RFID prepoznavanje pozicije za igre na ploči

Tomić, Krešimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski sveučilišni studij Računarstvo

RFID prepoznavanje pozicije za igre na ploči

Diplomski rad

Krešimir Tomić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 07.07.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Krešimir Tomić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1096R, 06.10.2019.
OIB studenta:	99922259497
Mentor:	Izv. prof. dr. sc. Ivan Aleksi
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Tomislav Matić
Član Povjerenstva 1:	Izv. prof. dr. sc. Ivan Aleksi
Član Povjerenstva 2:	Doc.dr.sc. Ivan Vidović
Naslov diplomskog rada:	RFID prepoznavanje pozicije za igre na ploči
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Temu rezervirao Krešimir Tomić - Zadatak ovog diplomskog rada je primjena polja RFID antena. Primjena ove metodologije je u prepoznavanju pozicija figura u raznim igrama na ploči, kao što su igre križić-kružić, šah, go,..
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	07.07.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 22.07.2021.

Ime i prezime studenta:	Krešimir Tomić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1096R, 06.10.2019.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom**: RFID prepoznavanje pozicije za igre na ploči**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivan Aleksi

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD

Cilj ovog diplomskog rada je omogućiti prepoznavanje pozicija figurica na igraćoj ploči korištenjem RFID tehnologije. U radu su prvo istražena već postojeća rješenja i druge korištene tehnologije. Zatim je objašnjen način rada RFID tehnologije, njene prednosti i mane te gdje se najčešće primjenjuje. Glavni dio rada je primjena RFID tehnologije na konkretnom zadatku, a u ovom slučaju je to prepoznavanje figurica za igru križić-kružić. Navedene su korištene komponente i njihove uloge u sustavu te način spajanja. Glavni problem rada je bio osmisliti na koji način će se izvesti prepoznavanje figurica, to jest je li moguće izvesti multipleksiranje RFID antena ili ne. Nađeno rješenje ima mogućnost prepoznavanja svih figurica i njihovih pozicija na igraćoj ploči te prepoznavanje kada je igra gotova. Također, postoje mogućnosti za poboljšanjem razvijenog sustava, gdje bi se mogla omogućiti autonomna igra čovjeka protiv računala, automatsko pomicanje figurica te multipleksiranje RFID antena kako bi se izbjeglo nepotrebno korištenje više RFID čitača.

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je primjena polja RFID antena. Primjena ove metodologije je u prepoznavanju pozicija figura u raznim igrama na ploči, kao što su igre križić-kružić, šah, go, itd. Ovaj rad je izrađen na primjeru igre križić-kružić.

2. POSTOJEĆA RJEŠENJA I KORIŠTENE TEHNOLOGIJE

Cilj ovog poglavlja je približiti već postojeća rješenja koja omogućuju prepoznavanje pozicija figurica na igraćoj ploči korištenjem raznih tehnologija. Rješenja koja su navedena u ovom poglavlju razlikuju se po korištenim tehnologijama i mogućnostima koje pružaju kao što je prikazano u tablici ispod.

Tablica 1. Karakteristike postojećih rješenja

	Autonomno pomicanje	Prepoznavanje zauzeća polja	Prepoznavanje vrste figure	Invazivno rješenje
DGT	NE	DA	DA	NE
Phantom	DA	DA	NE	NE
RIIDL	DA	DA	NE	NE
Square-Off	DA	DA	NE	NE
TASC	NE	DA	DA	NE

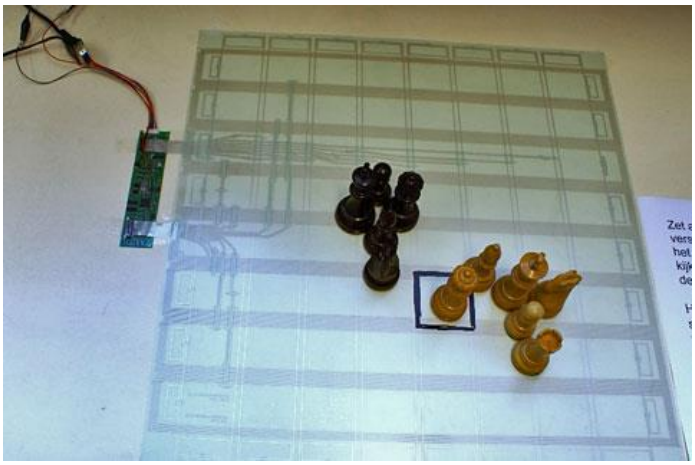
Autonomno pomicanje podrazumijeva mehaničko pomicanje figurica u igri protiv računala te takvo ponašanje odaje dojam igranja protiv čovjeka. Prepoznavanje figurica podrazumijeva mogućnost računalnog programa da u svakom trenutku zna koja je figurica na kojem polju, neovisno o tehnologiji koju primjenjuje. Na posljetku, bitno je naglasiti kako niti jedno od navedenih rješenja nije invazivno što znači da igraća ploča ima uobičajeni izgled te se ono ne narušava nikakvim mehaničkim rješenjima kao što su robotske ruke, kamere, itd.

2.1. DGT šahovska ploča

DGT šahovska ploča je razvijena od tvrtke Digital Game Technology (DGT) te joj je glavna funkcija registriranje svih poteza i time snimanje šahovskih partija. Najviše se koristi u natjecateljskom šahu gdje je snimanje poteza vrlo bitno za daljnje analize, ali se koristi i za „online“ partije, partije protiv računala, itd [1].

Slika 2.1. DGT šahovska ploča[1]

Prva verzija ovakve ploče se temeljila na RF (eng. Radio Frequency) polju u koje su se stavljale šahovske figurice. Prilikom stavljanja figurice u polje, one svojim prisustvom stvaraju rezonantne struje na prijemnoj anteni te se primljeni signali multipleksiraju kako bi se saznala pozicija određene figurice na ploči [2].



Slika 2.2. Prva verzija DGT šahovske ploče[2]

2.2. Phantom

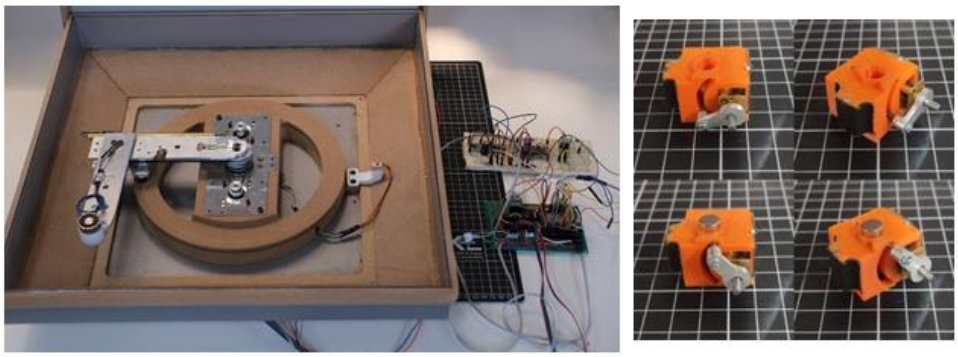
Phantom je naziv za pametnu šahovsku ploču. Ona ima mogućnost prepoznavanja igraćih figurica, a može i mehanički pomicati figurice po ploči. Problem pomicanja figurica je riješen pomoću SCARA mehanizma.

[1] Izvor slike: https://www.chesshouse.com/products/dgt-smartboard
[2] Izvor slike: https://www.chessprogramming.org/DGT_Board

Slika 2.3. Phantom[3]

SCARA (Selective Compliance Assembly Robot Arm) je tip industrijskog robota u obliku robotske ruke. Ovaj tip robotske ruke može obavljati poslove po X i Y osi, a fiksna je za Z os. SCARA sliči ljudskoj ruci tako što možemo reći da ima nadlakticu i podlakticu te joj ta osobina omogućava kretanje u svim smjerovima na način da se „lakat“ i „rame“ robotske ruke savijaju te omogućavaju nadlaktici i podlaktici pružanje do određenog šahovskog polja.

Figurice se pomiču pomoću neodimijskog magneta za koji se ispostavilo da je efikasniji od elektromagneta. Magnet se nalazi u kućištu koje ima mogućnost da ga podiže i spušta ovisno o radnji koju SCARA treba napraviti. Ukoliko je zadatak povući neku figuricu, magnet će se spustiti toliko da nekontrolirano ne povlači figurice po ploči te će se podići kada je figuricu potrebno „uhvatiti“ i pomaknuti [3].

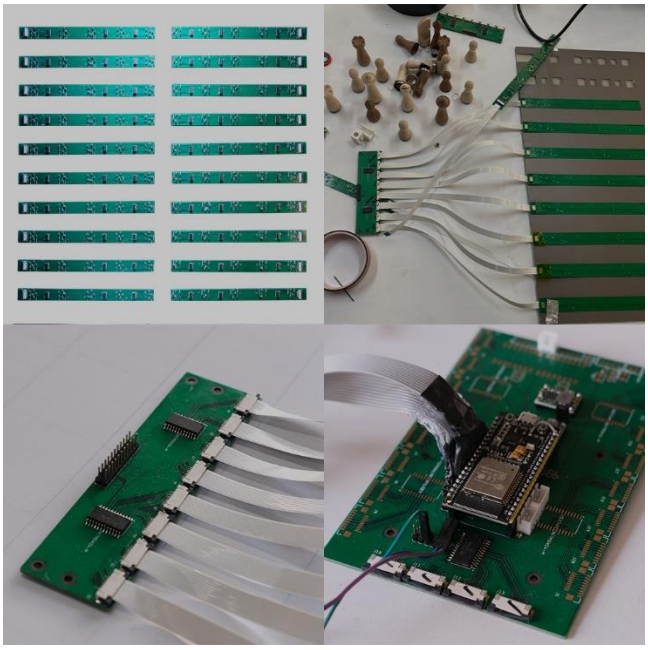


Slika 2.4. SCARA i mehanizam dizanja/spuštanja magneta[4]

Odmah ispod igraće ploče se nalazi polje hall senzora čija je uloga očitavanje prisutnosti magneta čime se registrira položaj figurica na ploči. Navedeno polje sadrži 500 SS39E hall senzora koji su međusobno spojeni na HC4067 multiplekser. Hallov senzor se temelji na efektu dvorane. Efekt dvorane se vodi načelom koje je opisano kao sljedeće: ako se vodič s izravnom strujom

[3] Izvor slike: https://hackaday.io/project/179268/logs?sort=oldest
[4] Izvor slike: https://hackaday.io/project/179268/logs?sort=oldest

postavi u magnetsko polje, tada će se u vodiču pojaviti poprečna razlika potencijala koja se još naziva i Hallov napon. Ukratko rečeno, Hallov senzor se koristi za mjerenje snage magnetskog polja.

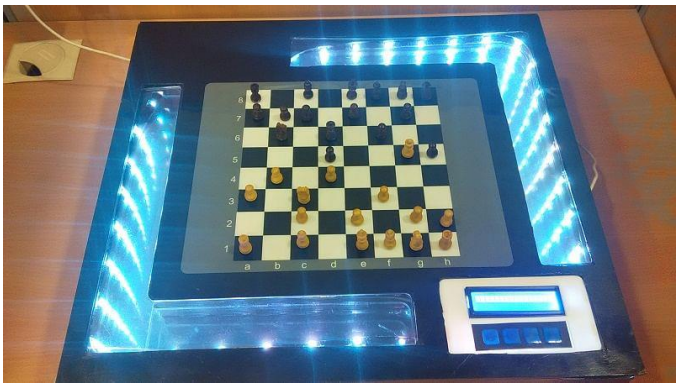


Slika 2.5. Polje hall senzora[5]

2.3. RIIDL automatizirana šahovska ploča

RIIDL šahovska ploča razvijena je od strane dva studenta iz Indije koji su došli na ideju kreiranja šahovske ploče koja ima mogućnost automatiziranog igranja protiv računala, ali ostavlja stari doživljaj fizičkog kontakta s igrom. Prvotno je ploča bila namijenjena za slabovidne osobe, ali se s vremenom opseg ciljanih korisnika proširio. Ploča ima mogućnost prepoznavanja pomaknutih figurica i njihovih pozicija pomoću potisnih senzora koji se nalaze ispod ploče. Također, ima mogućnost autonomnog pomicanja figurica od strane računala pomoću mehanizama koji se kreće u 2D prostoru [4].

[5] Izvor slike: https://cdn.hackaday.io/images/7172991622330207876.jpg



Slika 2.6. RIIDL automatizirana šahovska ploča[6]

2.4. Square-Off

Square-Off je slično rješenje kao i RIIDL. Za detekciju pokreta koristi potisne senzore, što znači da se figurica prilikom podizanja i spuštanja mora lagano pritisnuti od ploču kako bi računalo registriralo pomicanje. Također, ispod ploče se nalazi 2D CNC uređaj koji omogućava autonomno pomicanje figurica od strane računala. Na glavi CNC uređaja se nalazi elektromagnet koji se uključuje i isključuje ovisno o tome treba li pomaknuti figuru ili ne, a figurice u sebi sadrže magnete koji omogućavaju takvo ponašanje.

Square-Off ima razne opcije igranja šaha, a neke od tih su: igranje protiv računala, igranje protiv udaljenih protivnika, zapisivanje poteza, itd. Mana ovog rješenja je što nije posve kvalitetno odrađeno centraliziranje i pomicanje figurica jer se na nekim primjerima može vidjeti kako pri pomicanju jedne od figurica može doći do sudara s drugom i rušenja figurice [5].

Slika 2.7. Square-Off[7]

[6] Izvor slike: https://yourstory.com/2015/05/riidl-automated-chess-board/amp
[7] Izvor slike: https://squareoffnow.com/product/gks

2.5. TASC pametna ploča

TASC pametna ploča je razvijena od tvrtke TASC. Ploča je puštena u prodaju 1993. godine i bila je vrlo popularna za korištenje na šahovskim turnirima. Nema mogućnost autonomnog pomicanja figurica i na takav način igranja protiv računala, ali ima mogućnost prepoznavanja pozicija figurica na igraćoj ploči te se protiv računala igra tako što igrač sam pomiče protivničke figurice. Prepoznavanje pozicija omogućuju male tiskane pločice koje se nalaze u svakoj figurici te svaka figurica odašilje specifičnu frekvenciju u ovisnosti o impedanciji. Ploča je specifična po tome što na svakom polju ima LE diode koje ukazuju na to koji je potez odigran. Isto tako, LE diodama se ukazuje igraču na potez koji računalo želi da odigrate umjesto njega. Postoji mogućnost da se u računalnoj aplikaciji odabere i opcija koja će uključiti zvučnike na kojima će se čuti odigrani potezi i želje računala [6].

Slika 2.8. TASC pametna ploča[8]

[8] Izvor slike: https://www.chessprogramming.org/TASC_SmartBoard

3. REALIZACIJA SUSTAVA

3.1. Radio-frequency identification (RFID)

Radio-frequency identification ili poznatije kao RFID je tehnologija koja se zasniva na identifikaciji i/ili praćenju objekata. RFID je takozvana „not in line of sight“ tehnologija koja omogućava identifikaciju objekata u blizini koji ne moraju nužno biti vidljivi. Naime, slična je barkod tehnologiji, ali za razliku od nje RFID tehnologija može identificirati više od jednog objekta u isto vrijeme. Informacije koje sadrži objekt mogu biti mijenjane i osvježavane dok su kod barkod tehnologije one fiksne.

RFID ima veliku mogućnost primjene u gotovo svim granama ljudskih zanimanja, a posebice u transportu i logistici. Njegova primjena uvelike može olakšati razne poslove ljudima, ali isto tako može biti i prijetnja njihovoj privatnosti [7].

3.2. Komponente

U ovom potpoglavlju su navedene sve korištene komponente i njihove uloge u razvijenom sustavu.

3.2.1. RFID antene i pripadajući čitači

Antena je metalna struktura koja odašilje ili prima elektromagnetske valove koji predstavljaju neku informaciju. U ovom radu korištena je RFID antena koja prepoznaje elektromagnetske valove te dobivenu informaciju prosljeđuje RDM6300 čitaču. RDM6300 je uređaj koji dekodira informaciju koju je antena očitala. Oba uređaja rade na 125KHZ. Očitana informacija sadrži oznaku (eng. tag) po kojoj se razlikuju komponente koje emitiraju navedene elektromagnetske valove.

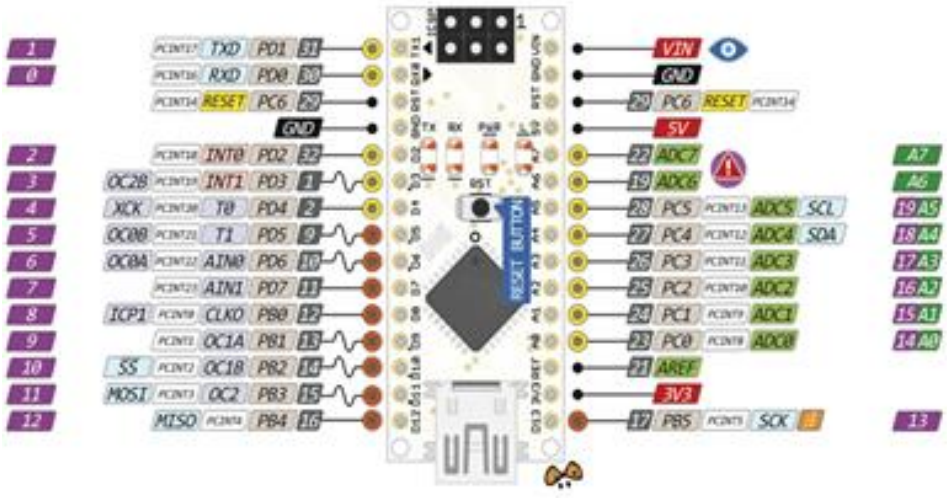
Slika 3.1. RDM6300 čitač[9]

[9] Izvor slike: https://ardubotics.eu/hr/rfid/1805-125k-rfid-karticni-citac-modul-rdm6300-uart-izlaz-arduino.html

3.2.2. Arduino Nano

Arduino Nano je mikroupravljač koji se bazira na Atmega328P procesoru. Arduino omogućava međusobnu komunikaciju raznih vrsta uređaja. U ovom radu se koristi za kontrolu i upravljanje periodičnim uključivanjima RFID čitača, provjeru stanja na igraćoj ploči te detektiranje pobjednika. Arduino radi na 16MHZ te ima 6 analogno/digitalnih pinova, 13 digitalnih i dva isključivo analogna pina. Za upravljanje A19T MOSFET-ima, a time i radom RFID čitača korišteni su digitalni pinovi D3, D4, D5, D6, D7, D8, D9, D10 i D11. Za serijsku komunikaciju s računalom koristi se digitalni pin D2 [8].

Slika 3.2. Arduino Nano[10]

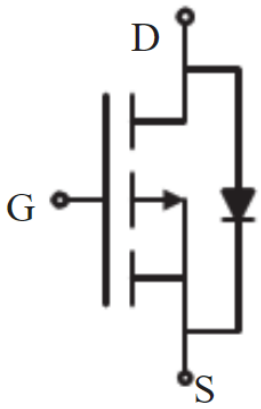


Slika 3.3. Arduino Nano raspored pinova[11]

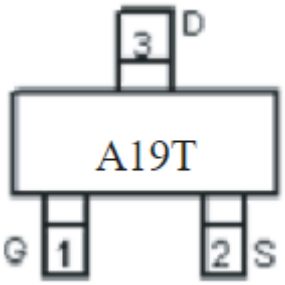
[10] Izvor slike: https://www.twinschip.com/Arduino_Nano_V3.0
[11] Izvor slike: https://components101.com/sites/default/files/component_pin/Arduino-Nano-Pinout.png

3.2.3. A19T MOSFET

A19T je P kanalni MOSFET koji se u ovom radu koristi kao sklopka (eng. switch). Namjena mu je uključiti i isključiti RFID čitače u ovisnosti o direktivama Arduina. Na *Source* nožicu mu se dovodi 5V, *Gate* nožica je spojena na Arduino, a *Drain* je izlaz MOSFET-a koji napaja RFID čitače. Arduino propusnost MOSFET-a koordinira dovođenjem nule ili jedinice na *Gate*, gdje će dovođenje nule propustiti 5V na RFID čitač.



Slika 3.4. A19T shema[12]



Slika 3.5. A19T raspored pinova[13]

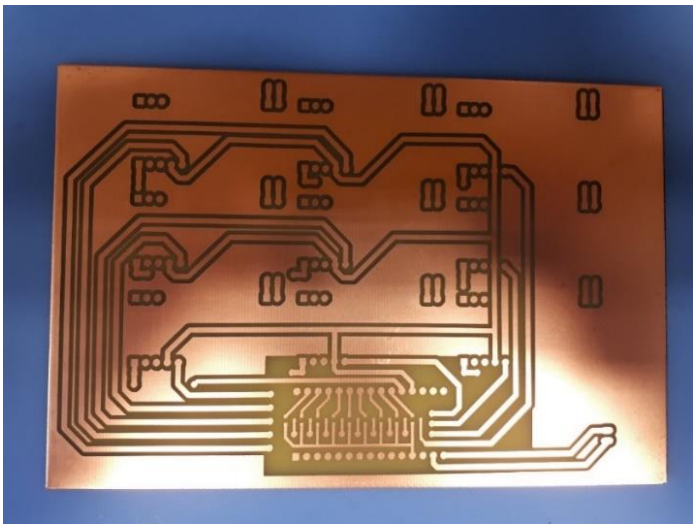
[12] Izvor slike: https://datasheetspdf.com/pdf/1401908/Rectron/A19T/1
[13] Izvor slike: https://datasheetspdf.com/pdf/1401908/Rectron/A19T/1

3.3. Maketa

3.3.1. Postupak izrade tiskane pločice

Tiskana pločica je uređena u softverskom programu *Eagle*, nakon što su funkcionalnosti komponenata provjerene na protoboardu. Namjena tiskane pločice je da objedini navedene komponente te pojednostavi međusobno spajanje komponenata. Pločica je izrađena na laseru, te je nakon izrade korištena metoda skidanja viška bakra pomoću feriklorid kiseline. Nakon skidanja viška bakra slijedi uklanjanje boje s vodova koji su preostali. Sveukupno su izrađene tri verzije tiskane pločice te u nastavku ovog poglavlja slijedi objašnjenje zašto su određena rješenja odbačena.

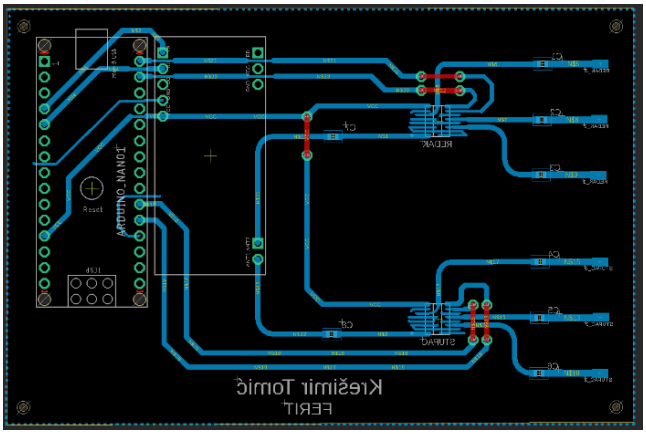
Slika 3.6. Postupak skidanja bakar pomoću feriklorid kiseline



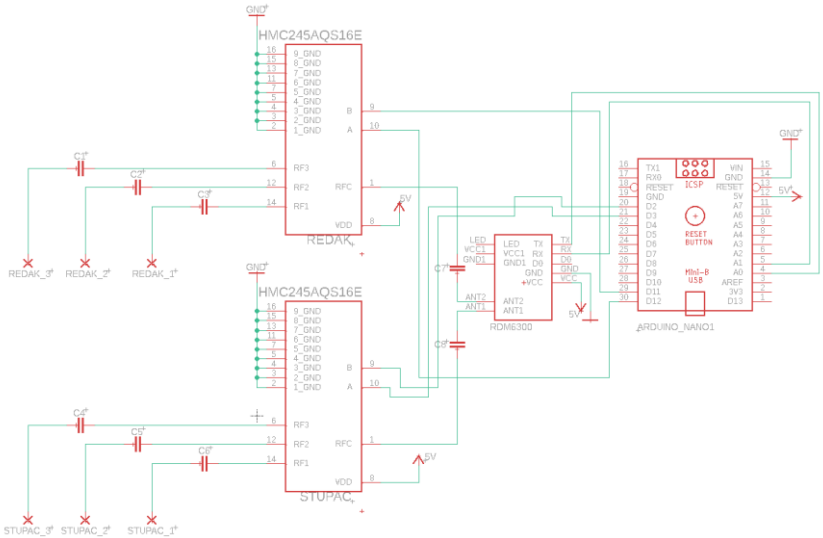
Slika 3.7. Primjer gotove tiskane pločice

3.3.2. Verzija 1 tiskane pločice: multipleksiranje RFID antena

Prvotna zamisao je bila napraviti 3x3 polje antena koje će se multipleksirati zasebno po stupcu i po retku te će se signal dovoditi na jedan RDM6300 čitač. Multipleksiranje bi vršila dva HMC245AQS16E čipa koji predstavljaju RF prekidače (eng. switch). Jedan čip bi multipleksirao stupce, drugi retke, odabirom odgovarajućih kombinacija na ulaze njih samih, a za dovođenje signala na čipove bi bio zadužen Arduino Nano. Ova tiskana pločica je dizajnirana u Eagle-u, ali je njena izrada naručena putem online trgovine. Glavni problem kod ovog rješenja je bilo krivo shvaćanje rada RFID tehnologije. Naime, u ovoj verziji je napravljen GND poligon koji je okruživao sve vodove koji su prenosili RF signale, a kasnije je uočeno da to stvara velike smetnje takvim signalima. Drugi problem ovog rješenja je što 3x3 polje antena predstavlja jednu veliku antenu koja sadrži 9 antena te dolazi do međusobnog ometanja signala i krivih očitavanja.



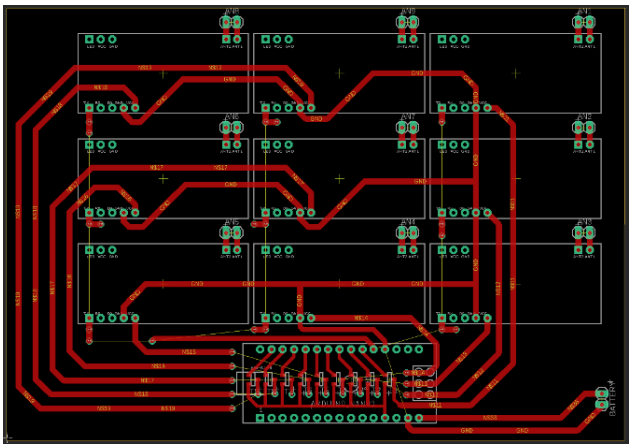
Slika 3.8. Eagle reprezentacija tiskane pločice za verziju 1



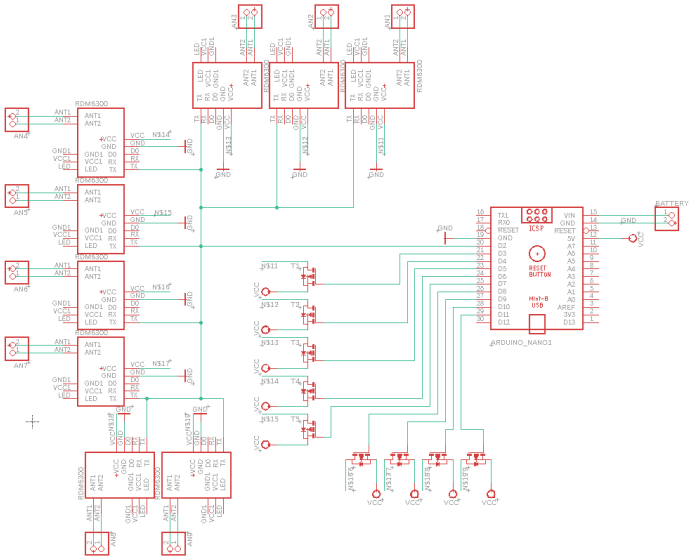
Slika 3.9. Električna shema za verziju 1

3.3.3. Verzija 2 tiskane pločice: multipleksiranje RFID čitača

Zamisao druge verzije je bila da se pojednostavi zadatak i da se izbjegne multipleksiranje polja antena na način da svaka antena posjeduje svoj RDM6300 čitač koji će u trenutku diktiranom od strane Arduina, slati očitava li ili ne očitava oznaku (eng. tag) igraće figurice. U shemi su korišteni P-kanalni MOSFET-i koji su omogućavali paljenje i gašenje čitača ovisno o direktivama Arduina. Ova verzija tiskane pločice nije radila u slučajevima kada bi se, kako je i predviđeno, RDM6300 čitači uboli u konektore koji su predviđeni za njih, ali bi radila kada bi se čitači s konektorima spojili žično. Točan razlog ovog problema nije detaljno istražen, ali se sumnja da promjena impendancije zbog duljine žice pozitivno utječe na ovaj sustav.

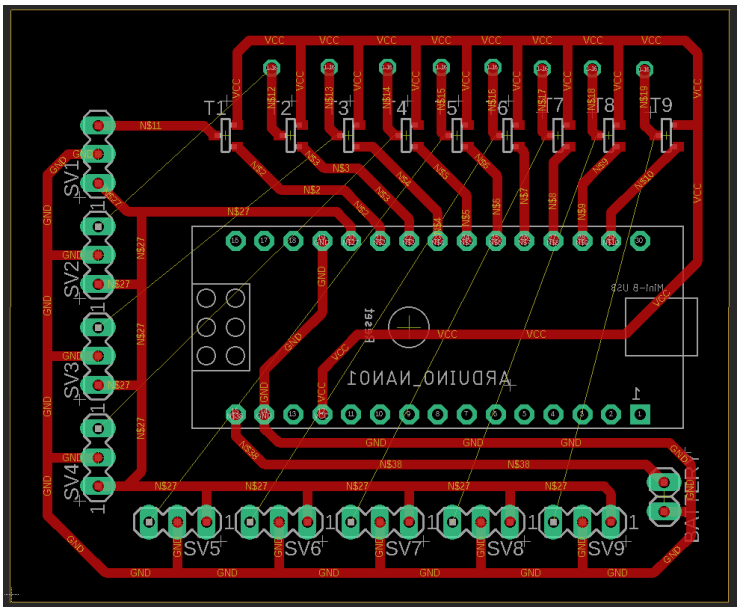


Slika 3.10. Eagle reprezentacija tiskane pločice za verziju 2

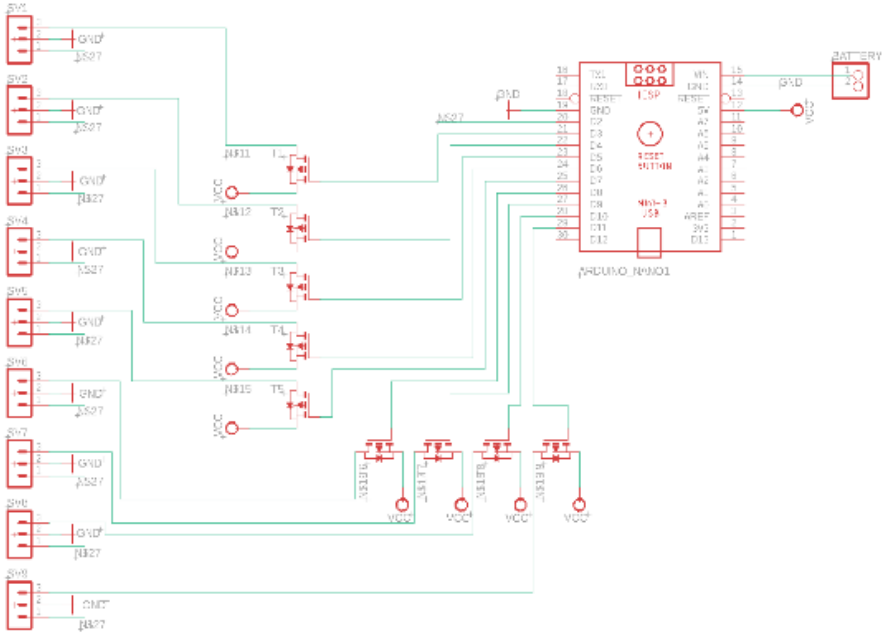


Slika 3.11. Elektricna shema za verziju 2

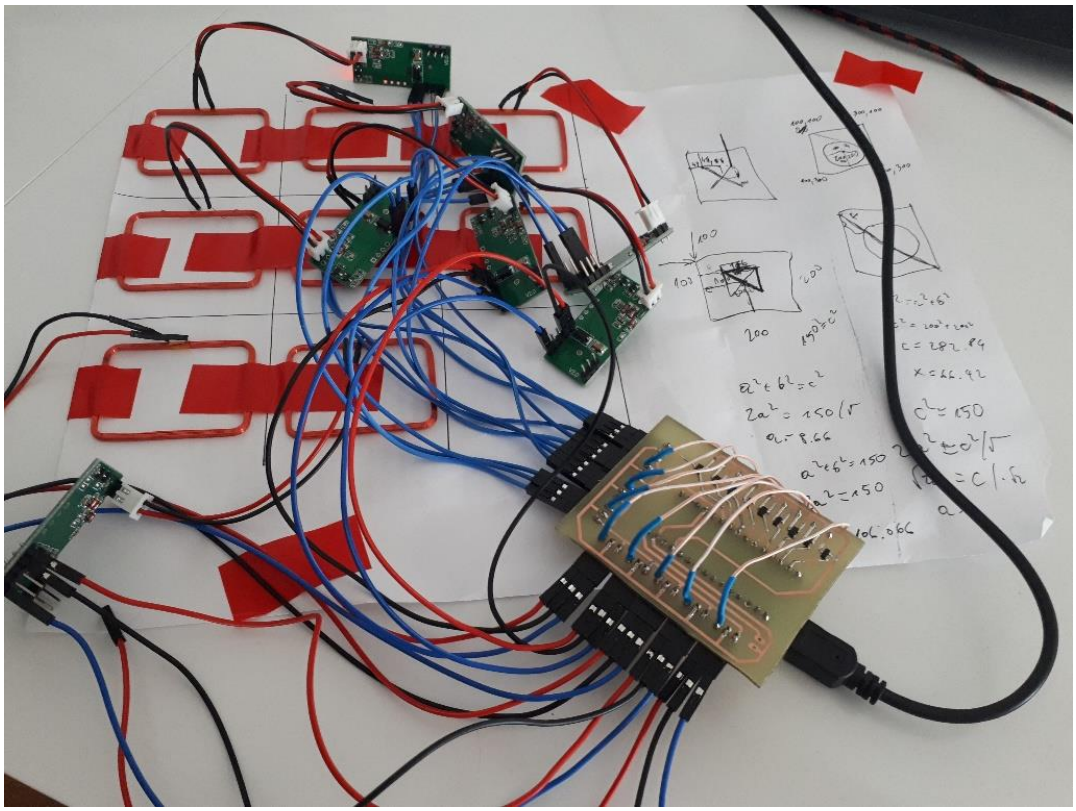
Na ovoj verziji su napravljene preinake kako bi se zaobišli prije navedeni problemi. Naime, razlika je u tome što su uklonjeni RDM3600 čitači te su postavljeni jedino konektori za njih kako bi se oni spajali žično sa sustavom, a razlog je već prije objašnjen. Ovom preinakom je drastično smanjena pločica te je ona postala i završna verzija tiskane pločice za ovaj rad.



Slika 3.12. Konačna verzije tiskane pločice



Slika 3.13. Konačna elektricna shema

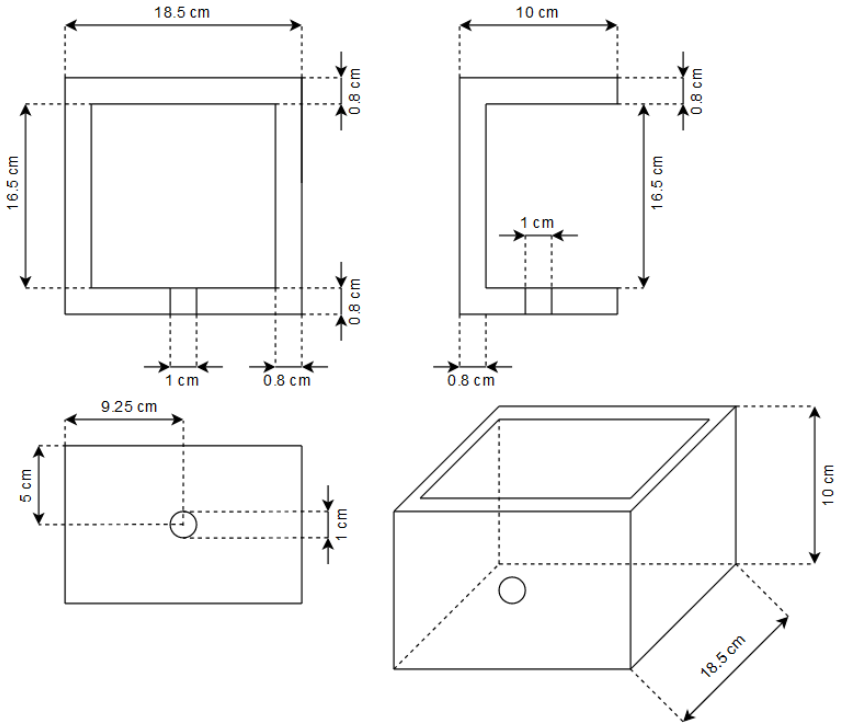


Slika 3.14. Testiranje rada konačne verzije tiskane pločice

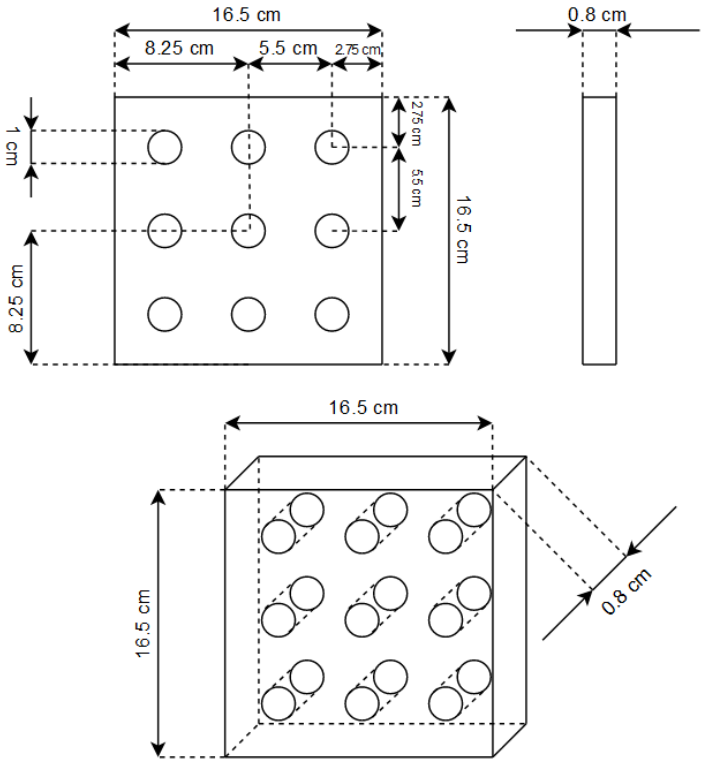
3.3.4. Kućište

Nakon što je cijeli sustav radio kako je očekivano, bilo je potrebno kreirati kućište u koje će se smjestiti. Tema diplomskog rada je RFID prepoznavanje pozicija na igraćoj ploči te je iz tih razloga ideja kućišta bila usmjerena prema nekoj vrsti kutije gdje se na jednoj od površina igra, u ovom slučaju križić-kružić, a komponente su skrivene u samoj kutiji/ploči.

Kutija je napravljena od drveta (šperploča) po dimenzijama i skicama koje se nalaze na slici ispod. Sastoji se od 3 dijela. Na slici (Slika 3.15.) se može vidjeti obična kutija s jednom rupom koja je predviđena za kabel koji napaja Arduino i vrši serijsku komunikaciju s računalom. Objektu na navedenoj slici pripada i odgovarajući poklopac koji je istih dimenzija kao i vanjski obrub kutije. Treći dio je drvena podloga koja ima 9 rupa (Slika 3.16). Podloga je predviđena da na sebi sadržava antene čije ožičenje prolazi kroz navedene rupe. Podloga se polaže u kutiju tako da antene gledaju prema dolje, to jest prema igraćoj površini. Uloga podloge je da drži antene fiksnima i što bliže igraćoj podlozi kako bi neometano mogle očitavati prisutnost igraćih figurica. Isto tako podloga odvaja antene od drugih dijelova sustava te na taj način osigurava neometano očitavanje.

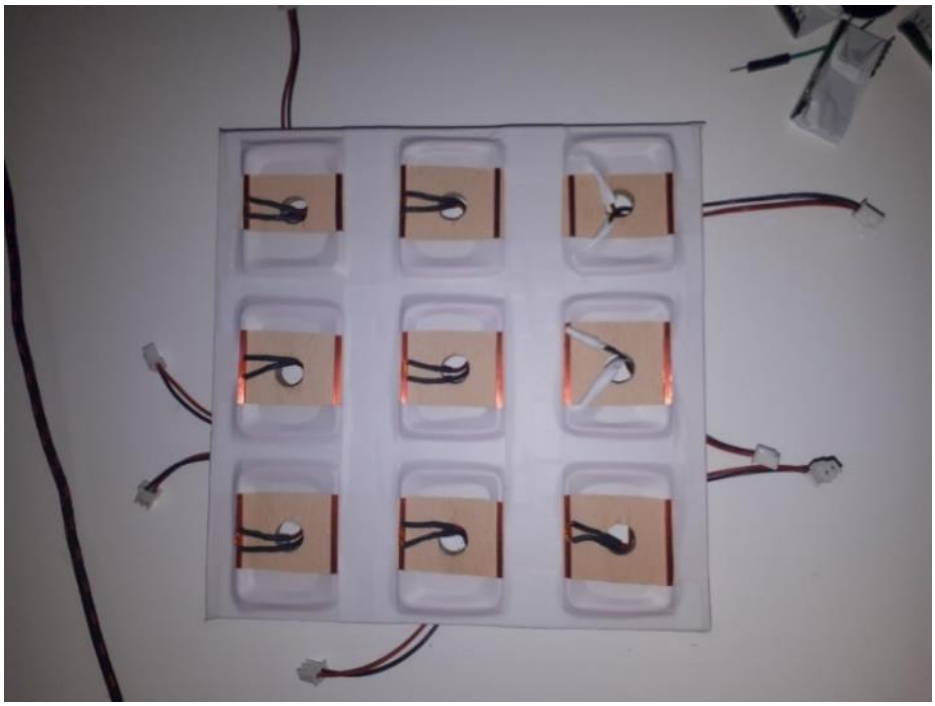


Slika 3.15. Nacrt kućišta/kutije

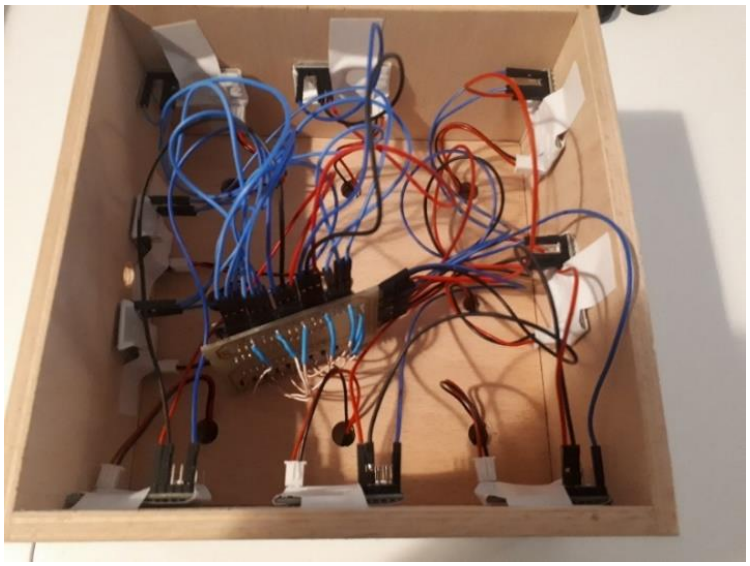


Slika 3.16. Nacrt podloge koja pridržava antene

Slika 3.17. Izrađeno kućište



Slika 3.18. Ožičene antene pričvršćene za podlogu



Slika 3.19. Kućište ispunjeno svom potrebnom elektronikom

Slika 3.20. Rad makete

3.4. Program

Program koji je pisan u svrhu ovog rada je podijeljen na dva dijela. Jedan dio je pisan u softverskom okruženju prikladnom za programiranje Arduina (Arduino IDE), a drugi dio je pisan u „Processing“ okruženju (PDE – Processing Development Environment) koje koristi Javu kao programski jezik.

3.4.1. Arduino program za prepoznavanje pozicije na ploči

Arduino IDE je programsko okruženje za programiranje Arduino mikroupravljača. Funkcionalnosti koje je bilo potrebno implementirati su sljedeće:

- Inicijalno isključiti sve RFID čitače postavljanjem logičke nule na *Gate* A19T MOSFETa
- Čitanje oznaka na igraćim figuricama pomoću rdm6300.h biblioteke
- Čitanje stanja na igraćoj ploči
- Prepoznavanje pobjednika
- Ispisivanje stanja putem serijske komunikacije
- Resetiranje stanja na ploči

Inicijalno isključivanje RFID čitača se koristi u *setup()* funkciji koja se pokreće samo pri paljenju Arduina. Na slici ispod se može vidjeti način na koji se isključuju svi RFID čitači. Polje *mosfetPins* sadrži 9 brojeva koji predstavljaju digitalne pinove na Arduinu na koje se šalje logička jedinica koja kada je dovedena na *Gate* MOSFET-a ne propušta 5V na RFID čitače te se na taj način osigurava da su oni isključeni.

```
2:      #define N 9
3:      int mosfetPins[N] = {3, 4, 5, 6, 7, 8, 9, 10, 11};
…       …
17:     for (int i = 0; i < N; i++) {
18:       pinMode(mosfetPins[i], OUTPUT);
19:       digitalWrite(mosfetPins[i], HIGH);
20:     }
```

Slika 3.21. Inicijalno isključivanje RDM6300 čitača – *main()* funkcija

Čitanje oznaka (eng. tag) se odvija u funkciji *loop()* koja se neprestano pokreće sve dok je Arduino uključen. U *loop()* funkciji se nalazi funkcija *readBoard()* čija je funkcionalnost uključiti određeni RFID čitač, provjeriti postoji li oznaka za čitanje u blizini RFID antene te ako postoji onda je potrebno odrediti o kojoj je figurici riječ. U ovom radu figurice su 'X' ili 'O', a njihove oznake su zapisane u poljima X_IDs i O_IDs, a ako ne postoji oznaka u blizini čitača ili je pročitana oznaka nepoznata u zapis se postavlja znak '-' koji predstavlja da je određeno polje prazno. Stanje na ploči se zapisuje u polje *xOx* koje je veličine od 12 elemenata. Prvih devet elemenata predstavlja stanje na ploči. Na desetom mjestu je oznaka koja predstavlja pobjednika, a

jedanaesto i dvanaesto mjesto označavaju mjesto na ploči na kojem su spojene tri iste figurice u niz. Nakon što je čitanje oznaka završeno potrebno je isključiti upaljeni RFID čitač. Funkcija *readBoard()* utječe samo na prvih 9 elemenata *xOx* polja. Navedeni programski kod se može vidjeti ispod.

```
1:      void readBoard() {
2:        for (int i = 0; i < N; i++) {
3:          digitalWrite(mosfetPins[i], LOW);
4:          delay(150);
5:          uint32_t tag = 0;
6:          bool isFound = false;
7:          while (rdm6300.isAvailable()) {
8:            tag = rdm6300.get_tag_id();
9:            if (isFound == false) {
10:             for (int j = 0; j < 5 ; j++) {
11:               if (tag == X_IDs[j]) {
12:                 xOx[i] = 'X';
13:                 isFound = true;
14:                 break;
15:               }
16:               else {
17:                 xOx[i] = '-';
18:                 isFound = false;
19:               }
20:             }
21:           }
…             …
35:         }
36:         digitalWrite(mosfetPins[i], HIGH);
37:         delay(50);
38:       }
39:     }
```

Slika 3.22. Čitanje oznaka igraćih figurica – *readBoard()* funkcija

Prepoznavanje pobjednika je omogućeno u funkciji *checkStatus()* koja se također poziva u *loop()* funkciji odmah nakon funkcije *readBoard()* koja pruža najnovije izvješće stanja na ploči. Funkcija *checkStatus()* se sastoji od mnoštva if-elseif grana u kojima se provjerava zapis sadržan u *xOx* polju. U igri križić-kružić postoji 8 načina na koje jedan igrač može pobijediti što znači da je sveukupno 16 tipova zapisa koji će rezultirati prepoznavanjem jednog od pobjednika. U funkciji se prvo provjerava je li 'X' pobjednik na način da se provjere 3 retka, 3 stupca, a potom glavna i sporedna dijagonala. Zatim, se na isti način provjerava je li 'O' pobjednik. Ako se utvrdi pobjednik na deseto mjesto u *xOx* polju se postavlja oznaka pobjednika, a na jedanaesto i dvanaesto mjesto se postavlja lokacija od tri iste povezane figurice. Ukoliko je polje s indeksima 0-8 ispunjeno oznakama 'X' ili 'O', a ostala mjesta u polju sadržavaju oznaku '-' tada zaključujemo da je partija završena te da je rezultat neriješen.

```
1:      void checkStatus() {
2:        if (xOx[0] == 'X' && xOx[1] == 'X' && xOx[2] == 'X'){
3:          endGame = true;
4:          winner = 'X';
5:          numberOfRowOrColumn = '1';
6:          rowOrColumn = 'r';
7:        }
8:        else if (xOx[3] == 'X' && xOx[4] == 'X' && xOx[5] == 'X'){
9:          endGame = true;
10:         winner = 'X';
11:         numberOfRowOrColumn = '2';
12:         rowOrColumn = 'r';
13:       }
…       …
50:       else if (xOx[0] == 'O' && xOx[1] == 'O' && xOx[2] == 'O'){
51:         endGame = true;
52:         winner = 'O';
53:         numberOfRowOrColumn = '1';
54:         rowOrColumn = 'r';
55:       }
56:       else if (xOx[3] == 'O' && xOx[4] == 'O' && xOx[5] == 'O'){
57:         endGame = true;
58:         winner = 'O';
59:         numberOfRowOrColumn = '2';
60:         rowOrColumn = 'r';
61:       }
…       …
98:       xOx[9] = winner;
99:       xOx[10] = numberOfRowOrColumn;
100:      xOx[11] = rowOrColumn;
101:      if (xOx[0] != '-' && xOx[1] != '-' && …) {
102:        tie = true;
103:      }
104:    }
```

Slika 3.23. Prepoznavanje pobjednika – *checkStatus()* funkcija

Ispisivanje *xOx* polja putem serijske komunikacije je potrebno zbog prosljeđivanja informacija o stanju ploče prema „Processing“ aplikaciji koja će na osnovi tog zapisa vizualno prikazivati stanje na ploči.

```
41:     Serial.println(xOx);
```

Slika 3.24. Ispisivanje *xOx* polja – *main()* funkcija

Resetiranje igraće ploče se izvodi pomoću funkcije *resetBoard()* koja se također poziva u *loop()* funkciji nakon provjere ima li pobjednika. Ukoliko nema pobjednika *resetBoard()* postavlja zapis u *xOx* polju na niz od 12 znakova koji označavaju da je polje prazno ('-') te postavlja varijablu pobjednika i varijable koje lociraju mjesto pobjede na '-'.

```
1:      void resetBoard() {
2:        for (int i = 0; i < 12; i++) {
3:          xOx[i] = '-';
4:        }
5:        winner = '-';
6:        numberOfRowOrColumn = '-';
7:        rowOrColumn = '-';
8:        endGame = false;
9:        tie = false;
10:     }
```

Slika 3.25. Resetiranje igraće ploče – *resetBoard ()* funkcija

3.4.2. Programski alat „Processing“

„Processing“ je softversko okruženje koje u ovom radu omogućava vizualni prikaz podataka koji se šalju serijskom komunikacijom između Arduina i računala. „Processing“ program sadrži dvije glavne funkcije, slično kao i Arduino. Te funkcije su *setup()* i *draw()*. Funkcija *setup()* ima istu ulogu kao što takva funkcija ima i u Arduino IDE okruženju, a to je da se pokreće samo jednom na početku pokretanja programa i u njoj se pokreću određene inicijalizacije. Funkcija *draw()* služi za kreiranje korisničkog sučelja u odnosu na podatke koje obrađujemo u njoj.

Funkcionalnosti koje je bilo potrebno implementirati u ovom programskom okruženju su sljedeće:

- Inicijalizacija i uspostavljanje serijske komunikacije
- Dohvaćanje podataka koji se prenose serijskom komunikacijom
- Crtanje igraćeg okvira
- Provjera pobjednika
- Crtanje figurica na igraćoj ploči
- Resetiranje igraće ploče

U funkciji *setup()* se inicijalizira veličina korisničkog sučelja, uspostavlja serijska komunikacija s brzinom prijenosa od 115200 baud-a te se poziva funkcija *reset()*. Funkcija *reset()* inicijalizira *xOx* varijablu na početno stanje („----------“), postavlja font, dodaje gumb za resetiranje igre i resetira varijablu koja označava postojanje pobjednika. Sve navedeno iz funkcije

reset() je potrebno postaviti u *setup()*, ali je bilo potrebno i odvojiti u posebnu funkciju iz potrebe za resetiranjem igre nakon završetka pa je *reset()* funkcija smještena u *setup()*.

```
11:     void setup(){
12:       size(800, 800);
13:       mySerial = new Serial(this, "COM5", 115200);
14:       reset();
15:     }
16:
17:     void reset (){
18:       xOx = "------------";
19:       nl = 10;
20:       isThereAWinner = false;
21:
22:       background(40);
23:       strokeWeight(4);
24:
25:       font = createFont("calibri light bold",30);
26:       textSize(32);
27:       text("TicTacToe", 320, 50);
28:
29:       myControl = new ControlP5(this);
30:       myControl.addButton("Restart")
31:                .setPosition(300, 715)
32:                .setSize(200, 50)
33:                .setFont(font);
34:     }
```

Slika 3.26. Inicijalizacija i uspostavljanje serijske komunikacije – *setup()* i *reset()* funkcija

Dohvaćanje podataka koji se prenose putem serijske komunikacije se odrađuje u *draw()* funkciji. Podatci se čitaju sve do znaka za novi red. Nakon pročitanih podataka slijedi dio koda za crtanje i translatiranje križić-kružić tablice kako bi se tablica našla u sredini korisničkog sučelja. Zatim se pozivaju 4 funkcije, jedna za detekciju pobjednika, a ostale tri za crtanje figurica na ploči.

```
36:     void draw(){
37:        while(mySerial.available()>0){
38:           xOx = mySerial.readStringUntil(nl);
39:           println(xOx);
40:        }
41:
42:        pushMatrix();
43:        translate(100, 100);
44:        for(int x=0;x<3;x++){
45:          for(int y=0;y<3;y++){
46:            rect(200*x,200*y,200,200);
47:          }
48:        }
49:        popMatrix();
50:
51:        checkTheWinner(xOx);
52:        drawFirstRow(xOx);
53:        drawSecondRow(xOx);
54:        drawThirdRow(xOx);
55:     }
```

Slika 3.27. Dohvaćanje podataka i kreiranje igraće ploče – *draw()* funkcija

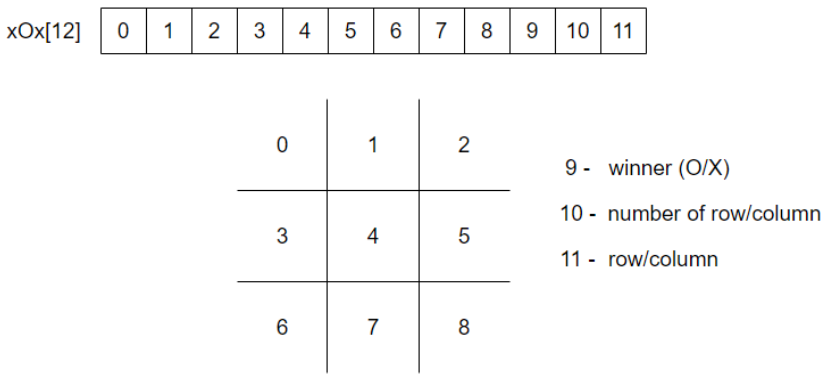
Pobjednik se provjerava pozivom funkcije *checkTheWinner()*. Funkcija prima podatak koji je prenesen s Arduina i provjerava deseto mjesto u polju jer ono, kao što je već prije u radu navedeno, sadržava oznaku pobjednika. Nakon utvrđivanja pobjednika na ekran se ispisuje figurica koja je pobijedila te ovisno o poziciji pobjedničkih figurica jednom linijom se spajaju 3 uzastopno povezane pobjedničke figurice. Pozicija pobjednika se čita s jedanaestog i dvanaestog mjesta u *xOx* polju.

```
57:     void checkTheWinner(String xOx){
58:       if(xOx.charAt(9) == 'X'){
59:         isThereAWinner = true;
60:         text("Winner is: X", 100, 750);
61:       }
62:       else if(xOx.charAt(9) == 'O'){
63:         isThereAWinner = true;
64:         text("Winner is: O", 100, 750);
65:       }
66:       if(isThereAWinner == true){
67:         if(xOx.charAt(10) == '1'){
68:           if(xOx.charAt(11) == 'r'){
69:             line(100, 200, 700, 200);
70:           }
71:           if(xOx.charAt(11) == 'c'){
72:             line(200, 100, 200, 700);
73:           }
74:         }
75:         if(xOx.charAt(10) == '2'){
76:           if(xOx.charAt(11) == 'r'){
77:             line(100, 400, 700, 400);
78:           }
79:           if(xOx.charAt(11) == 'c'){
80:             line(400, 100, 400, 700);
81:           }
82:         }
```

```
83:         if(xOx.charAt(10) == '3'){
84:           if(xOx.charAt(11) == 'r'){
85:             line(100, 600, 700, 600);
86:           }
87:           if(xOx.charAt(11) == 'c'){
88:             line(600, 100, 600, 700);
89:           }
90:         }
91:         if(xOx.charAt(11) == 'd'){
92:           if(xOx.charAt(10) == 'm'){
93:             line(100,100,700,700);
94:           }
95:           if(xOx.charAt(10) == 's'){
96:             line(700, 100, 100, 700);
97:           }
98:         }
99:       }
100:      if(xOx.charAt(4) == 't'){
102:          text("It's a tie!", 100, 750);
103:      }
104:    }
```

Slika 3.28. Ispitivanje pobjednika – *checkTheWinner()* funkcija

Crtanje figurica na igraćoj ploči se izvodi pomoću 3 funkcije. Svaka funkcija ima zadaću iscrtati jedan red križić-kružić tablice. Prolaskom kroz petlju od 3 iteracije provjeravaju se oznake koje su sadržane na određenim mjestima u *xOx* polju te na temelju oznake se crta 'O' ili 'X'. Indeks za prvu petlju počinje od 0 što nam govori da je riječ o prvom redu, za drugu petlju indeks je 3 što nam ukazuje na 2 red, a za treću petlju početni indeks je 6 što podrazumijeva 3 red.



Slika 3.29. Pozicije na igraćoj ploči u ovisnosti o *xOx* polju

```
105:    void drawFirstRow(String xOx){
106:      for(int i=0;i<3;i++){
107:        if(xOx.charAt(i)=='X'){
108:          line(147+(i*200),147,((i+1)*200)+53,253);
109:          line(((i+1)*200)+53,147,(i*200)+147,253);
110:        }
111:        if(xOx.charAt(i)=='O'){
112:          ellipse(((i+1)*200),200,130,130);
113:        }
114:      }
115:    }
116:
118:    void drawSecondRow(String xOx){
118:      for(int i=3;i<6;i++){
119:        if(xOx.charAt(i)=='X'){
120:          line(147+((i-3)*200),347,53+((i-2)*200),453);
121:          line(53+((i-2)*200),347,147+((i-3)*200),453);
122:        }
123:        if(xOx.charAt(i)=='O'){
124:          ellipse(((i-2)*200),400,130,130);
125:        }
126:      }
127:    }
128:
129:    void drawThirdRow(String xOx){
130:      for(int i=6;i<9;i++){
131:        if(xOx.charAt(i)=='X'){
132:          line(147+((i-6)*200),547,53+((i-5)*200),653);
133:          line(53+((i-5)*200),547,147+((i-6)*200),653);
134:        }
135:        if(xOx.charAt(i)=='O'){
136:          ellipse(((i-5)*200),600,130,130);
137:        }
138:      }
139:    }
```

Slika 3.30. Crtanje figurica po igraćoj ploči

Resetiranje igre se postiže pozivom funkcije *Restart()* čija je dužnost serijskom komunikacijom poslati logičku jedinicu prema razvojnom okruženju Arduina, a Arduino će se povodom toga resetirati te pokrenuti igru ispočetka. Također, resetirat će se i „Processing“ program pozivom *reset()* funkcije.

```
140:    void Restart(){
141:      mySerial.write('1');
142:      reset();
143:    }
```

Slika 3.31. Resetiranje Arduina i nacrtane igraće ploče – funkcija *Restart()*

4. ZAKLJUČAK

U ovom radu izrađena je igraća ploča s mogućnošću prepoznavanja pozicija figurica pomoću RFID tehnologije. RFID je tehnologija koja se zasniva na identifikaciji i/ili praćenju objekata. Prepoznavanje pozicija se odvija pomoću slijedećih komponenti: Arduino Nano, A19T MOSFET i RDM6300. Sve su komponente međusobno spojene pomoću tiskane pločice koja je izrađena u programskom paketu Eagle. Arduino Nano koordinira paljenjem i gašenjem svih RDM6300 čitača pomoću A19T MOSFET-a. Dovođenjem logičke '0' na A19T MOSFET pali se jedan RDM6300, a dovođenjem logičke '1' gasi se taj isti RDM6300 čitač. Uključeni RDM6300 dekodira informaciju primljenu od pripadajuće antene te se dobivena informacija koristi za prepoznavanje ima li na promatranom polju jedna od figurica. Iscrtavanje stanja na igraćoj ploči se odvija u programskom alatu „Processing“ koji serijskom komunikacijom s Arduinom dohvaća potrebne podatke za kreiranje ploče i zauzetih pozicija na istoj od strane odgovarajućih figurica. Ovaj rad je zanimljiv jer je RFID tehnologija u usponu i traže se rješenja koja su optimalna kod RFID prepoznavanja pozicija na igraćim pločama kao što je šah. No, trenutačno još ne postoji idealno rješenje, ali se pretpostavlja da bi multipleksiranje RFID antena bilo najbolje moguće rješenje za ovaj tip problema. Ovaj rad je uspješno izrađen multipleksiranjem RFID čitača u drugoj verziji tiskane pločice.

LITERATURA

[1] Digital Game Technology, Electronic Boards, Digital Game Technology, dostupno na: http://www.digitalgametechnology.com/index.php/products/electronic-boards [30.06.2021.]

[2] Chess programming wiki, DGT Board, Chess programming wiki, 2018., dostupno na: https://www.chessprogramming.org/DGT_Board [30.06.2021.]

[3] C. Lopez, Automatic Chessboard, Hackaday, 2021., dostupno na: https://hackaday.io/project/179268/logs?sort=oldest [02.07.2021.]

[4] J. Mehta, Backed by RiiDL, hardware enthusiasts from Mumbai build an automated chess board, Yourstory, 2015., dostupno na: https://yourstory.com/2015/05/riidl-automated-chess-board/amp [02.07.2021.]

[5] A. James, What's inside one of those magic chess boards?, BoingBoing, 2018., dostupno na: https://boingboing.net/2018/05/15/whats-inside-one-of-those-ma.html [02.07.2021.]

[6] Chess programming wiki, TASC, Chess programming wiki, 2018., dostupno na: https://www.chessprogramming.org/TASC [02.07.2021.]

[7] V. D. Hunt; A. Puglia; M. Puglia, RFID: A Guide to Radio Frequency Identification, Wiley, 2006.

[8] J. Blum, Exploring Arduino, Wiley, 2002.

SAŽETAK

Naslov: RFID prepoznavanje pozicije za igre na ploči

U ovom radu izrađena je igraća ploča za križić-kružić koja ima mogućnost RFID prepoznavanja pozicija na ploči. Rad sadrži dio u kojem su navedena već postojeća rješenja za ovaj tip problema i tehnologije koje su korištene za rješavanje istih. Kasnije se u radu može vidjeti na koji je način realizirana igraća ploča za križić-kružić. Korišteni programski paketi su: Eagle, Arduino IDE i „Processing“, a pod realizacijom sustava spada: izrada tiskane pločice, izrada prikladnog kućišta, međusobno spajanje komponenata, programiranje Arduina i kreiranje GUI-a. Moguće je unaprijediti rješenje ovog sustava multipleksiranjem RFID antena te primijeniti to rješenje na kompleksnije sustave kao što je šah.

Ključne riječi: RFID, prepoznavanje pozicija, križić-kružić, šah, Arduino, tiskana pločica, mikroupravljač, RDM6300.

ABSTRACT

Title: Board games based on RFID position recognition

In this paper is created a board game for TicTacToe that has the capability of RFID position recognition. The paper contains a section listing existing solutions for this type of problem and the technologies used to solve them. Later in the paper, it can be seen how the TicTacToe board game was realized. The software packages that are used: Eagle, Arduino IDE, and „Processing“, and the realization of the board includes: making a PCB, making a suitable housing, interconnecting all components, programming Arduino, and creating GUI. It is possible to improve the solution of this system by multiplexing RFID antennas and apply this solution to more complex systems such as chess.

Keywords: RFID, position recognition, TicTacToe, Chess, Arduino, PCB, microcontroller, RDM6300.

ŽIVOTOPIS

Krešimir Tomić rođen 24. veljače 1998. godine u Zagrebu. Osnovnu školu završio 2012. godine u Zagrebu. Nakon toga upisujem Tehničku školu Ruđera Boškovića u Zagrebu. Nakon završetka srednje škole, 2016. godine upisujem preddiplomski studij Računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Po završetku prediplomskog studija, 2019. godine upisujem diplomski studij, modul Računalno inženjerstvo.