



Osnovna šola Stranje

RAZISKOVALNA NALOGA

AVTOMATSKO RAZVRŠČANJE VLAKOV

iz področja aplikativni inovacijski predlogi in projekti

marec 2021

avtor

Andraž Kočar, 9. a

mentor

Andrej Kočar, mag. org. inf.

Kazalo

CILJ NALOGE / RAZISKOVALNA HIPOTEZA	4
1 UVOD.....	5
1. 1 MODULNA ŽELEZNICA	5
1. 2 TESTNI MODUL.....	6
2 TESTNI MODUL.....	7
2. 1 IDEJNI NAČRT OŽIČENJA IN OZNAČEVANJA	8
2. 2 PRIKLOP KRETNICE IN SEMAFORJA.....	10
2. 3 PRIKLOP SENZORJEV	11
3 PROGRAMSKI DEL.....	12
3. 1 GLAVNI DELI PROGRAMSKE KODE	13
4 UGOTOVITVE.....	16
5 LITERATURA, VIRI	17

Povzetek

V nalogi je predstavljena idejna rešitev (modela) krmiljenja avtomatskega razvrščanja vlakov na železniške perone. Ko sem pred časom v Ljubljani na glavni železniški postaji poslušal napovednik, kateri vlak prihaja na kateri peron, se mi je porodila ideja, ali bi bilo izvedljivo narediti računalniški »sortirnik«, ki bi usmerjal vlake na perone. To sem si zadal kot cilj raziskovalne naloge. Ker se sicer ukvarjam z malimi železnicami (modulna železnica), se mi je zdela rešitev kot na dlani. Izdelal sem samostojen modul, ki deluje na podlagi računalniškega razvrščanja vlakov. Opravil sem nekaj meritev. Na podlagi teh sem preveril svojo hipotezo. Hipoteza se je potrdila.

Cilj naloge / raziskovalna hipoteza

V raziskovalni nalogi bom poskusil prikazati uspešno delovanje avtomatskega razvrščanja vlakov na postaji z uporabo računalnika. Za potrebe modela bom uporabil enostavno krmilje *Arduino Mega*, pri pravem vlaku bi bilo zagotovo potrebno izbrati drugo, ustrežnejšo elektroniko. Hipotezo bom potrdil empirično, z izdelavo prototipa in testiranjem. Če bodo rezultati dela na modelu uspešni, pomeni, da lahko računalnik samodejno razvršča vlake na perone. Z ustreznim krmiljem bi bila možna tudi implementacija v realnem svetu.

1 Uvod

Že odkar pomnim, sem imel veliko fascinacijo nad železnico. Tudi za mojega očeta pravijo, da je »zastrupljen« s tem in da je uspešno to prenesel name. Jaz to razumem kot kompliment, saj sem se z odkrivanjem železniških čarov naučil dosti o modeliranju, mehaniki, elektriki in kasneje elektroniki.

Tako imam doma bogato zbirko železnice v HO merilu (1:87). Za potrebe raziskovalne naloge sva z mentorjem dorekla, kaj bo potrebno storiti in narediti. Za potrebe naloge sva z mentorjem izdelala nadpovprečno velik modul, na katerem bova naredila postajo, opremila z elektroniko in testirala hipotezo.

1. 1 Modulna železnica

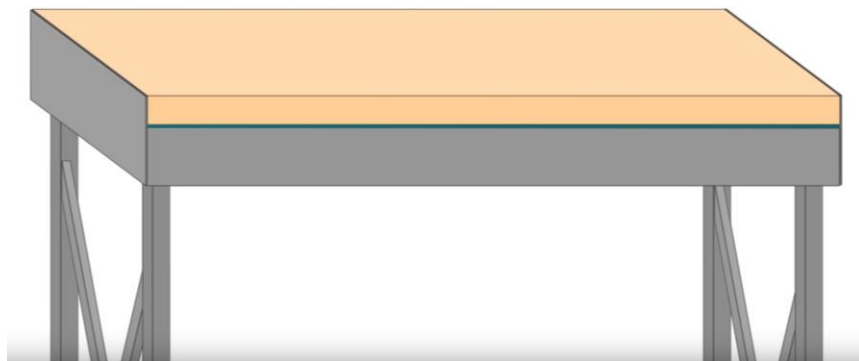
Ljubitelji malih železnic so običajno rezervirali en prostor v hiši, kjer so razvijali svojo dejavnost – postavili so maketo železnice. Maketa, kljub temu, da je v razmerju 1:87, zahteva veliko prostora, takoj ko želiš narediti kaj več kot samo krog, po katerem vozi vlak.



Slika 1: primer - osnova za gradnjo klasične makete

Običajno je prostor problem, zato so si ljubitelji izmislili *t. i. modulno* železnico. Gre za gradnjo makete, lahko rečemo tudi diorame, po delih (modulih), kjer se gradnja izvaja po vnaprej določenih pravilih, katere se morajo držati vsi člani (sicer modul ni združljiv). Tako vsak ljubitelj izdelava nekaj svojih modulov, ki jih lahko odpelje na srečanje ali prireditve ter se pridruži ostalim. Gre dejansko za izdelavo namenske »mize«, katere potem združujemo skupaj.

Svoj modul priključi zraven in tako nastane vedno nova ali pa ista dolga maketa, kakršne doma ne bi mogel imeti, predvsem pa je stopnja realizma precej večja v primerjavi z maketo, ker vlak vozi zgolj v krogu.

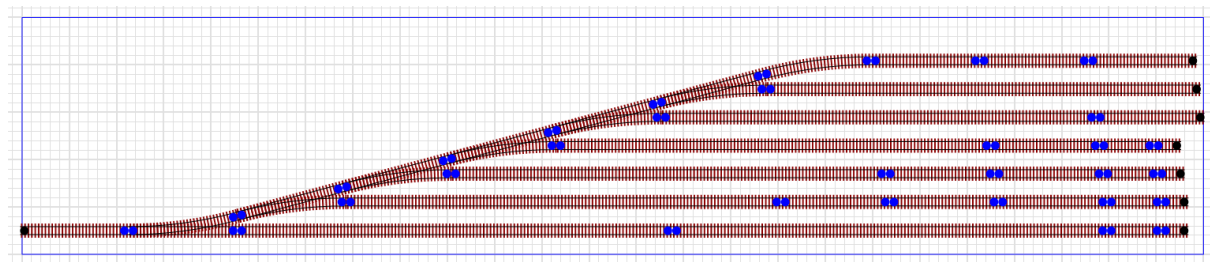


Slika 2: primer modula dimenzije 50cm x 120cm

Module običajno sestavimo iz lesa: noge masiven les, za površine pa uporabimo vezano ploščo različnih debelin. Na vrhu tipično pokrijemo s 3 ali 5 cm stirodur ploščo, katero lažje predelujemo ravnine v neravnine (teren makete).

1. 2 Testni modul

Kot sem že omenil, sem za potrebe raziskovalne naloge izdelal daljši modul, da sem lahko nanj namestil široko železniško postajo. Tako je na modulu en uvodni tir iz katerega je razširjena proga na sedem peronov.



Slika 3: računalniški načrt proge za testni modul narisano z Railroad Professional

Kadar se načrtuje modelno železnico, se vedno prej naredi ustrezen načrt. Včasih se je to izdelovalo ročno, danes imamo na uporabo računalniška (CAD) podprta orodja za risanje. Za risanje sem uporabil orodje *RailRoad professional 3.1.1*. Orodje podpira tire vseh znanih proizvajalcev. Kakor je shema narisana v orodju, tako bo tudi v praksi. Največji problem so različne krivine, kjer se pri prostem oblikovanju lahko pojavijo zgolj nekaj milimetrске vrzeli, ki jih ni mogoče kar tako zapolniti. Če za spojitve dveh tirov manjka recimo 5 mm, bi prisilna sklenitev povzročila delno razklenitev nekje drugje.

Z orodjem lahko simuliram in večkrat virtualno zelo natančno postavim ter spremenim celoten koncept, še pred samim nakupom tirov. Orodje seveda omogoča tudi izpis kosovnice vseh tirov in porabljenega materiala, izračuna ceno »projekta« in se poveže s spletno trgovino, kjer je možen nakup.

2 Testni modul

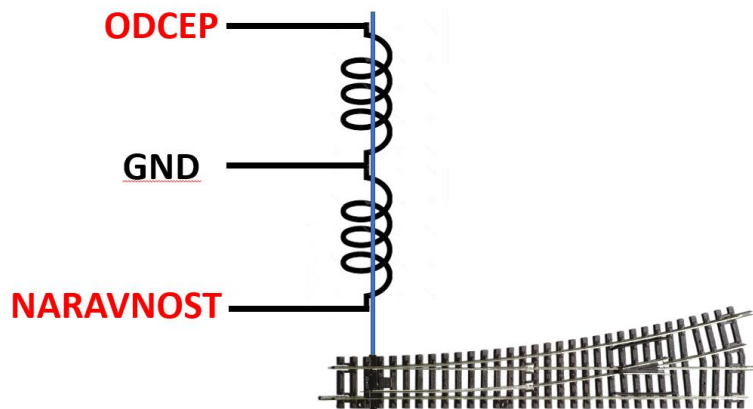
Po izdelanem načrtu sem se lotil gradnje. Uporabil sem tirnice brez podlage, zato sem se moral poigrati še s posipom proge.



Slika 4: primer - testni modul, pogled vzdolž

Pri gradnji sem moral uporabiti električne kretnice, da jih bom lahko krmilil električno. Mikrokrmilnik Arduino sem spoznal pri izbirnem predmetu Robotika v tehniki, kjer sem se tudi naučil osnovnega programiranja.

Mehanizem kretnic je zgrajen zelo različno. Nekateri proizvajalci uporabljajo servo motor, drugi pa zgolj elektromagnet, ki predstavlja kretnico. V mojem primeru gre za elektromagnet. Kretnice imajo tri priključne žice. Srednja je »-« pol, med skrajnima pa preklapljamo: ali odcep ali naravnost.



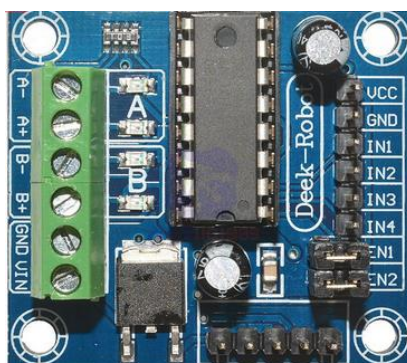
Slika 5: pogon kretnice

Na modulu je torej sedem peronov oz. šest kretnic. Vsaka kretnica potrebuje dva digitalna izhoda. Ker bom v nadaljevanju priključil še semaforje, potrebujem še dodatnih sedem oz. štirinajst izhodov.

Proti koncu vsakega perona je nameščen IR senzor, ki zazna prisotnost vlaka; torej potrebujem vsaj sedem vhodov.

Pri pregledu števila vhodov in izhodov sem ugotovil, da Arduino Uno ne bo ustrezna izbira, ker ima premajhno število vhodov in izhodov. Zato sem moral uporabiti Arduino Mega.

Ker pa je signal iz mikrokrmilnika prešibak, sem za krmiljenje uporabil H-mostiček, kakršnega smo pri robotiki uporabljali za krmiljenje elektromotorčka.

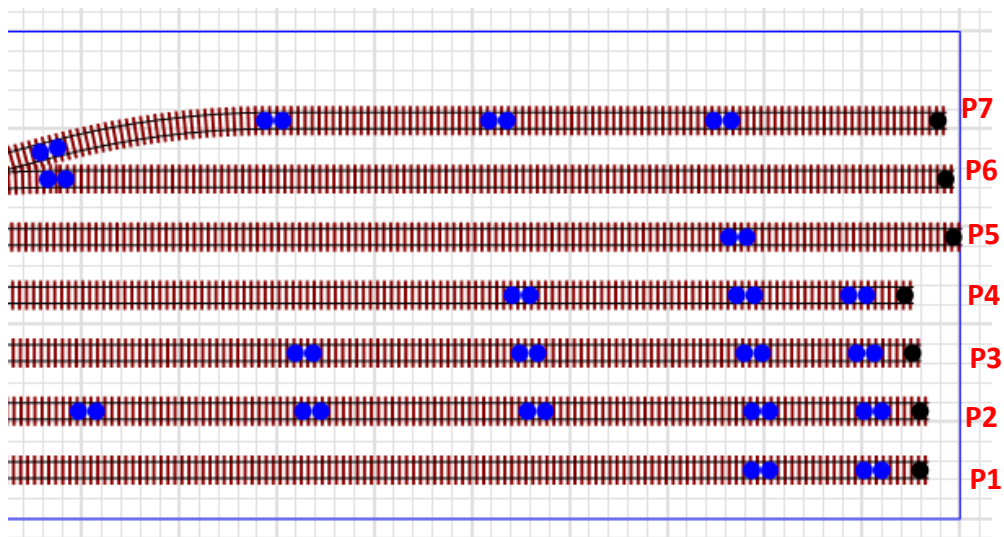


Slika 6: modul H-mostička

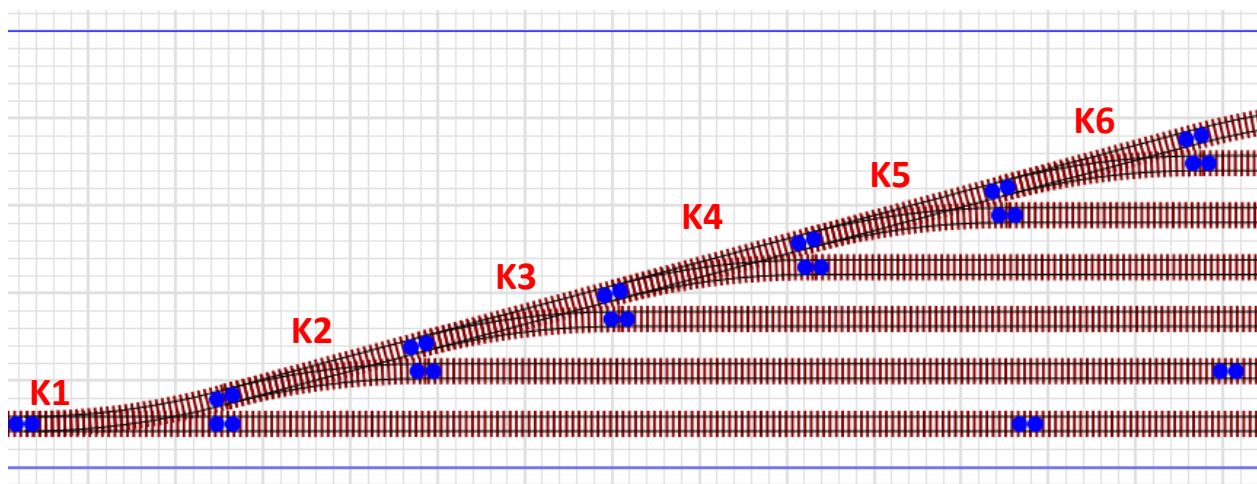
Tako bom A izhoda uporabil za krmiljenje kretnice, B pa za krmiljenje semaforja. Izhod A bo za krmiljenje preklon kretnice aktiven za 200ms (pulz), medtem ko bo izhod B ostal v dotičnem stanju. Do tega časa sem prišel s preizkušanjem.

2. 1 Idejni načrt ožičenja in označevanja

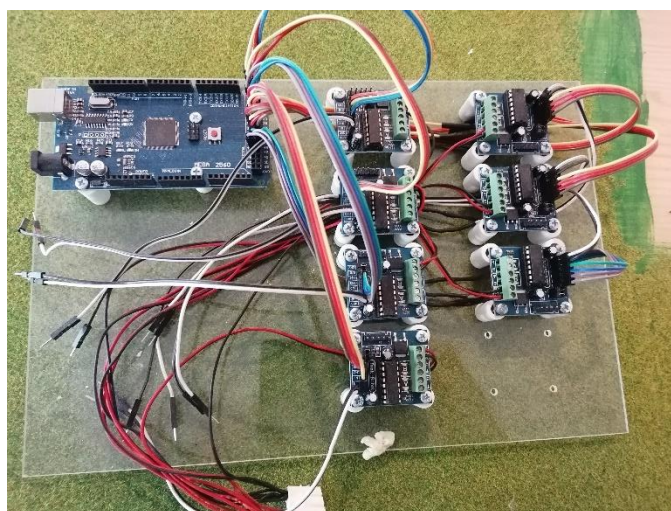
Kretnice in senzorje, ki sem jih vgradil proti koncu perona, je potrebno dobro označiti, saj bo na podlagi tega sledila izdelava programa.



Ravno tako sem označil kretnice z oznakami K1 do K6.



Potem sem izdelal tabelo vhodov in izhodov za kretnice, semaforje in senzorje. Na podlagi te tabele sem začel izdelovati program. Pri programiranju mi je pomagal mentor, ker sam še nimam dovolj izkušenj za izdelavo celotnega programa. Izhode za kretnice in semaforje sem uporabil od vključno 22 naprej (robna stran Arduina Mega).



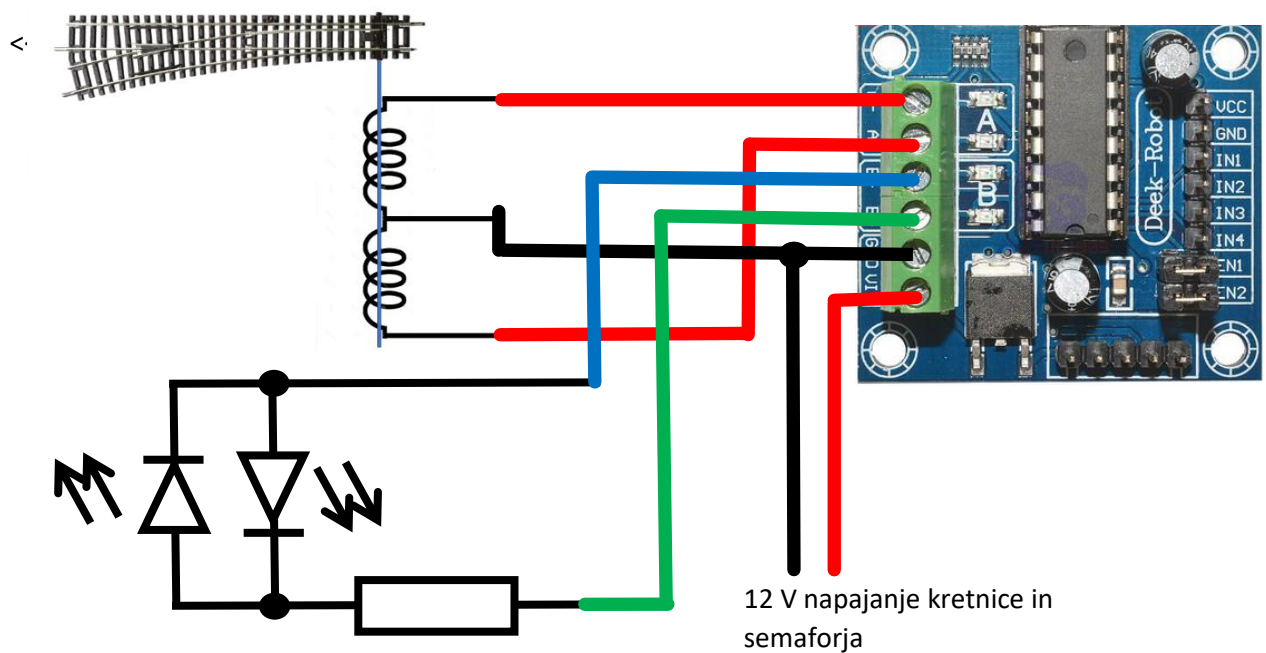
Slika 7: Arduino Mega s H-mostički v nastajanju

št. izhoda	h-bridge	opis
22	#1	kretnica 1
23		semafor 1
24		
25		
26	#2	kretnica 2
27		semafor 2
28		
29		
30	#3	kretnica 3
31		semafor 3
32		
33		
34	#4	kretnica 4
35		semafor 4
36		
37		
38	#5	kretnica 5
39		semafor 5
40		
41		
42	#6	kretnica 6
43		semafor 6
44		
45		

Slika 8: tabela priklopov na Arduino

2. 2 Priklop kretnice in semaforja

Najprej sem moral testirati krmiljenje kretnice. Na H-mostiček sem priključil kretnico in testiral čas preklopa. Pri napajanju 12V se kretnica z 200ms impulzom preizkušeno preklopi.



Na drugi izhod mostička pa pride priključen par svetlečih diod, ki so povezane obratno. S tem dosežem, da po dveh žicah lahko krmilim dve luči. Gre za talne železniške semaforje, kjer sveti ali rdeča luč ali bela. Zaradi kretnic mora biti napajanje 12V, kar pomeni, je potrebno dodati predupor pri svetlečih diodah. Uporabil sem upor 1k ohm.

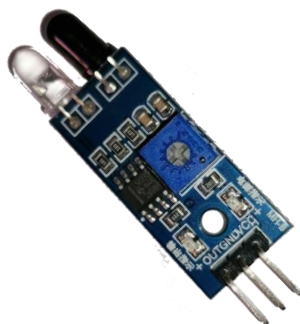
Tako potrebujem šest H-mostičkov.

2. 3 Priklop senzorjev

Za preverjanje ali je vlak na peronu ali ne sem izbral brezstične senzorje Pololu, ki zaznajo oviro na pet cm razdalje. Najprej sem imel namen uporabiti klasične Arduino IR senzorje, vendar zaradi svoje velikosti nekako ne pašejo na maketo; vsekakor pa jih bom uporabil za samo testiranje. Vseh sedem senzorjev bom namestil in njihove izhode priključil – kot so označeni peroni – na vhode A1 do A7. A0 sem namenoma izpustil.



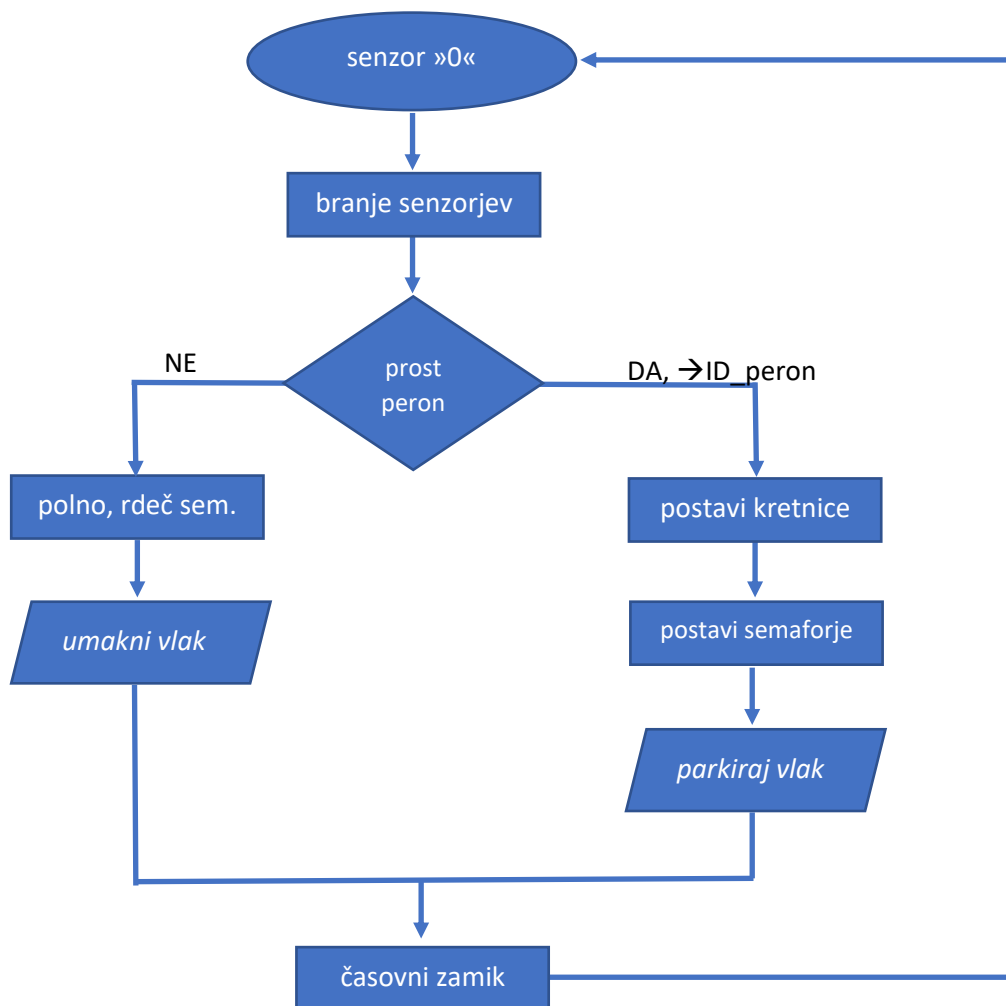
Slika 9: Pololu senzor GP2Y0D80SZ0F



Slika 10: Arduino IR senzor za testiranje

3 Programski del

Za lažje razumevanje in drobljenje problema sem si narisal diagram poteka, potem pa sem šel v izdelavo programa.



Na začetku, predno vlak pripelje na sam modul, sproži senzor »0«. Ta senzor sproži branje senzorjev zasedenosti peronov. Pri modelu sem predpostavil, da se dolžine vlakovnih kompozicij ujemajo z dolžino peronov. Stanje senzorjev se vpiše v tabelo. Preverim celo tabelo. V primeru, da ni prostega perona, semafor ne spusti vlaka naprej – torej se mora vlak odpeljati nazaj. V primeru, da je prost peron, se najprej postavijo kretnice do perona, potem pa se prižgejo še semaforji. Ko vlak zapelje na dodeljeni peron, senzor na peronu zazna njegovo prisotnost. Ob naslednjem branju senzorjev bo senzor dal stanje »zasedeno«.

3. 1 Glavni deli programske kode

Ker problematika uporablja veliko izhodov in vhodov, sem določene logične sklope »razbil« v posamezne dele programa. Pri programiranju mi brez mentorjeve pomoči ne bi uspelo.

```
int preveri() {
    int zacasna=-1;
    for (int peron=1; peron<8; peron++){
        if (tabela[peron] == 0) {
            zacasna = peron;
            break;
        }
    }
    return zacasna;
}
```

Slika 11: funkcija preveri

V funkciji *preveri* preberem vrednosti iz tabele *tabela* od 1 do 7. Takoj, ko naletim na vrednost »0«, izstopim iz zanke in vrnem ta položaj – torej številko perona – kot prosto mesto. Spremenljivka *tabela* je deklarirana globalno – torej jo vidijo vse funkcije v programu.

```
void beri_senzorje() {
    tabela[1]=digitalRead(A1);
    tabela[2]=digitalRead(A2);
    tabela[3]=digitalRead(A3);
    tabela[4]=digitalRead(A4);
    tabela[5]=digitalRead(A5);
    tabela[6]=digitalRead(A6);
    tabela[7]=digitalRead(A7);
}
```

Slika 12: funkcija za branje senzorjev

V vsako polje tabele preberem stanje senzorja, 0 ali 1.

```

void kretnica(int ID_kretnica, int smer){
    // ID_kretnica - št. kretnice
    // smer: 0-naravnost 1-odcep
    // impulz: 200ms za preklon
    int impulz = 200;

    if (ID_kretnica == 1){
        if (smer == 0){
            digitalWrite(22, HIGH);
            delay(impulz);
            digitalWrite(22, LOW);
        }
        else {
            digitalWrite(23, HIGH);
            delay(impulz);
            digitalWrite(23, LOW);
        }
    }
    if (ID_kretnica == 2){
        if (smer == 0){
            digitalWrite(26, HIGH);
            delay(impulz);
            digitalWrite(26, LOW);
        }
        else {
            digitalWrite(27, HIGH);
            delay(impulz);
            digitalWrite(27, LOW);
        }
    }
}

void semafor(int ID_semafor, int luc){
    // id_semafor - št. semaforja
    // luc: 0-rdeča 1-bela(GO GO GO!)
    if (ID_semafor == 1) {
        if (luc == 0){
            //rdeca
            digitalWrite(24,HIGH);
            digitalWrite(25,LOW);
        }
        else{
            //bela
            digitalWrite(24,LOW);
            digitalWrite(25,HIGH);
        }
    }
    if (ID_semafor == 2) {
        if (luc == 0){
            //rdeca
            digitalWrite(28,HIGH);
            digitalWrite(29,LOW);
        }
        else{
            //bela
            digitalWrite(28,LOW);
            digitalWrite(29,HIGH);
        }
    }
}

```

Slika 13: začetek funkcije za premik kretnic (levo) in semaforje (desno)

Za premik kretnic je potrebno podati podatek katera kretnica naj se premakne in smer. S smerjo povemo ali je smer naravnost ali gre v odcep. Na podoben način je narejena funkcija *semafor* (glej slika 13). Tudi tu podam podatek kateri semafor naj se prižge in kako naj se prižge. V funkciji – tako za premik kretnic kot semaforja – pa sem iz tabele črpal podatke na katere izhode je kaj priključeno.

```

void setup() {

    int temp=-1;
    int casovni_zamik = 3000;
    cakaj_na_zacetni_senzor(); //čakam na senzor A0 - prihajajoči vlak
    beri_senzorje();           //berem senzorje
    temp = preveri();           //iščem prvi prosti peron, dobim št.perona
    semafor_all_red();          //vse semaforje dam v rdečo luč
    if (temp == 1){             // če je peron 1
        //prvi peron je prost
        kretnica(1,0);          //postavim kretnico naravnost
        semafor(1,1);           //prižgem bel semafor
    }
    if (temp == 2){             //če je prost peron 2
        //drugi peron je prost
        kretnica(1,1);          //dam prvo kretnico v odcep
        kretnica(2,1);          //in tudi drugo v odcep
        semafor(2,1);           //prižgem bel semafor
    }
    if (temp == 3){             //če je prost peron 3
        //tretji peron je prost
        kretnica(1,1);          // prva kretnica v odcep
        kretnica(2,0);          // druga naravnost
        kretnica(3,1);          // tretja v odcep
        semafor(3,1);           // prižgem bel semafor
    }
    if (temp == 4){
        //četrty peron je prost
        kretnica(1,1);          // prva kretnica v odcep
        kretnica(2,0);          // druga naravnost
        kretnica(3,0);          // tretja naravnost
        kretnica(4,1);          // četrta v odcep
        semafor(3,1);           // prižgem bel semafor
    }
}

```

Slika 14: začetek glavnega programa

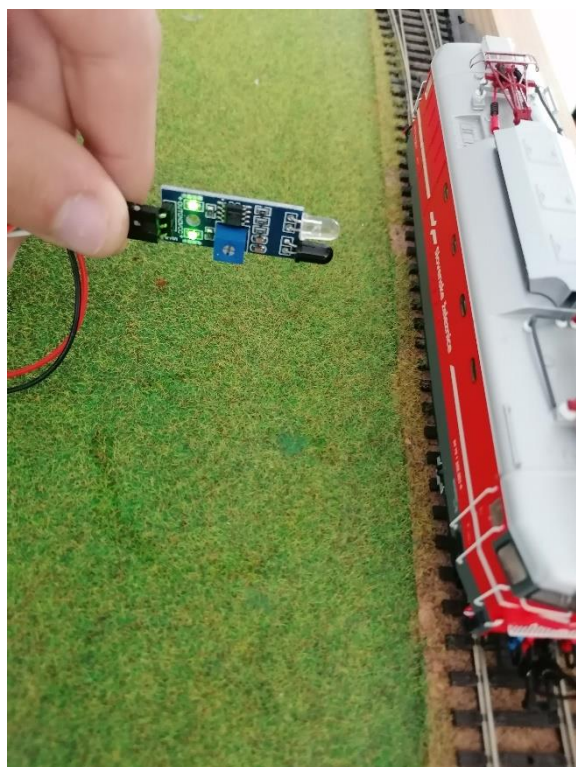
4 Ugotovitve

Zaradi obilice testov v šoli nisem uspel modula dokončati v celoti, zato sem na Arduino priključil zgolj dve krenici in tri senzorje perona za testne potrebe.



Slika 15: Končni senzor

Naredil sem nekaj uspešnih poskusov, vendar sem kaj kmalu naletel na nepredvidene situacije. Če vlaka nisem peljal čisto do konca perona, ni bil v dometu senzorja, zato je ob naslednjem razvrščanju pokazalo peron kot prost. Če se je vlak ustavil ali premaknil tako, da je bil izven dometa senzorja, ravno tako ni delovala prepoznava zasedenosti perona.



Problem detekcije vlaka bom moral narediti drugače. Predvsem bi bilo potrebno uporabiti večje število senzorjev ter na podlagi tega tudi izvesti ugotavljanje dolžine vlaka.

Drug problem, katerega bi izpostavil in ga moja rešitev ne predvideva, je dolžina vlaka. Glede na informacijo o dolžini kompozicije bi bilo potrebno izbrati ustrezen peron. Ker gre za modelno železnico, bi bilo merjenje dolžine kompozicije težje izvedljivo.

5 Zaključek

V fazi testiranja sem prišel do zaključka, da računalniško razvrščanje vlakov v proste perone dejansko deluje. Na mojem modulu bom v nadaljevanju to razvrščanje izpopolnil in dodal še ročno izbiro perona.

Vsekakor pa izvedba ni čisto enostavna, kot je morda izgledalo sprva. Ob določenih predpostavkah (dolžina vlaka ni pomembna) model deluje popolnoma zadovoljivo. Vendar bi bilo v realnem svetu potrebnih večje število tipal in dodatnih varnostnih mehanizmov, ki bi zagotavljali varno »odločanje« kateri peron izbrati.

Ob upoštevanju predpostavk lahko hipotezo potrdim – računalniško razvrščanje je mogoče.

6 Literatura, viri

- <http://www.vlaki.jesenice.net/fremo.html>, marec 2021
- <https://www.arduino.cc>, primeri, marec 2021
- https://create.arduino.cc/projecthub/Tech_build/simple-automated-model-railway-layout-arduino-controlled-aed192, marec 2021