne faze motora, a M1A i M1B za upravljanje drugom fazom. Na VM treba spojiti napajanje motora. DIR konekcija služi za definiranje smjera vrtnje, a STEP konekcija za upravljanje koracima. CLK konekcija se spaja kada se želi sinkronizirati sat drajvera s nekim drugim uređajem. PDN konekcija služi kako bi motor po potrebi mogao biti u modu slabog napajanja i UART konekcija može se koristiti kao alternativni način upravljanja. MS1 i MS2 konekcije služe za postavljanje konfiguracije mikro koraka. U slučaju da te konekcije nisu spojene motor će raditi 8 mikro koraka kao što je prikazano u tablici 3. EN konekciju treba spojiti na uzemljenje kako bi se upravljač aktivirao.

Tablica 2. Konfiguracije mikro koraka upravljača motora

MS1	MS2	Broj mikro koraka
0	0	8
1	0	2
0	1	4
1	1	16

Između uzemljenja i VM konekcije, odnosno paralelno s napajanjem motora treba spojiti električni kondenzator prikazan na slici 13. Električni kondenzator potrebno je spojiti kako bi se ublažili skokovi u naponu uzrokovani urušavanjem magnetskog polja u zavojnicama elektromotora i rezonancom između induktivnosti i kapaciteta strujnog kruga koja može uzrokovati oscilacije. Također, prevenira iznenadne padove u naponu prilikom uključenja motora i smanjuje buku filtrirajući visoke frekvencije. Izabran je kondenzator kapaciteta od 47 μF koji može raditi na naponu do 16 V.

Slika 13. Električni kondenzator

Dovod struje elektromotoru postavljen je na 80% nazivne struje motora kako bi se preveniralo pregrijavanje i osigurao pravilni rad motora. Dovod struje podešava se pomoću vijka koji služi kao potenciometar. Odvrtanjem vijka povećava se pad napona na otporniku, a time i struja koja se dovodi motoru. Maksimalna struja koju ovaj upravljač motora može pružati elektromotoru iznosi 2 A.

Referentni napon koji treba narinuti elektromotoru izračunat je sljedećom formulom (1) [8]

$$V_{ref} = I_{naz} \times 1.41 \times \eta$$

$$V_{ref1} = 1{,}7 \times 1.41 \times 0{,}8 = 1{,}91\ V \quad \textbf{(1)}$$

$$V_{ref2} = 0{,}7 \times 1.41 \times 0{,}8 = 0{,}79\ V$$

Za prvi motor potrebno je napon postaviti na 1,91 V, a za slabiji na 0,79 V.

3.2.2. Elektromagnet

Princip rada elektromagneta poprilično je jednostavan, kada je potrebno pomaknuti figuru, struja se provede kroz zavojnicu namotanu oko metala unutar elektromotora. Protok struje u zavojnici stvara magnetno polje. Prvo je isproban elektromagnet privlačne sile od 50 N, no ta sila nije bila dovoljna, pa je korišten elektromagnet privlačne sile od 100 N. Dimenzije elektromagneta su Ø 30 x 22 mm. Ima snagu od 3 W i troši struju od 0,25 A.

Slika 14. Elektromagnet privlačne sile 100 N

Arduino može jednoj konekciji dati maksimalno 40 mA, stoga je potrebno elektromagnet spojiti direktno na napajanje. Kako bi se upravljalo elektromagnetom, između uzemljenja i elektromagneta stavljen je mosfet tranzistor IRF3205 (Slika 15.)

Slika 15. Mosfet tranzistor IRF3205

Mosfet tranzistori osiguravaju minimalni pad napona, brzo mijenjaju stanje i pošto se njima upravlja naponom nije potreban stalni dovod struje. Minimalni napon koji je potrebno dovesti tranzistoru kako bi se aktivirao iznosi 2 V. Izmjereni pad napona na korištenom tranzistoru iznosi 0,11 V.

Elektromagnet pohranjuje energiju u magnetnom polju kojeg stvara i naglim prestankom dovoda struje to magnetno polje se uruši inducirajući napon koji može biti višestruko veći od originalnog napona. Takav skok u naponu može oštetiti ostale komponente, pa je u suprotnom smjeru u odnosu na normalan tok struje spojena dioda slobodnog hoda (anoda na negativnu stranu, katoda na pozitivnu stranu). Ona omogućuje da preostala struja u zavojnici disipira putujući u krug između elektromagneta i diode.

Slika 16. Dioda FR607

3.2.3. Napajanje

Izabrano je napajanje od 12 V kako bi elektromagnet i upravljači motora pravilno funkcionirali. Nazivna struja jačeg motora je 1,7 A, a slabijeg motora iznosi 0,7 A, Arduino Nano ESP32 povlači oko 0,5 A i elektromagnetu je potrebno 0,25 A. Ukupno najveće moguće opterećenje iznosi 2,7 A. Odabrano je napajanje od 3,2 A. Kućište napajanja napravljeno je od aluminija i u slučaju kratkog spoja može biti opasan po život, pa je jako bitno koristiti kabel sa žicom za uzemljenje kućišta.

Slika 17. DC napajanje od 12 V i 3,2 A

Arduino mikrokontroler radi na 5 V i na njemu se nalazi regulator napona koji može raditi s ulazom od 12 V, no postoji opasnost od pregrijavanja. Kako bi se to izbjeglo korišten je LM2596 step-down modul s rasponom ulaza od 3-40 V i izlaza od 1.5-35 V (Slika 18.) .

Slika 18. Step-down modul

Izlaz je podešen na 6 V kako bi regulator napona na razvojnoj pločici radio učinkovito i pružao stabilnih 5 V. Izlazni napon regulira se malim vijkom zlatne boje vidljivim na slici. Modul ima izvrsnu efikasnost od 92%, stoga se minimalno struje pretvara u toplinu.

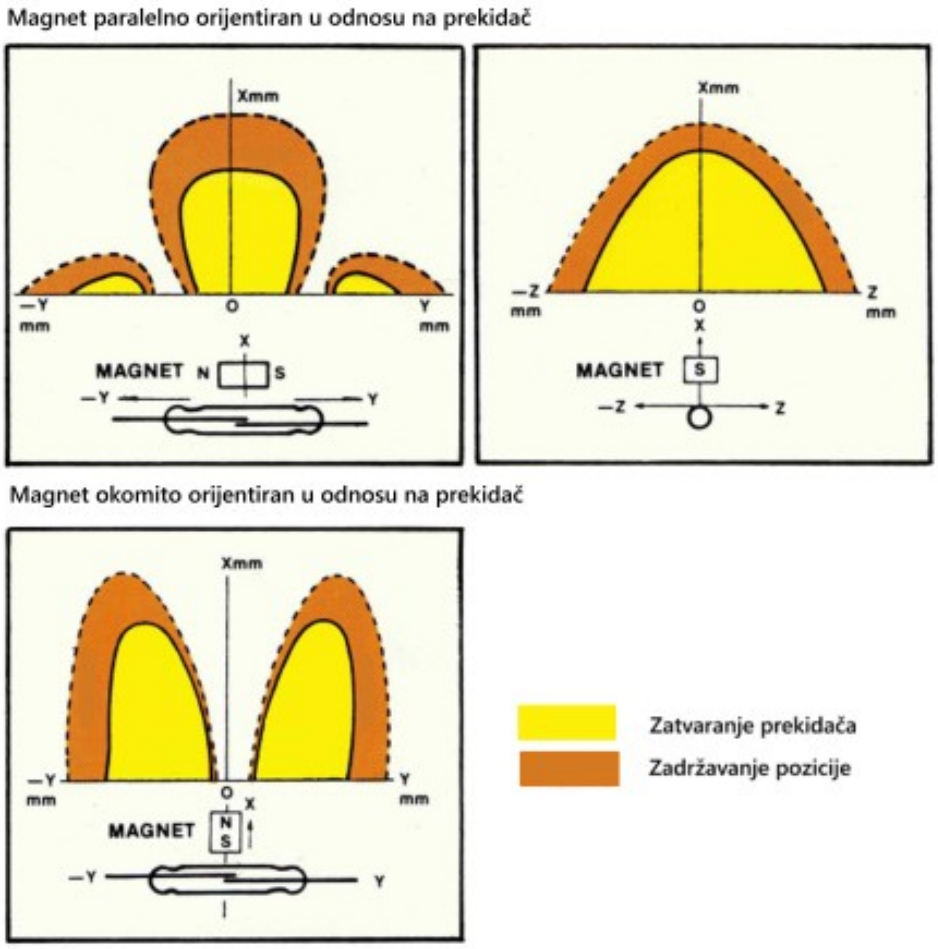
3.2.4. Komponente za prepoznavanje šahovskih poteza

Startna pozicija u šahu je uvijek ista, jedina varijabla je boja figura koja govori tko će igrati prvi. Kako bi se pojednostavilo programiranje igrač uvijek igra bijelim figurama, stoga ploča odmah nakon kalibracije čeka na potez igrača. Potezi igrača detektiraju se koristeći 64 reed prekidača, jedan prekidač za svako šahovsko polje. Reed prekidač se sastoji od dvije odvojene metalne pločice u staklenom kućištu ispunjenim inertnim plinom. Kada se magnetno polje približi prekidaču, metalne pločice uspostave kontakt i krenu provoditi struju signalizirajući mikrokontroleru da se na polju iznad nalazi figura.

Slika 19. Reed prekidač

Jednostavno rješenje za detekciju stanja ploče bilo bi snimanje ploče nakon pritiska gumba igrača, odnosno na kraju poteza. Međutim takva izvedba nije moguća s pomoću reed prekidača jer oni mogu samo detektirati postoji li figura na polju iznad i ne može znati je li figura crna ili bijela. Ovo ograničenje predstavlja problem u situaciji kada igrač pojede figuru, a imao je i izbor da pojede neku drugu figuru. U toj situaciji ploča ne bi mogla znati koja je figura pojedena. Stoga ploča treba konstantno provjeravati da li se promijenilo stanje na ploči kako bi se detektirao trenutak kada igrač podigne figuru koju želi pojesti i zamjeni je svojom. Može se pretpostaviti da je u tom slučaju reed prekidač otvoren na otprilike pola sekunde. Dakle kako bi ova izvedba funkcionirala mikrokontroler treba svake sekunde dva puta očitati stanje svih 64 prekidača.

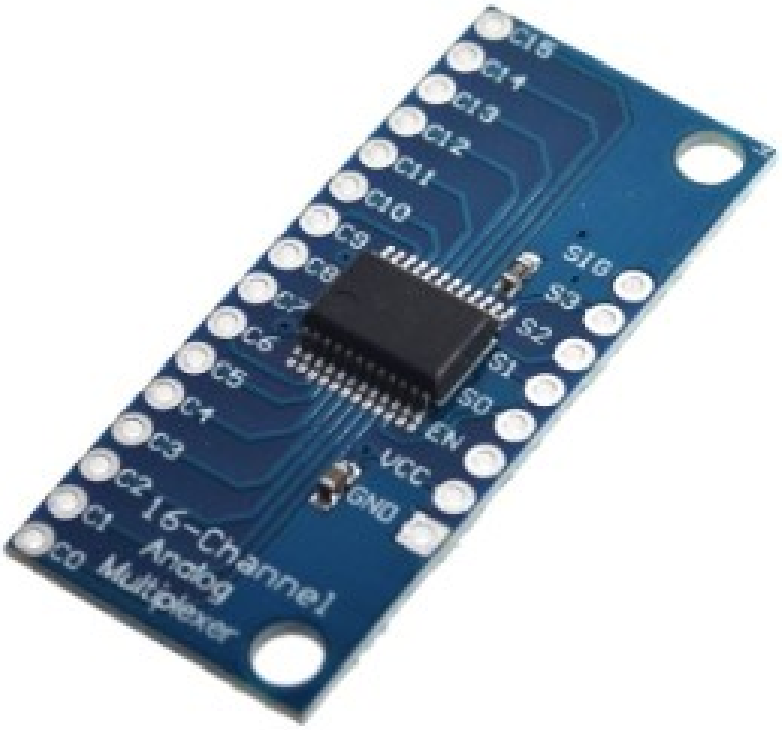
Na slici 20. prikazano je otvaranje i zatvaranje reed prekidača u ovisnosti o orijentaciji magneta.



Slika 20. Otvaranje i zatvaranje reed prekidača [9]

Kako bi elektromagnet mogao pomicati figure magneti u njima moraju biti okomito orijentirani u odnosu na prekidače. Ovo predstavlja problem jer reed prekidači bolje detektiraju paralelno orijentirane magnete. U detekciji okomito orijentiranih magneta postoji mrtvi prostor detekcije koji se nalazi odmah iznad prekidača, stoga se prekidači trebaju postaviti s odmakom od centra šahovskih polja. U teoriji ovo rješava problem, no zbog varijacije u poljima detekcije prekidača i netočnosti pozicioniranja uvijek postoji šansa da prekidač neće funkcionirati.

Mikrokontroler treba biti paralelno povezan sa svakim prekidačem kako bi očitavao da li provodi struju, no nema dovoljno pinova da se spoji 64 prekidača. Kako bi se to postiglo korišteno je četiri multipleksera na kojima svaki povezani prekidač ima svoju binarnu adresu. Na primjer kada je stanje konekcija 0101 aktivirat će se kanal 5, odnosno snimat će se stanje šestog prekidača.



Slika 21. Multiplekser CD74HC4067

Multiplekseri imaju 8 konekcija kojima se spajaju na Arduino. GND je uzemljenje, VCC treba biti spojen na napon između 2-6 V. EN konekcija u aktivnom stanju deaktivira multiplekser, aktivirajući ga kada se isključi dovod struje. SIG konekcija provodi analogni output, a kanali S0, S1, S2, S3 provode binarnu adresu kanala koji treba biti aktivan. Svi multiplekseri dijele sve konekcije osim EN konekcije koja treba biti posebno povezana. Tako se smanjuje broj potrebnih konekcija za detekciju stanja ploče sa 64 na 9.

3.2.5. Ostale elektroničke komponente

Ekran se koristi kako bi se prikazalo preostalo vrijeme igrača i kako bi se riješili problemi nastali tijekom igre npr. ilegalan potez. Ekranom program može i igraču poželjeti sreću na početku ili mu čestitati na dobro odigranoj partiji na kraju igre.

Slika 22. Ekran LCD1602

Kako bi se smanjio broj potrebnih konekcija ekran je spojen koristeći I2C konverter.

I2C (eng. *Inter-Integrated Circuit*) široko je korišten serijski komunikacijski protokol koji je razvio *Philips Semiconductor* ranih 1980-ih. Dizajniran je za sporu, ali učinkovitu komunikaciju između integriranih sklopova. Svaki uređaj ima jedinstvenu adresu na sabirnici, što omogućuje spajanje više uređaja na istu sabirnicu. Standardne I2C adrese duge su 7 bita, što omogućuje do 128 različitih adresa. Najveća prednost I2C protokola je mogućnost upravljanja uređajima koristeći samo dvije žice. Prva služi za prijenos podataka (SDA - *Serial Data Line*), a druga za sinkronizaciju uređaja (SCL - *Serial Clock Line*). U I2C sabirnici jedan uređaj obično djeluje kao glavni, započinjući komunikaciju s ostalim podređenim uređajima. Glavni uređaj kontrolira sabirnicu, pokreće prijenos podataka i sinkronizira uređaje u odnosu na sebe. Podređeni uređaji odgovaraju na naredbe glavnog uređaja s mogućnošću slanja povratnih informacija. I2C protokol je vrlo popularan zbog svoje jednostavnosti i velikog broja različitih uređaja koji su razvijeni kako bi bili kompatibilni s ovom tehnologijom. Na početku je tehnologija mogla funkcionirati samo na 5 V, no većina je uređaja danas prilagođena da rade u rasponu od 3,3V do 5V, a neki moderni uređaji mogu funkcionirati i s manjim naponom. Prijamni uređaj po I2C protokolu potvrđuje primanje svakog prenesenog bajta. Ovakav mehanizam potvrde primitka podataka osigurava pouzdanu komunikaciju. [10]

Slika 23. I2C adapter

Prilikom isključenja ploča ne sprema zadnju lokaciju elektromagneta, stoga se pri svakom uključenju uređaj treba kalibrirati.

Uređaj se kalibrira tako da se elektromotori pokreću sve dok se ne aktiviraju granični prekidači. Kada se granični prekidači aktiviraju pozicija magneta postaje poznata mikrokontroleru.

Slika 24. Granični prekidač

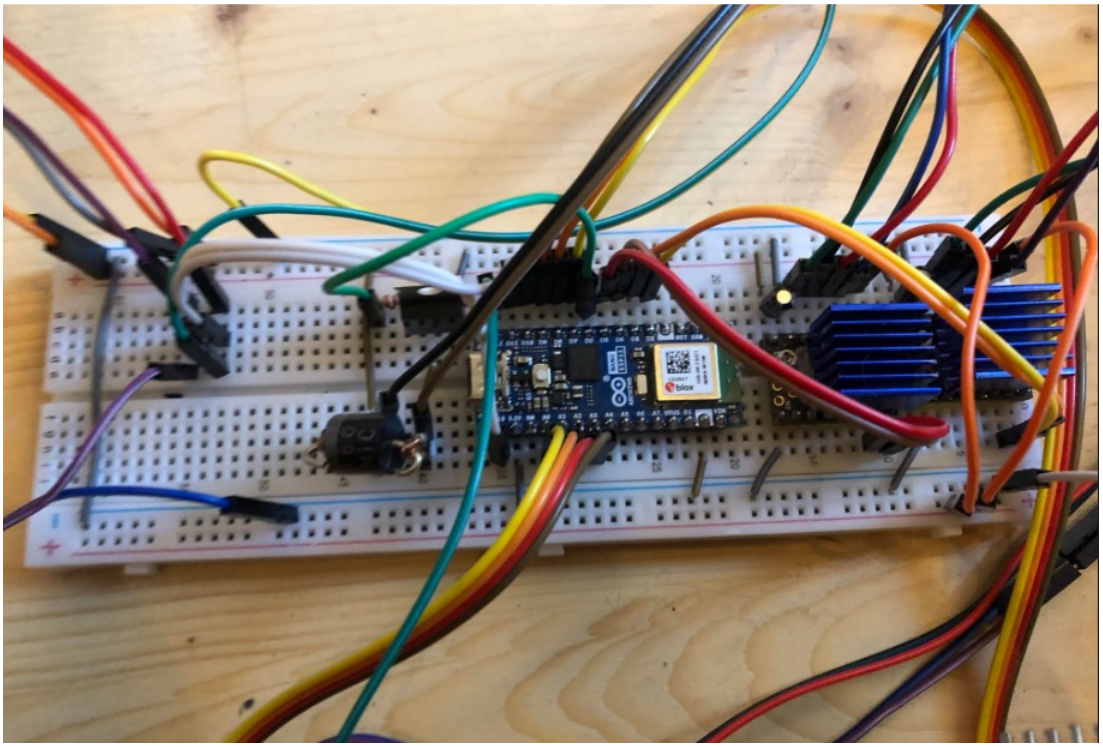
Granični prekidač spojen je u nominalno otvorenoj poziciji što znači da se strujni krug uspostavlja kada je aktiviran.

Kako bi igrač mogao interaktirati s pločom, montirana su dva gumba s desne strane ploče. Pošto su iskorišteni svi digitalni pinovi Arduino pločice, gumbovi i granični prekidači spojeni su na isti pin, jer nikada nisu potrebni istovremeno. Granični prekidači služe samo pri paljenju za kalibraciju i nakon toga se vrate u otvorenu poziciju i ne smetaju radu gumbova.

Slika 25. Arcade gumb

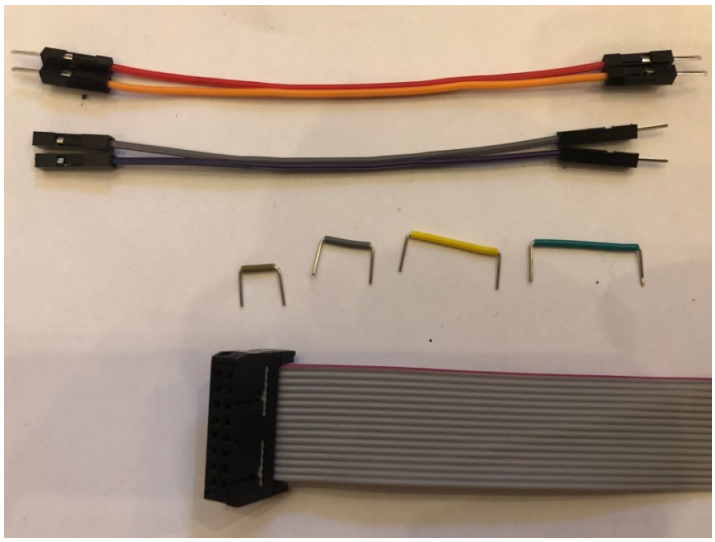
Konekcije za Arduino, upravljače motora, elektromagnete, granične prekidače, gumbe, diode, LCD zaslon i mosfet tranzistor stavljeni su na eksperimentalnu pločicu dimenzija 5.5 x 17 x 1 cm s 830 rupica.

Eksperimentalne pločice (eng. *Breadboard*) su jako praktičan alat za spajanje Arduino pločice s ostalim komponentama. Sastoji se od interno povezane mreže rupa. Na vrhu i dnu nalaze se dva reda povezanih rupica, a u sredini dva stupca. Ovakav alat omogućuje brzu izradu, testiranje i mijenjanje elektroničkih sklopova bez potrebe za lemljenjem.



Slika 26. Eksperimentalna pločica s komponentama

U sklopu projekta korišteno je tri vrste konekcija. Muško-muški i muško-ženski *DuPont* kablovi za povezivanje komponenti s eksperimentalnom pločicom, premosne žice za povezivanje na eksperimentalnoj pločici i ravni vrpčasti kablovi sa ženskim IDC konektorom za povezivanje reed prekidača.



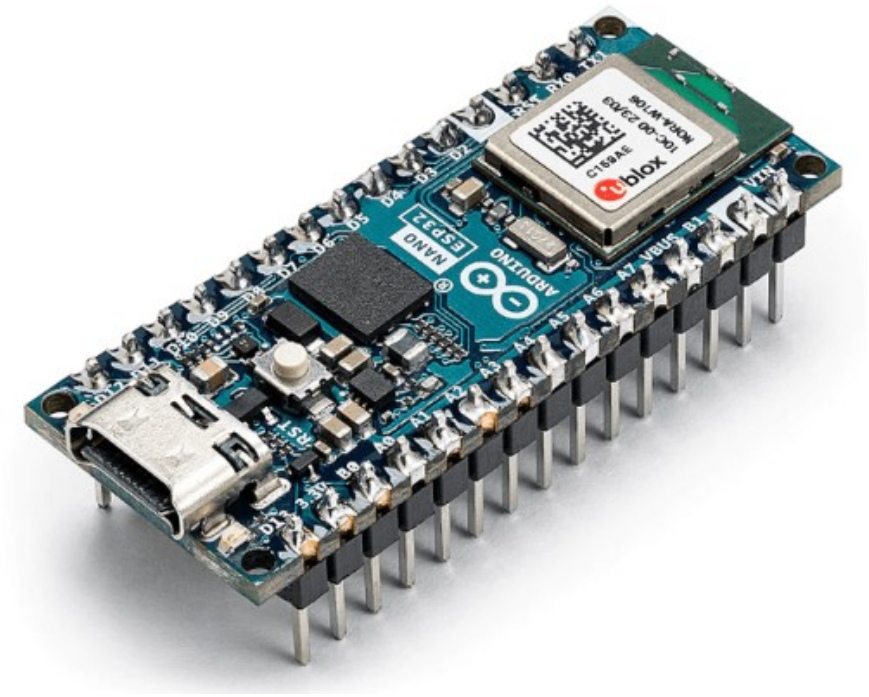
Slika 27. Korišteni konektori

4. UPRAVLJANJE SUSTAVOM

Program za upravljanje sustavom preuzet je od sličnih projekata. Potrebno je dodatno prilagoditi program. [11][12][13]

4.1. Arduino Nano ESP32

Za potrebe ovog projekta odabrana je Arduino Nano razvojna pločica s ugrađenim ESP32 mikrokontrolerom (Slika 28.).



Slika 28. Arduino Nano ESP32 [14]

ESP32 je snažan mikrokontroler kineske kompanije Espressif Systems s izvrsnim omjerom cijene i kvalitete. Ima dvojezgreni mikroprocesor koji ima vrlo impresivne specifikacije prikazane u tablici 2. Arduino Nano ESP32 kombinira snagu u-blox NORA-W106 procesora s jednostavnošću Arduino platforme.

Tablica 2. Specifikacije razvojne pločice Arduino Nano ESP32 [14]

Mikrokontroler	u-blox NORA-W106
Radni napon	5 V
Preporučeni ulazni napon	6 – 21 V
Broj digitalnih I/O konektora	14
Broj analognih I konektora	8
Struja po konektoru	40 mA
Flash memorija	16 MB
SRAM	512 kB
ROM	384 kB
Frekvencija takta	do 240 MHz
Gabaritne dimenzije	45 mm x 18 mm
Priključak	USB - C

Flash memorija trajna je memorija koja ne gubi memoriju kada nema napajanja. Ona funkcionira tako da zapisuje podatke po memorijskim blokovima na razini bajtova. Memorijski blokovi sastoje se od ćelija koje koriste tranzistore s pomičnim vratima za pohranjivanje i dohvaćanje podataka. U ćelijama se pokretnim vratima zarobe elektroni. Kada treba pročitati podatke očitava se naboj na vratima. Za brisanje podataka ćelije se isprazne i nakon toga moguće ih je ponovno koristiti.

SRAM (eng. *Static random access memory*) vrsta je statičke memorije koja gubi podatke kada nestane struje. Brzina pohranjivanja i čitanja je puno brža u odnosu na flash memoriju. Obično se koristi za pohranjivanje *cache* memorije sustava, odnosno memorije koja se često koristiti, kao imena varijabli. Skuplja je za proizvesti i koristi više struje.

ROM (eng. *Read-only memory*) je također vrsta statičke memorije. Na njoj se nalaze pretprogramirani podaci koje proizvođač zapisuje tijekom proizvodnje i nužni su za funkcioniranje uređaja.

Frekvencija takta je zajedno s brojem jezgri glavni pokazatelj brzine procesora. Govori o broju operacija koje procesor može napraviti u jednoj sekundi. Jedna operacija, odnosno jedan herc, predstavlja prebacivanje stanja jednog tranzistora na čipu.

Programiranje Arduina odvija se putem integriranog razvojnog okruženja Arduino IDE (integrated development environment). Program Arduino IDE je napisan u Java programskom jeziku, no za programiranje Arduina putem njega koristi se modificirani C++ programski jezik. Program napisan u okruženju Arduino IDE naziva se Sketch. Svaki program ima dvije osnovne funkcije. Setup() funkcija se pokreće jednom prilikom uključenja. Ona se koristi kako bi se definirale input i output konekcije i kako bi se inicirale varijable i ostale funkcije koje se trebaju izvršiti samo jedanput. Druga funkcija je Loop() koja se pokreće nakon završetka Setup() funkcije i ona se ponavlja sve do isključenja ploče.

Prilikom programiranja u Arduino IDE programskom okruženju koriste se 'knjižnice' koda koje predstavljaju unaprijed napisani kod koji pruža dodatne funkcionalnosti potrebne za funkcioniranje nekih komponenti, komunikacijskih protokola itd. Zbog dugovječnosti Arduino platforme i velikog broja korisnika postoji ogromna količina gotovog koda koji uvelike olakšava programiranje. Program Arduino IDE uključuje *Library Manager* koji pojednostavljuje pronalazak i upravljanje knjižnicama.

U ovom radu koristi se ekran koji zahtijeva implementaciju „*LiquidCrystal.h*“ knjižnice koda i on je povezan I2C adapterom koji zahtijeva implementaciju *„Wire.h“* koda.

4.2. Šahovski program

Korišten je šahovski program Micro-Max kojeg je napravio nizozemski programer Harm Geert Muller. Micro-Max je minimalistički šahovski program napisan u samo 133 linije koda. Zasniva se na minimax i alpha-beta pruning algoritmima. [15]

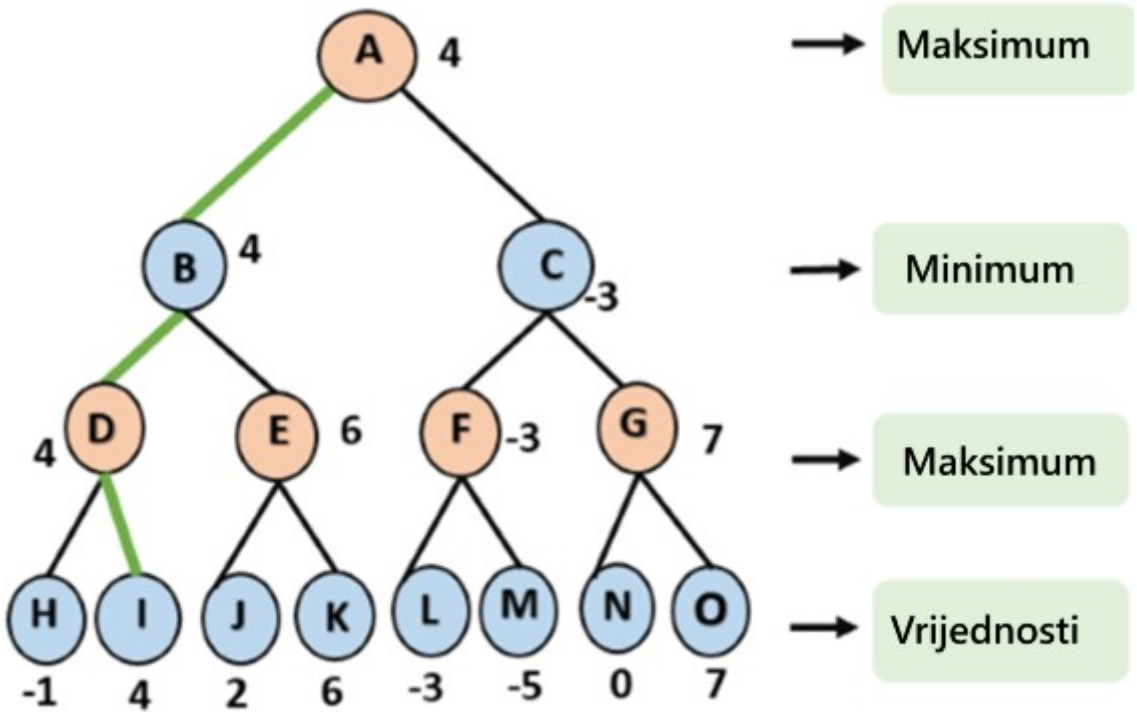
4.2.1. Minimax algoritam

Minimax je rekurzivan algoritam razvijen za donošenje odluka u igrama nulte sume, odnosno u igrama u kojima prednost jednog igrača znači gubitak za drugoga. Algoritam donosi odluke razmišljajući i za protivnika pod pretpostavkom da će i on odigrati najbolji potez generiran tim algoritmom. Drugim riječima algoritam se ponaša kao da igra sam protiv sebe. Krećući od trenutne pozicije generiraju se svi potezi i odgovori na te poteze, do postavljene dubine ovisno

o jačini procesora i vremenu za razmišljanje. Nakon generiranja pozicija algoritam ih kreće rekurzivno evaluirati prema predefiniranim faktorima za određivanje prednosti i slabosti u pozicijama. U šahu faktori koji se većinom uzimaju u obzir su broj i kvaliteta figura svakog igrača, sigurnost kralja, mogućnost manevriranja, mogućnost promocije pijuna itd.

Uvijek anticipirajući optimalnu igru, kompjuter odabire linije poteza minimalne evaluacije za sebe na potezu protivnika i maksimalne na svojim potezima. Pozicije koje kompjuter generira predstavljaju se stablom igre (Slika 29.) gdje je trenutna pozicija deblo stabla, a svaka podređena predstavlja jedan od mogućih poteza.

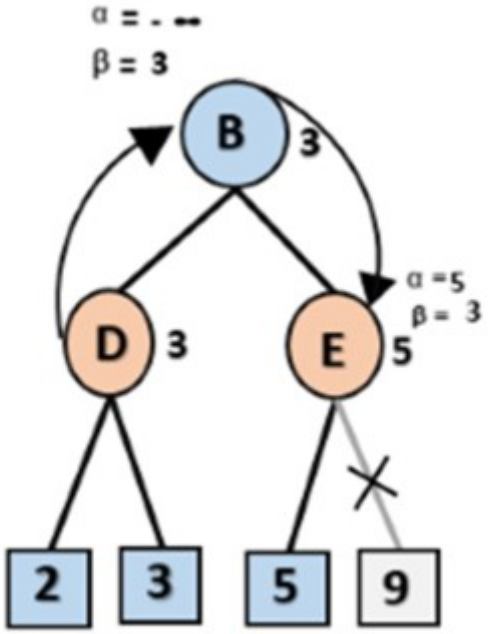
Glavni nedostatak minimax algoritma je eksponencijalni rast stabla igre s brojem poteza koji se uzimaju u obzir. Da bi se smanjila kompleksnost stabla često se primjenjuju optimizacije. Ovdje će se primijeniti algoritam alfa-beta rezidba kako bi se smanjio broj pozicija koje se trebaju evaluirati.



Slika 29. Primjer minimax algoritma [16]

U primjeru prikazano je kako minimax algoritam prolazi kroz stablo igre naizmjenično izabirući maksimalne i minimalne vrijednosti.

Alpha-Beta obrezivanje je tehnika optimizacije minimax algoritma. Ova tehnika reže grane u stablu igre koje ne bi imale utjecaja na rezultat jer je već pronađen potez koji je bolji od najboljeg potencijalnog poteza eliminirane grane. Primjena ove tehnike omogućuje brži prelazak kroz stablo igre i time mogućnost dubljeg analiziranja igre u istom vremenu. Algoritam funkcionira tako da prosljeđuje alfa i beta parametre kroz stablo igre. Alfa parametar određuje minimalnu garantiranu vrijednost funkcije za igrača koji želi maksimalnu vrijednost funkcije, a beta suprotno. Kada je u stablu igre već pronađena solucija koja je nepovoljnija od vrijednosti parametra alfe ili bete, njezini se susjedi u stablu igre mogu uopće ne analizirati, jer iako takvi potezi mogu biti bolji za jednu od strana, nikada neće biti izabrani od obje strane, što je potrebno da potez dođe do vrha stabla.



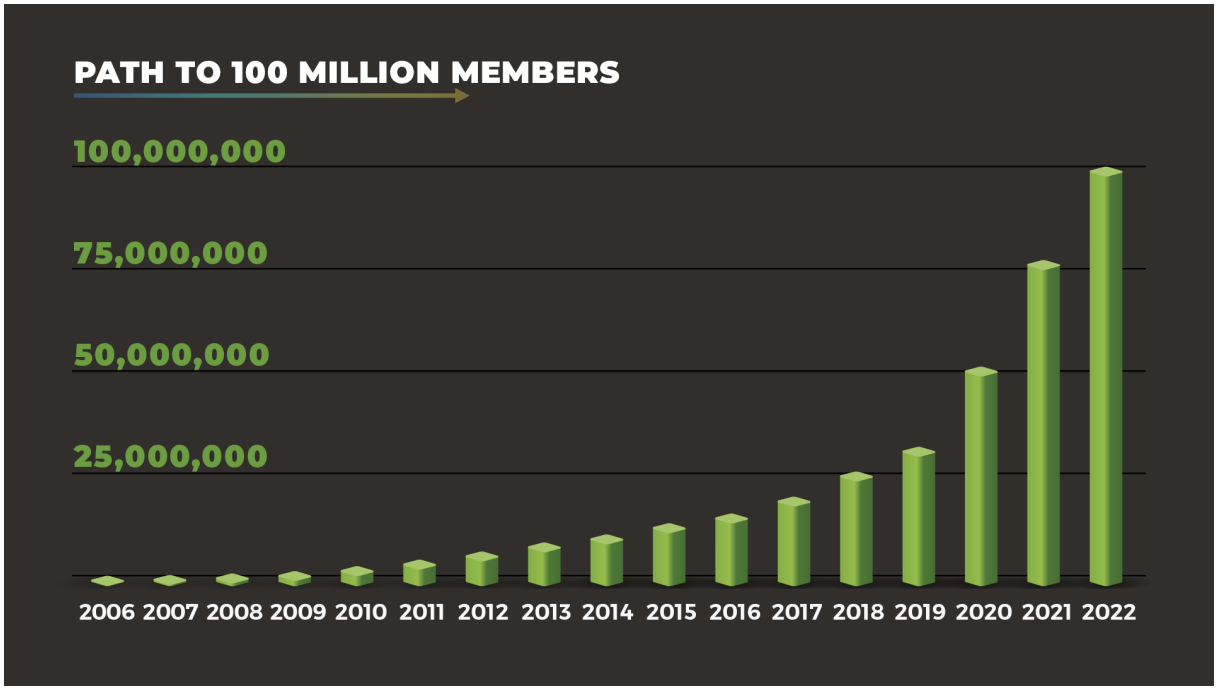
Slika 30. Primjer alpha-beta obrezivanja [17]

Čvorovi za maksimiziranje poprimaju alfa vrijednost i nasljeđuju beta vrijednost. Čvorovi za minimiziranje funkcioniraju suprotno. Na slici 30. prikazan je prvi korak u algoritmu. U maksimiziranju D čvora vrijednost čvora je proslijeđena u vrijednost beta parametra u B čvoru. Čvor E je automatski naslijedio vrijednost B parametra. Nakon što je otkriveno da je vrijednost prvog člana u E čvoru 5, alfa parametar poprimio je njegovu vrijednost. Kada je alfa veća od bete u čvoru za maksimiziranje vrijednosti, ostale se vrijednosti odbacuju bez razmatranja.

5. SLIČNI UREĐAJI

5.1. Šahovsko tržište

Internetska povezanost omogućila je da ljudi iz udobnosti svojeg doma u roku od par sekundi pronađu ljudskog protivnika jednake jačine, da gledaju visokokvalitetne analize partija velemajstora i prate turnire koji se odvijaju na drugom kraju svijeta. Tome je doprinijela i *Netflix* serija Damin Gambit koja je izašla krajem 2020. godine kada je većina svijeta bila kod kuće zbog epidemije koronavirusa i tako u savršeno vrijeme nanovo zainteresirala ljude za ovu igru. Zbog toga je *online* igranje šaha postalo vrlo popularno što se vidi iz podatka da je platforma za igranje šaha Chess.com 2022. godine dostigla 100 milijuna korisnika. Rapidan rast broja igrača registriranih na ovu aplikaciju prikazan je na slici 31.



Slika 31. Rast registriranih igrača na platformi Chess.com [18]

Svjetsko šahovsko tržište bilo je procijenjeno na 2.3 milijarde dolara 2023. godine s procijenjenim godišnjim porastom od 4 % u periodu od 2023.-2030. godine. To znači da bi do 2030. godine trebalo doseći 3 milijarde dolara. [19]

5.2. Slični proizvodi

Tako velika populacija igrača stvara veliko tržište za inovativnim proizvodima koji mogu na neki način pružati novi način igranja šaha.

U trenutku pisanja ovog rada samo jedna tvrtka aktivno proizvodi i isporučuje šahovske ploče koji sami igraju poteze protiv igrača. To je tvrtka *Square Off* bazirana u Indiji koja proizvodi *Grand Kingdom Set* po cijeni od 479 €. Ima i prateću aplikaciju preko koje igrač može igrati s drugim ljudima. Ovaj proizvod je razvijen 2016. i generalno je dobro prihvaćen od strane tržišta zbog inovativnosti, no ovaj proizvod ima i mana kao što su sporo pomicanje figura, glasnoća motora i povremeno krivo detektiranje stanja ploče. Ostale ploče ove kompanije Neo i Swap su najavljene i nekoliko je primjerka isporučeno, no nema dovoljno podataka o kvaliteti proizvoda. Također, može se naručiti i *Phantom Chess Board* od drugog proizvođača, no već nekoliko godina tvrtka kasni s rokom isporuke proizvoda.

Slika 32. Automatska šahovska ploča Grand Kingdom Set [20]

Najnovija automatizirana šahovska ploča na tržištu je *GoChess* koja je prikupila 2 milijuna dolara preko završene Kickstarter kampanje od 5500 podupiratelja, a trenutno traje kampanja na platformi Indiegogo koja je do sada prikupila preko 3 milijuna dolara od 8500 podupiratelja.

Slika 333. GoChess automatizirana šahovska ploča [21]

Inovacije koje nudi ovaj proizvod su:

1. Evaluacija poteza u stvarnom vremenu: Igrači mogu izabrati opciju pomoći računala tijekom igre. Sva polja imaju led svjetlo koje može poprimiti više boja. Kada igrač podigne figuru sva polja na koja se figura može pomaknuti zasvijetle različitim bojama ovisno o jačini poteza. S ovom pomoći igrač može igrati protiv najjače razine kompjutera što može poboljšati igračevu mogućnost evaluacije pozicija.
2. Integracija s *Lichess* i *Chess.com* platformama: Igrači ovim putem mogu igrati u stvarnom vremenu protiv igrača diljem svijeta. Ovo je jedna od glavnih prednosti ove ploče, jer ni jedna prije nije podržavala obje glavne aplikacije za igranje šaha.

Navedene ploče su generirale više milijuna dolara na platformama za sakupljanje novaca kao što su *Kickstarter* i *Indiegogo*. Takav oblik sakupljanja investicija zove se *crowdfunding*. *Crowdfunding* kampanjama prikupljaju se sredstva putem donacija od velikog broja ljudi.

Donatori koji prilože više od određenog iznosa efektivno kupuju proizvod čije je vrijeme isporuke često upitno. Tako velika potpora unatoč poznatim problemima s kršenjem roka isporuke i netransparentnom kvalitetom proizvoda upućuje da tržište za sličnim proizvodima još uvijek nije zasićeno. Na slici 34. prikazana je SWOT analiza *crowdfunding* kampanja za automatske šahovske ploče.

Slika 344. SWOT analiza

5.3. Troškovnik

Troškovi materijala za izradu ove automatske šahovske ploče iznosili su 223,40 €. Najskuplje komponente su drveno kućište i linearne vodilice. Navedene su cijene na dan kupnje.

Tablica 3. Troškovnik

Komponenta	*Cijena / €*	*Kom*	*Uk. cijena / €*
Drveno kućište	22	1	22
Šahovske figure	7,4	1	7,4
Staklo	20	1	20
Permanentni magneti	0,15	32	4,8
Elektromagnet	4,5	1	4,5
Elektromotor 1.7 A	12,1	1	12,1
Elektromotor 0.7 A	9,5	1	9,5
TMC2208 upravljači motora	2,5	2	5
Arduino Nano ESP32	35	1	35
Linearne vodilice	11,3	2	22,6
Linearni ležajevi	3,1	2	6,2
Remenice	1,2	5	6
Remen	3,7	1	3,7
Nosiljka	2	1	2
Držači vodilica	2,2	2	4,4
Držači remenica	0,25	4	1
Eksperimentalne pločice	3	3	9
Multiplekseri	0,6	4	2,4
Reed prekidači	0,1	64	6,4
Granični prekidači	1	2	4

LCD zaslon	3,6	1	3,6
Arcade gumbovi	1,3	2	2,6
Napajanje 12 V 3,2 A	7,3	1	7,3
Step-down modul	2,5	1	2,5
Kopče za remen	1,5	2	3
Otpornik	0,2	1	0,2
Mosfet IRF3205	0,3	1	0,3
Dioda slobodnog hoda	0,5	1	0,5
Električni kondenzator	0,4	1	0,4
Razni vijci	-	-	5
Razni kablovi i konektori	-	-	10
Ukupna cijena	-	-	**223,4**

6. ZAKLJUČAK

U ovom projektu projektirana je i izrađena automatizirana šahovska ploča. Izrađen je 3D model, opisane su sve komponente kao i način njihovog povezivanja i upravljanja. Nadalje, napravljen je pregled sličnih komercijaliziranih proizvoda na tržištu.

Ploča funkcionira tako da prepoznaje poteze igrača koristeći reed prekidače smještene ispod svakog šahovskog polja. Zatim Arduino mikrokontroler smisli potez koristeći minimalni šahovski program koji odabire poteze Minimax algoritmom. Figure se pomiču elektromagnetom koji se aktivira kada je pozicioniran ispod figure, vežući permanentni magnet u figuri za sebe i vodeći je do odabranog polja. Dva Nema 17 elektromotora prenose gibanje na elektromagnet remenicama i remenom. Precizno gibanje postiže se paralelno postavljenim linearnim vodilicama, a minimalno trenje linearnim kugličnim ležajevima.

Ovakva ploča može pomoći odvojiti ljude od ekrana i pružati novi, interesantan način igranja šaha.

LITERATURA

[1] Pravila šaha: https://new.uschess.org/news/evolution-modern-chess-rules-enter-queen-and-bishop, Pristupljeno: 1. 5. 2024.

[2] Prednosti šaha: https://educaplus-zg.com/index.php/program/skola-saha, Pristupljeno: 1. 5. 2024.

[3] Kasparov protiv računala: https://www.kasparov.com/timeline-event/deep-blue/, Pristupljeno: 1. 5. 2024.

[4] Slika elektromotora: https://www.robotgear.com.au/Product.aspx/Details/459-Stepper-Motor-Bipolar-200-Steps-Rev-3-9V-600mA-180g-cm-torque, Pristupljeno: 1. 5. 2024.

[5] Elektromotor: https://www.phidgets.com/docs21/Stepper_Motor_and_Controller_Primer, Pristupljeno: 1. 5. 2024.

[6] Specifikacije jačeg elektromotora nazivne struje 1,7 A: https://www.aliexpress.com/item/1005001303445983.html?spm=a2g0o.order_list.order_list_main.5.57961802wGK9rj, Pristupljeno 1. 5. 2024.

[7] Specifikacije slabijeg elektromotora nazivne struje 0,7 A: https://www.aliexpress.com/item/1005004963609855.html?spm=a2g0o.order_list.order_list_main.32.57961802wGK9rj, Pristupljeno 1. 5. 2024.

[8] TMC2208: https://howtomechatronics.com/tutorials/arduino/stepper-motors-and-arduino-the-ultimate-guide/, Pristupljeno: 1. 5. 2024.

[9] Reed prekidač: https://www.sparkfun.com/datasheets/Components/Buttons/AN104.pdf, Pristupljeno: 1. 5. 2024.

[10] I2C protokol: https://www.i2c-bus.org/voltage-level/, Pristupljeno 1. 5. 2024.

[11] Sličan projekt: https://www.instructables.com/Automated-Chessboard/, Pristupljeno 1. 5. 2024.

[12] Sličan projekt: https://github.com/sumit11899/Automated-ChessBoard, Pristupljeno 1. 5. 2024.

[13] Sličan projekt: https://github.com/AndChen153/ChessBoard, Pristupljeno 1. 5. 2024.

[14] Arduino Nano ESP32: https://store.arduino.cc/products/nano-esp32-with-headers, Pristupljeno: 1. 5. 2024.

[15] Šahovski program Micromax: https://home.hccnet.nl/h.g.muller/max-src2.html, Pristupljeno: 1. 5. 2024.

[16] Minimax algoritam: https://www.javatpoint.com/mini-max-algorithm-in-ai, Pristupljeno: 1. 5. 2024.

[17] Alfa-beta obrezivanje: https://www.javatpoint.com/ai-alpha-beta-pruning, Pristupljeno: 1. 5. 2024.

[18] Broj korisnika *Chess.com* aplikacije: https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.chess.com%2Farticle%2Fview%2Fchesscom-reaches-100-million-members&psig=AOvVaw1ec3C8Ggv1gWLXgyenEbjd&ust=1713899657297000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCIjRjYTG1oUDFQAAAAAdAAAAABAE, Pristupljeno: 1. 5. 2024.

[19] Projekcija rasta šahovskog tržišta: https://www.cognitivemarketresearch.com/regional-analysis/europe-chess-market-report, Pristupljeno: 1. 5. 2024.

[20] *Square Off* šahovska ploča: https://squareoffnow.com/product/gks, Pristupljeno: 1. 5. 2024.

[21] *GoChess* kampanja: https://www.indiegogo.com/projects/gochess-most-powerful-chess-board-ever-invented#/, Pristupljeno: 1. 5. 2024.

PRILOZI

I. Arduino kod

II. Tehnička dokumentacija

automatizirana_sahovska_ploca § | global.h | micro_max.cpp | micro_max.h

```
#include "global.h"
#include "micro_max.h"
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x20, 16, 2);

void setup() {
  Serial.begin(9600);
  //  Elektromagnet
  pinMode (MAGNET, OUTPUT);
  //  Motori
  pinMode (MOTOR_WHITE_STEP, OUTPUT);
  pinMode (MOTOR_WHITE_DIR, OUTPUT);
  pinMode (MOTOR_BLACK_STEP, OUTPUT);
  pinMode (MOTOR_BLACK_DIR, OUTPUT);
    //  Arcade gumbovi - granični prekidači
  pinMode (BUTTON_WHITE_SWITCH_MOTOR_WHITE, INPUT_PULLUP);
  pinMode (BUTTON_BLACK_SWITCH_MOTOR_BLACK, INPUT_PULLUP);
  for (byte i = 0; i < 4; i++) {
    pinMode (MUX_ADDR [i], OUTPUT);
    digitalWrite(MUX_ADDR [i], LOW);
    pinMode (MUX_SELECT [i], OUTPUT);
    digitalWrite(MUX_SELECT [i], HIGH);
  }
  pinMode (MUX_OUTPUT, INPUT_PULLUP);
  //  Postavljanje početne šahovske pozicije
  for (byte i = 2; i < 6; i++) {
    for (byte j = 0; j < 8; j++) {
      reed_sensor_status[i][j] = 1;
      reed_sensor_status_memory[i][j] = 1;
    }
  }
  //  MicroMax
  lastH[0] = 0;
  //  LCD
  lcd.init();
  lcd_display();
  //  Vrijeme
  timer = millis();
}

void loop() {
  switch (sequence) {
    case calibration:
      lcd_display();
      calibrate();
      sequence = player_white;
      break;
    case player_white:
      if (millis() - timer > 995) {  // Potez igrača
        countdown();
        lcd_display();
      }
      detect_human_movement();
      if (button(WHITE) == true) {  // Kraj poteza igrača
        new_turn_countdown = true;
        player_displacement();
        AI_HvsC();
        sequence = player_black;
        break;
      case player_black:
          black_player_movement();  //  Potez kompjutera
          sequence = player_white;
        break;
      }
  }
}
```

```
// PREKIDAČI
boolean button(byte type) {
  if (type == WHITE && digitalRead(BUTTON_WHITE_SWITCH_MOTOR_WHITE) != HIGH) {
    delay(250);
    return true;
  }
  if (type == BLACK && digitalRead(BUTTON_BLACK_SWITCH_MOTOR_BLACK) != HIGH) {
    delay(250);
    return true;
  }
  return false;
}
// KALIBRACIJA
void calibrate() {
  //  Spori pomak do graničnih prekidača
  while (digitalRead(BUTTON_WHITE_SWITCH_MOTOR_WHITE) == HIGH) motor(B_T, SPEED_SLOW, calibrate_speed);
  while (digitalRead(BUTTON_BLACK_SWITCH_MOTOR_BLACK) == HIGH) motor(L_R, SPEED_SLOW, calibrate_speed);
  delay(500);
  //  Startna pozicija e7
  motor(R_L, SPEED_FAST, TROLLEY_START_POSITION_X);
  motor(T_B, SPEED_FAST, TROLLEY_START_POSITION_Y);
  delay(500);
}

//  MOTOR
void motor(byte direction, int speed, float distance) {
  float step_number = 0;
  //  Izračun udaljenosti
  if (distance == calibrate_speed) step_number = 4;
  else if (direction == LR_BT || direction == RL_TB || direction == LR_TB || direction == RL_BT)
                                          step_number = distance * SQUARE_SIZE * 1.44;
  else step_number = distance * SQUARE_SIZE;
  //  Smjer gibanja elektromotora
  if (direction == R_L || direction == T_B || direction == RL_TB) digitalWrite(MOTOR_WHITE_DIR, HIGH);
  else digitalWrite(MOTOR_WHITE_DIR, LOW);
  if (direction == B_T || direction == R_L || direction == RL_BT) digitalWrite(MOTOR_BLACK_DIR, HIGH);
  else digitalWrite(MOTOR_BLACK_DIR, LOW);
  //  Aktivacija elektromotora
  for (int x = 0; x < step_number; x++) {
    if (direction == LR_TB || direction == RL_BT) digitalWrite(MOTOR_WHITE_STEP, LOW);
    else digitalWrite(MOTOR_WHITE_STEP, HIGH);
    if (direction == LR_BT || direction == RL_TB) digitalWrite(MOTOR_BLACK_STEP, LOW);
    else digitalWrite(MOTOR_BLACK_STEP, HIGH);
    delayMicroseconds(speed);
    digitalWrite(MOTOR_WHITE_STEP, LOW);
    digitalWrite(MOTOR_BLACK_STEP, LOW);
    delayMicroseconds(speed);
  }
}
//  ELEKTROMAGNET
void electromagnet(boolean state) {
  if (state == true)  {
    digitalWrite(MAGNET, HIGH);
    delay(600);
  }
  else  {
    delay(600);
    digitalWrite(MAGNET, LOW);
  }
}
```

```
// VRIJEME
void countdown() {
  //  Set the time of the current player
  if (new_turn_countdown == true ) {
    new_turn_countdown = false;
    if (sequence == player_white) {
      second = second_white;
      minute = minute_white;
    }
    else if (sequence == player_black) {
      second = second_black;
      minute = minute_black;
    }
  }
  timer = millis();
  second = second - 1;
  if (second < 1) {
    second = 60;
    minute = minute - 1;
  }
  if (sequence == player_white) {
    second_white = second;
    minute_white = minute;
  }
  else if (sequence == player_black) {
    second_black = second;
    minute_black = minute;
  }
}

  //  Vraćanje elektromagneta do startne pozicije
  int convert_table [] = {0, 7, 6, 5, 4, 3, 2, 1, 0};
  byte white_capturing = 1;
  if (reed_sensor_status_memory[convert_table[arrival_coord_Y]][arrival_coord_X - 1] == 0) white_capturing = 0;

  for (byte i = white_capturing; i < 2; i++) {
    if (i == 0) {
      displacement_X = abs(arrival_coord_X - trolley_coordinate_X);
      displacement_Y = abs(arrival_coord_Y - trolley_coordinate_Y);
    }
    else if (i == 1) {
      displacement_X = abs(departure_coord_X - trolley_coordinate_X);
      displacement_Y = abs(departure_coord_Y - trolley_coordinate_Y);
    }
    if (departure_coord_X > trolley_coordinate_X) motor(T_B, SPEED_FAST, displacement_X);
    else if (departure_coord_X < trolley_coordinate_X) motor(B_T, SPEED_FAST, displacement_X);
    if (departure_coord_Y > trolley_coordinate_Y) motor(L_R, SPEED_FAST, displacement_Y);
    else if (departure_coord_Y < trolley_coordinate_Y) motor(R_L, SPEED_FAST, displacement_Y);
    if (i == 0) {
      electromagnet(true);
      motor(R_L, SPEED_SLOW, 0.5);
      motor(B_T, SPEED_SLOW, arrival_coord_X - 0.5);
      electromagnet(false);
      motor(L_R, SPEED_FAST, 0.5);
      motor(T_B, SPEED_FAST, arrival_coord_X - 0.5);
      trolley_coordinate_X = arrival_coord_X;
      trolley_coordinate_Y = arrival_coord_Y;
    }
  }
  trolley_coordinate_X = arrival_coord_X;
  trolley_coordinate_Y = arrival_coord_Y;
  //  Pomaci do željenog polja
  displacement_X = abs(arrival_coord_X - departure_coord_X);
  displacement_Y = abs(arrival_coord_Y - departure_coord_Y);
  electromagnet(true);
```

```
//  Micanje konja
if (displacement_X == 1 && displacement_Y == 2 || displacement_X == 2 && displacement_Y == 1) {
  if (displacement_Y == 2) {
    if (departure_coord_X < arrival_coord_X) {
      motor(T_B, SPEED_SLOW, displacement_X * 0.5);
      if (departure_coord_Y < arrival_coord_Y) motor(L_R, SPEED_SLOW, displacement_Y);
      else motor(R_L, SPEED_SLOW, displacement_Y);
      motor(T_B, SPEED_SLOW, displacement_X * 0.5);
    }
    else if (departure_coord_X > arrival_coord_X) {
      motor(B_T, SPEED_SLOW, displacement_X * 0.5);
      if (departure_coord_Y < arrival_coord_Y) motor(L_R, SPEED_SLOW, displacement_Y);
      else motor(R_L, SPEED_SLOW, displacement_Y);
      motor(B_T, SPEED_SLOW, displacement_X * 0.5);
    }
  }
  else if (displacement_X == 2) {
    if (departure_coord_Y < arrival_coord_Y) {
      motor(L_R, SPEED_SLOW, displacement_Y * 0.5);
      if (departure_coord_X < arrival_coord_X) motor(T_B, SPEED_SLOW, displacement_X);
      else motor(B_T, SPEED_SLOW, displacement_X);
      motor(L_R, SPEED_SLOW, displacement_Y * 0.5);
    }
    else if (departure_coord_Y > arrival_coord_Y) {
      motor(R_L, SPEED_SLOW, displacement_Y * 0.5);
      if (departure_coord_X < arrival_coord_X) motor(T_B, SPEED_SLOW, displacement_X);
      else motor(B_T, SPEED_SLOW, displacement_X);
      motor(R_L, SPEED_SLOW, displacement_Y * 0.5);
    }
  }
}
//  Dijagonalno kretanje
else if (displacement_X == displacement_Y) {
  if (departure_coord_X > arrival_coord_X && departure_coord_Y > arrival_coord_Y)
      motor(RL_BT, SPEED_SLOW, displacement_X);
  else if (departure_coord_X > arrival_coord_X && departure_coord_Y < arrival_coord_Y)
      motor(LR_BT, SPEED_SLOW, displacement_X);
  else if (departure_coord_X < arrival_coord_X && departure_coord_Y > arrival_coord_Y)
      motor(RL_TB, SPEED_SLOW, displacement_X);
  else if (departure_coord_X < arrival_coord_X && departure_coord_Y < arrival_coord_Y)
  motor(LR_TB, SPEED_SLOW, displacement_X);
}
//  Rohada
else if (departure_coord_X == 5 && departure_coord_Y == 8 && arrival_coord_X == 7 && arrival_coord_Y == 8) {
  motor(R_L, SPEED_SLOW, 0.5);
  motor(T_B, SPEED_SLOW, 2);
  electromagnet(false);
  motor(T_B, SPEED_FAST, 1);
  motor(L_R, SPEED_FAST, 0.5);
  electromagnet(true);
  motor(B_T, SPEED_SLOW, 2);
  electromagnet(false);
  motor(T_B, SPEED_FAST, 1);
  motor(R_L, SPEED_FAST, 0.5);
  electromagnet(true);
  motor(L_R, SPEED_SLOW, 0.5);
}
else if (departure_coord_X == 5 && departure_coord_Y == 8 && arrival_coord_X == 3 && arrival_coord_Y == 8) {
  motor(R_L, SPEED_SLOW, 0.5);
  motor(B_T, SPEED_SLOW, 2);
  electromagnet(false);
  motor(B_T, SPEED_FAST, 2);
  motor(L_R, SPEED_FAST, 0.5);
  electromagnet(true);
  motor(T_B, SPEED_SLOW, 3);
  electromagnet(false);
  motor(B_T, SPEED_FAST, 1);
  motor(R_L, SPEED_FAST, 0.5);
  electromagnet(true);
  motor(L_R, SPEED_SLOW, 0.5);
}
```

```
  //  Okomito pomicanje
  else if (displacement_Y == 0) {
    if (departure_coord_X > arrival_coord_X) motor(B_T, SPEED_SLOW, displacement_X);
    else if (departure_coord_X < arrival_coord_X) motor(T_B, SPEED_SLOW, displacement_X);
  }
  //  Uzdužno pomicanje
  else if (displacement_X == 0) {
    if (departure_coord_Y > arrival_coord_Y) motor(R_L, SPEED_SLOW, displacement_Y);
    else if (departure_coord_Y < arrival_coord_Y) motor(L_R, SPEED_SLOW, displacement_Y);
  }
  electromagnet(false);
  //  Mijenjanje statusa reed prekidača nakon poteza
  reed_sensor_status_memory[convert_table[departure_coord_Y]][departure_coord_X - 1] = 1;
  reed_sensor_status_memory[convert_table[arrival_coord_Y]][arrival_coord_X - 1] = 0;
  reed_sensor_status[convert_table[departure_coord_Y]][departure_coord_X - 1] = 1;
  reed_sensor_status[convert_table[arrival_coord_Y]][arrival_coord_X - 1] = 0;
}
```

```
// LCD
void lcd_display() {
  lcd.backlight();
  if (no_valid_move == true) {
    lcd.setCursor(0, 0);
    lcd.print("  INVALIDAN POTEZ  ");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    delay(2000);
    no_valid_move = false;
    return;
  }
  switch (sequence) {
    case calibration:
      lcd.setCursor(0, 0);
      lcd.print("  CALIBRATION   ");
      lcd.setCursor(0, 1);
      lcd.print("                ");
      break;
    case player_white:
      lcd.setCursor(0, 0);
      lcd.print("     BIJELI      ");
      lcd.setCursor(0, 1);
      lcd.print("     " + String(minute) + " : " + String(second) + "     ");
      break;
    case player_black:
      lcd.setCursor(0, 0);
      lcd.print("     CRNI      ");
      lcd.setCursor(0, 1);
      lcd.print("     " + String(minute) + " : " + String(second) + "     ");
      break;
  }
}
```

```
// DETEKCIJA POTEZA IGRAČA
void detect_human_movement() {
  //  Record the reed switches status
  byte column = 6;
  byte row = 0;
  for (byte i = 0; i < 4; i++) {
    digitalWrite(MUX_SELECT[i], LOW);
    for (byte j = 0; j < 16; j++) {
      for (byte k = 0; k < 4; k++) {
        digitalWrite(MUX_ADDR [k], MUX_CHANNEL [j][k]);
      }
      reed_sensor_record[column][row] = digitalRead(MUX_OUTPUT);
      row++;
      if (j == 7) {
        column++;
        row = 0;
      }
    }
    for (byte l = 0; l < 4; l++) {
      digitalWrite(MUX_SELECT[l], HIGH);
    }
    if (i == 0) column = 4;
    if (i == 1) column = 2;
    if (i == 2) column = 0;
    row = 0;
  }
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
      reed_sensor_status_memory[7 - i][j] = reed_sensor_record[i][j];
    }
  }
  // Usporedba prošlog i sadašnjeg stanja reed prekidača
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
      if (reed_sensor_status[i][j] != reed_sensor_status_memory[i][j]) {
        if (reed_sensor_status_memory[i][j] == 1) {
          reed_colone[0] = i;
          reed_line[0] = j;
        }
        if (reed_sensor_status_memory[i][j] == 0) {
          reed_colone[1] = i;
          reed_line[1] = j;
        }
      }
    }
  }

  //  Mijenjanje memorije ploče
  for (byte i = 0; i < 8; i++) {
    for (byte j = 0; j < 8; j++) {
      reed_sensor_status[i][j] = reed_sensor_status_memory[i][j];
    }
  }
}
// GENERIRANJE POTEZA
void player_displacement() {
  //  Pretvorba notacije poteza
  char table1[] = {'8', '7', '6', '5', '4', '3', '2', '1'};
  char table2[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'};
  mov[0] = table2[reed_line[0]];
  mov[1] = table1[reed_colone[0]];
  mov[2] = table2[reed_line[1]];
  mov[3] = table1[reed_colone[1]];
}
```

automatizirana_sahovska_ploca § | global.h § | micro_max.cpp § | micro_max.h

```
int reed_sensor_status [8][8];
int reed_sensor_record [8][8];
int reed_sensor_status_memory [8][8];
const byte SQUARE_SIZE = 195;
const int SPEED_SLOW (3000);
const int SPEED_FAST (1000);
const float TROLLEY_START_POSITION_X = 0.78;
const float TROLLEY_START_POSITION_Y = 4.65;
byte trolley_coordinate_X = 5;
byte trolley_coordinate_Y = 7;
char mov [4] = {0, 0, 0, 0};
boolean no_valid_move = false;
byte reed_colone[2];
byte reed_line[2];
enum {calibration, player_white, player_black};
byte sequence = calibration;
enum {T_B, B_T, L_R, R_L, LR_BT, RL_TB, LR_TB, RL_BT, calibrate_speed};
const byte MAGNET (6);
const byte MOTOR_WHITE_DIR (2);
const byte MOTOR_WHITE_STEP (3);
const byte MOTOR_BLACK_DIR (4);
const byte MOTOR_BLACK_STEP (5);
byte second = 60;
byte minute = 9;
byte second_white = second;
byte second_black = second;
byte minute_white = minute;
byte minute_black = minute;
unsigned long timer = 0;
boolean start_black = true;
boolean new_turn_countdown = false;

const byte MUX_ADDR [4] = {A3, A2, A1, A0};
const byte MUX_SELECT [4] = {13, 9, 8, 7};
const byte MUX_OUTPUT (12);
const byte MUX_CHANNEL[16][4] = {
  {0, 0, 0, 0},
  {1, 0, 0, 0},
  {0, 1, 0, 0},
  {1, 1, 0, 0},
  {0, 0, 1, 0},
  {1, 0, 1, 0},
  {0, 1, 1, 0},
  {1, 1, 1, 0},
  {0, 0, 0, 1},
  {1, 0, 0, 1},
  {0, 1, 0, 1},
  {1, 1, 0, 1},
  {0, 0, 1, 1},
  {1, 0, 1, 1},
  {0, 1, 1, 1},
  {1, 1, 1, 1}
};
const byte BUTTON_WHITE_SWITCH_MOTOR_WHITE (11);
const byte BUTTON_BLACK_SWITCH_MOTOR_BLACK (10);
enum {WHITE, BLACK};
extern char lastH[], lastM[];
```

automatizirana_sahovska_ploca § | global.h § | micro_max.cpp § | micro_max.h

```
#include "Arduino.h"
#include "Micro_Max.h"
#define W while
#define M 0x88
#define S 128
#define I 8000
#define MYRAND_MAX 65535
long  N, T;
short Q, O, K, R, k = 16;
char *p, c[5], Z;
char L,
     w[] = {0, 2, 2, 7, -1, 8, 12, 23},
           o[] = { -16, -15, -17, 0, 1, 16, 0, 1, 16, 15, 17, 0, 14, 18, 31, 33, 0,
                   7, -1, 11, 6, 8, 3, 6,
                   6, 3, 5, 7, 4, 5, 3, 6
                 };
char b[] = {
  22, 19, 21, 23, 20, 21, 19, 22, 28, 21, 16, 13, 12, 13, 16, 21,
  18, 18, 18, 18, 18, 18, 18, 18, 22, 15, 10,  7,  6,  7, 10, 15,
  0,  0,  0,  0,  0,  0,  0,  0, 18, 11,  6,  3,  2,  3,  6, 11,
  0,  0,  0,  0,  0,  0,  0,  0, 16,  9,  4,  1,  0,  1,  4,  9,
  0,  0,  0,  0,  0,  0,  0,  0, 16,  9,  4,  1,  0,  1,  4,  9,
  0,  0,  0,  0,  0,  0,  0,  0, 18, 11,  6,  3,  2,  3,  6, 11,
  9,  9,  9,  9,  9,  9,  9,  9, 22, 15, 10,  7,  6,  7, 10, 15,
  14, 11, 13, 15, 12, 13, 11, 14, 28, 21, 16, 13, 12, 13, 16, 21, 0
};
char bk[16 * 8 + 1];
unsigned int seed = 0;
uint32_t  byteBoard[8];
char sym[17] = {".?pnkbrq?P?NKBRQ"};
int mn = 1;
char lastH[5], lastM[5];
unsigned short ledv = 1;
String inputString = "";
bool stringComplete = false;
int r;
unsigned short myrand(void) {
  unsigned short r = (unsigned short)(seed % MYRAND_MAX);
  return r = ((r << 11) + (r << 7) + r) >> 1;
}

short D(short q, short l, short e, unsigned char E, unsigned char z, unsigned char n) {
  short m, v, i, P, V, s;
  unsigned char t, p, u, x, y, X, Y, H, B, j, d, h, F, G, C;
  signed char r;
  if (++Z > 30) {
    --Z; return e;
  }
  q--;
  k ^= 24;
  d = Y = 0;
  X = myrand() & ~M;
  W(d++ < n || d < 3 ||
    z & K == I && (N < T & d < 98 ||
                   (K = X, L = Y & ~M, d = 3)))
  { x = B = X;
```

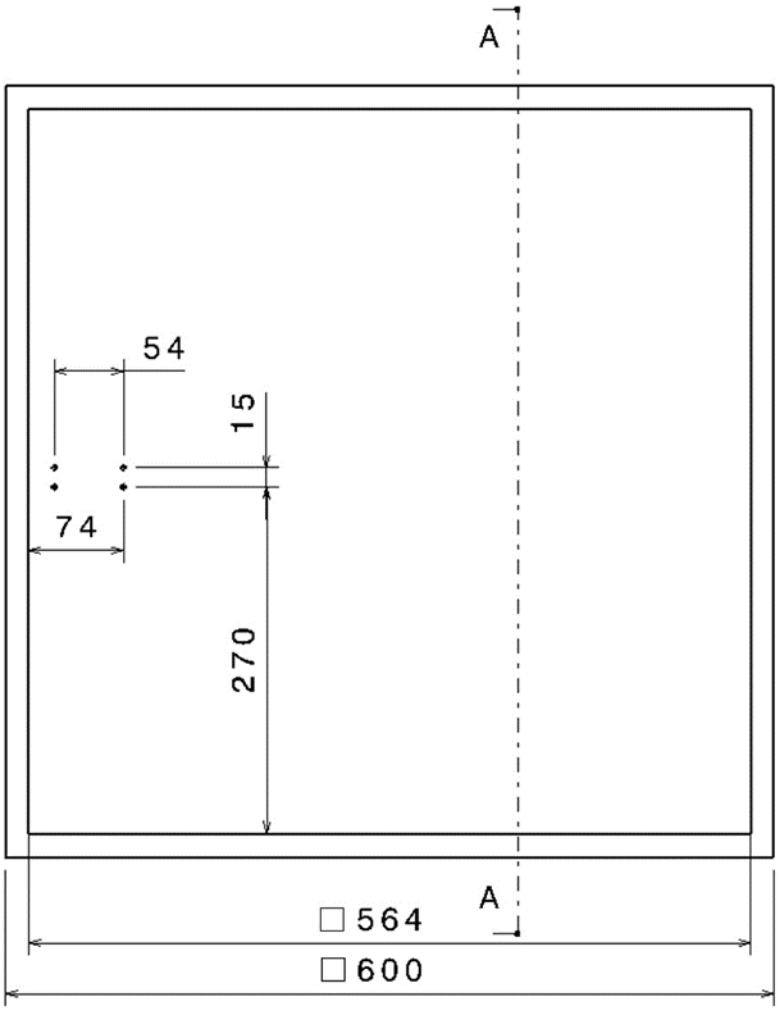
```
h = Y & S;
P = d < 3 ? I : D(-l, l - l, -e, S, 0, d - 3);
m = -P < l | R > 35 ? d > 2 ? -I : e : -P;
++N;
do {
  u = b[x];
  if (u & k) {
    r = p = u & 7;
    j = o[p + 16];
    W(r = p > 2 & r < 0 ? -r : -o[++j])
    { A:
      y = x; F = G = S;
      do {
        H = y = h ? Y ^ h : y + r;
        if (y & M)break;
        m = E - S & b[E] && y - E < 2 & E - y < 2 ? I : m;
        if (p < 3 & y == E)H ^= 16;
        t = b[H]; if (t & k | p < 3 & !(y - x & 7) - !t)break;
        i = 37 * w[t & 7] + (t & 192);
        m = i < 0 ? I : m;
        if (m >= l & d > l)goto C;
        v = d - l ? e : i - p;
        if (d - !t > l)
        { v = p < 6 ? b[x + 8] - b[y + 8] : 0;
          b[G] = b[H] = b[x] = 0; b[y] = u | 32;
          if (!(G & M))b[F] = k + 6, v += 50;
          v -= p - 4 | R > 29 ? 0 : 20;
          if (p < 3)
          { v -= 9 * ((x - 2 & M || b[x - 2] - u) +
                      (x + 2 & M || b[x + 2] - u) - l
                      + (b[x ^ 16] == k + 36))
                 - (R >> 2);
            V = y + r + l & S ? 647 - p : 2 * (u & y + 16 & 32);
            b[y] += V; i += V;
          }
          v += e + i; V = m > q ? m : q;
          C = d - l - (d > 5 & p > 2 & !t & !h);
          C = R > 29 | d < 3 | P - I ? C : d;
          do
            s = C > 2 | v > V ? -D(-l, -V, -v,
                                   F, 0, C) : v;

          W(s > q&++C < d); v = s;
          if (z && K - I && v + I && x == K & y == L)
          { Q = -e - i; O = F;
            R += i >> 7; --Z; return l;
          }
          b[G] = k + 6; b[F] = b[y] = 0; b[x] = u; b[H] = t;
        }
        if (v > m)
          m = v, X = x, Y = y | S & F;
        if (h) {
          h = 0;
          goto A;
        }
        if (x + r - y | u & 32 |
            p > 2 & (p - 4 | j - 7 ||
                     b[G = x + 3 ^ r >> 1 & 7] - k - 6
                     || b[G ^ 1] | b[G ^ 2])
           )t += p < 5;
        else F = y;
      } W(!t);
  }
} W((x = x + 9 & ~M) - B);
```

```
C: if (m > I - M | m < M - I)d = 98;
    m = m + I | P == I ? m : 0;
    if (z && d > 2)
    { *c = 'a' + (X & 7); c[1] = '8' - (X >> 4); c[2] = 'a' + (Y & 7); c[3] = '8' - (Y >> 4 & 7); c[4] = 0;
      char buff[150];
    }
  }
  k ^= 24;
  --Z; return m += m < e;
}

void gameOver() {
  for (;;);
}

void bkp() {
  for (int i = 0; i < 16 * 8 + 1; i++) {
    bk[i] = b[i];
  }
}

void serialBoard() {
  Serial.println("  +------------------+");
  for (int i = 0; i < 8; i++) {
    Serial.print(' ');
    Serial.print(8 - i);
    Serial.print("| ");
    for (int j = 0; j < 8; j++) {
      char c = sym[b[16 * i + j] & 15];
      Serial.print(c);
      Serial.print(' ');
    }
    Serial.println('|');
  }
  Serial.println("  +------------------+");
  Serial.println("    a b c d e f g h");
}
```

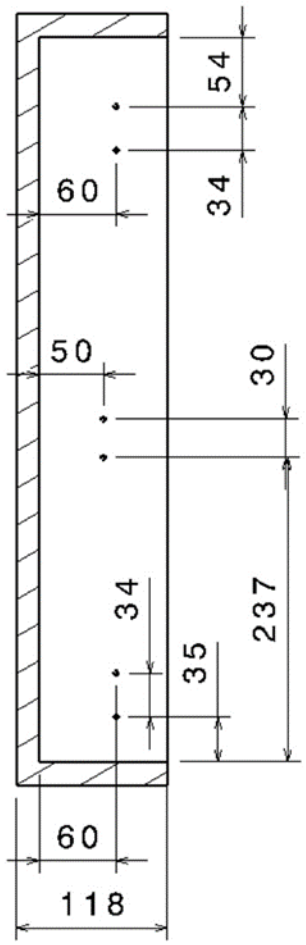
```
void AI_HvsC() {
  inputString += mov;
  Serial.print(mn);
  Serial.print(". ");
  Serial.print(inputString.substring(0, 4));
  c[0] = inputString.charAt(0);
  c[1] = inputString.charAt(1);
  c[2] = inputString.charAt(2);
  c[3] = inputString.charAt(3);
  c[4] = 0;
  // clear the string:
  inputString = "";
  stringComplete = false;
  Serial.print(" Think ");
  K = *c - 16 * c[1] + 799, L = c[2] - 16 * c[3] + 799;
  N = 0;
  T = 0x3F;
  bkp();
  r = D(-I, I, Q, O, 1, 3);
  if ( !(r > -I + 1) ) {
    Serial.println("Lose ");
    gameOver();
  }
  if (k == 0x10) {
    Serial.println("No valid move");
    return;
  }
  strcpy(lastH, c);
  mn++;
  K = I;
  N = 0;
  T = 0x3F;
  r = D(-I, I, Q, O, 1, 3);
  if ( !(r > -I + 1) ) {
    Serial.println("Lose*");
    gameOver();
  }
  strcpy(lastM, c);
  r = D(-I, I, Q, O, 1, 3);
  if ( !(r > -I + 1) ) {
    Serial.println(lastM);
    gameOver();
  }
  Serial.println(lastM);
  serialBoard();
}
```

automatizirana_sahovska_ploca § | global.h § | micro_max.cpp § | micro_max.h §

```
unsigned short myrand(void);
void gameOver();
void bkp();
void serialBoard();
void AI_HvsH();
void AI_HvsC();

extern char mov [];
extern byte sequence;
extern boolean no_valid_move;
```



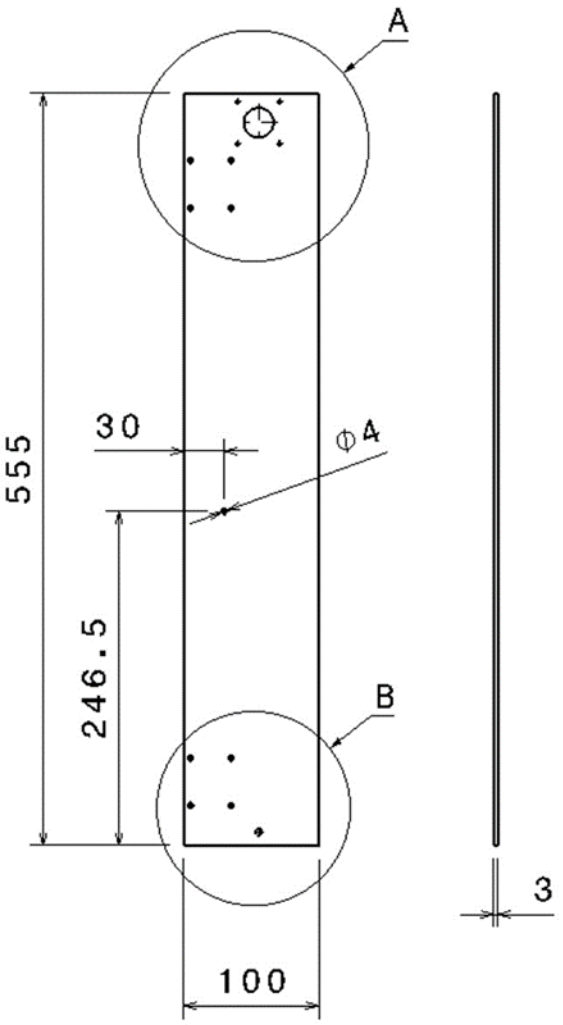
Presjek A-A

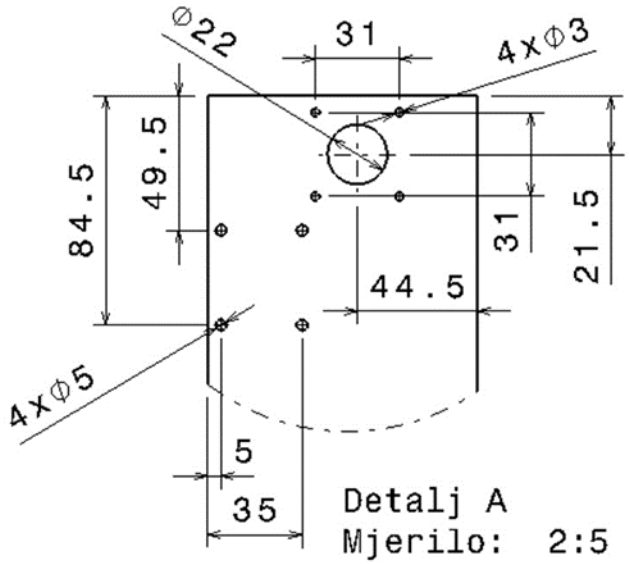


	Datum	Ime i prezime	Potpis
Projektirao	20. 04. 2024.	Eros Stemberger	
Razradio	20. 04. 2024.	Eros Stemberger	
Crtao	20. 04. 2024.	Eros Stemberger	
Pregledao			

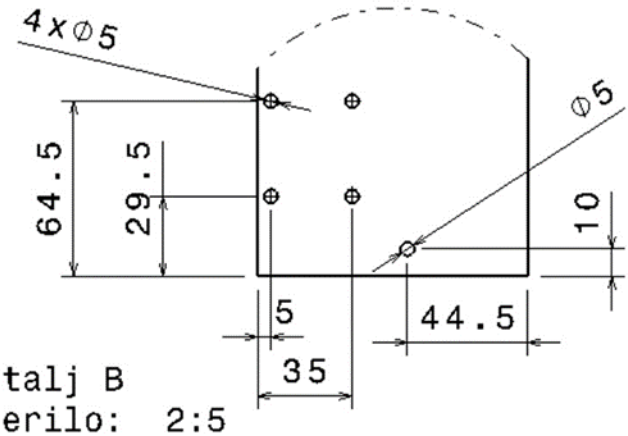
FSB Zagreb

Objekt:	Objekt broj:
	R. N. broj:
Napomena: Svi promjeri su Ø4 mm	Kopija
Materijal: Drvo	Masa: 4,2 kg
Mjerilo orginala 1:5	Naziv: Kućište
	Pozicija:
	Format: A4
	Listova: 3
Crtež broj: 202404ASP1	List: 1





Detalj A
Mjerilo: 2:5

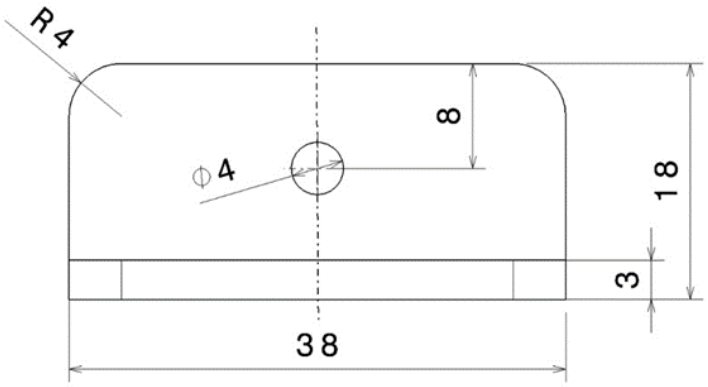


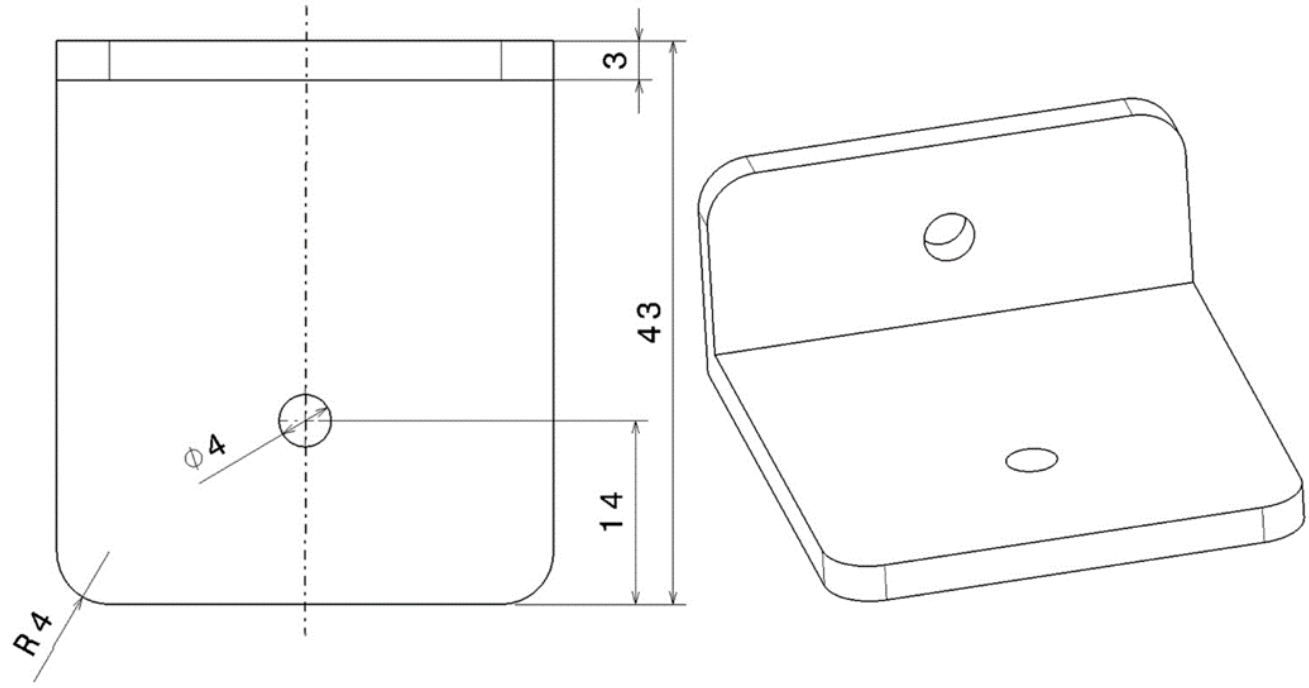
Detalj B
Mjerilo: 2:5

	Datum	Ime i prezime	Potpis
Projektirao	20. 04. 2024.	Eros Stemberger	
Razradio	20. 04. 2024.	Eros Stemberger	
Crtao	20. 04. 2024.	Eros Stemberger	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija
Materijal: Drvo / Masa: 0,1 kg		
Mjerilo orginala 1:5	Naziv: Nosiljka / Pozicija:	Format: A4
		Listova: 3
	Crtež broj: 202404ASP2	List: 2





	Datum	Ime i prezime	Potpis
Projektirao	20. 04. 2024.	Eros Stemberger	
Razradio	20. 04. 2024.	Eros Stemberger	
Crtao	20. 04. 2024.	Eros Stemberger	
Pregledao			

FSB Zagreb

Objekt:	Objekt broj:	
	R. N. broj:	
Napomena:		Kopija
Materijal: PET / Masa: 0,15 g		
Mjerilo orginala 2:1	Naziv: Nosač elektromagneta	Pozicija:
	Crtež broj: 202404ASP3	Format: A4
		Listova: 3
		List: 1