UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

ZAKLJUČNA NALOGA

MARTIN MLAKAR

UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

ZAKLJUČNA NALOGA

Model avtomatskih vrat z IR senzorjem

Martin Mlakar

Mentor: doc. dr. Tadej Tuma

LJUBLJANA, 2000

Kazalo

1 Povzetek ... 5
2 Uvod ... 7
3 Zgradba sistema ... 8
3.1 Maketa ... 10
3.1.1 Ogrodje ... 10
3.1.2 Vrata in vodila ... 10
3.1.3 Motor ... 11
3.1.4 Prenosni mehanizem ... 11
3.1.5 Senzor ... 12
3.1.6 Tehnični načrt makete ... 12
3.2 Krmilnik makete ... 15
3.2.1 Krmilnik motorja ... 15
3.2.2 Tipkovnica ... 17
3.2.3 Vezje za prižig LED ... 18
3.2.4 Detektor senzorja ... 18
3.2.5 Tiskano vezje ... 19
3.2.6 I/O konektor (Povezava makete z mikrokrmilniškim sistemom) ... 21
3.3 Programski del ... 22
3.3.1 Delovanje programa ... 22
3.3.2 Programiranje v realnem času ... 22
3.3.3 Opravilo MOT ... 23
3.3.4 Opravilo SENZOR ... 26
3.3.5 Opravilo LOGIC ... 27
3.3.6 Opravilo KBD ... 28
3.3.7 Opravilo LED ... 29
3.3.8 Opravilo DEFAULT ... 30
3.3.9 Nastavitev paralelnega vmesnika PIA6821 ... 31
3.3.10 Glavni program (MAIN) ... 31
3.3.11 Podprogram KBDCHR ... 33
3.3.12 Podprogram TIPKI ... 33
4 Sklep ... 35
5 Zahvala ... 36
6 Priloga: Izpis celotnega programa ... 37
7 Seznam uporabljenih virov ... 44
8 Izjava ... 45

Kazalo slik

Slika št. 1: Blok shema sistema, str. 8
Slika št. 2: Slika makete avtomatskih vrat, str. 9
Slika št. 3: Motor, str. 11
Slika št. 4: Senzor, str. 12
Slika št. 5: Naris makete, str. 13
Slika št. 6: Tloris makete, str. 14
Slika št. 7: Električna shema D/A pretvornika z izhodno stopnjo, str. 15
Slika št. 8: Električna shema regulatorja motorja, str. 16
Slika št. 9: Električna shema tipkovnice, str. 17
Slika št. 10: Električna shema vezja za prižiganje svetlečih diod, str. 18
Slika št. 11: Detektor senzorja, str. 19
Slika št. 12: Tiskano vezje, str. 20
Slika št. 13: Razpored elementov tiskane ploščice, str. 20
Slika št. 14: I/O konektor, str. 21
Slika št. 15a: Diagram poteka opravila MOT, str. 24
Slika št. 15b: Diagram poteka opravila MOT, str. 25
Slika št. 16: Diagram poteka opravila SENZOR, str. 26
Slika št. 17: Diagram poteka opravila LOGIC, str. 27
Slika št. 18: Diagram poteka opravila KBD, str. 28
Slika št. 19: Diagram poteka opravila LED, str. 29
Slika št. 20: Diagram poteka opravila DEFAULT, str. 30
Slika št. 21: Diagram poteka glavnega programa, str. 32
Slika št. 22: Zbirniška koda glavnega programa, str. 33
Slika št. 23: Diagram poteka rutine TIPKI, str. 34

Seznam uporabljenih kratic

IC	Integred Circuit
LED	Light Emiting Diode
PIA	Parallel Interface Adapter
GND	Ground
ABS	Akril Butadien Syrol
D/A	Digital / Analog
I/O	Input / Output
IDT	Integrated Development Tools

1 Povzetek

Mikrokrmilniški sistem HC11 je zelo univerzalen in ga lahko v kombinaciji z razvojnim orodjem uporabljamo za izdelavo najrazličnejših aplikacij. Ko pa je aplikacija oziroma program razvit, lahko sistem deluje tudi samostojno. Problem pa nastane, kadar bi se študent rad naučil pisanja zbirniškega programa. Seveda lahko uporabi mikrokrmilniški sistem kot osnovno orodje, vendar ga mora priključiti na nek konkreten primer, kjer bo lahko, v realnem času, opazoval delovanje njegove aplikacije. Vendar pa v realnem svetu ne obstaja veliko možnosti kamor bi lahko priključili mikrokrmilniški sistem in se tam učili programiranja. Prav zato rabimo maketo na kateri bomo lahko opazovali obnašanje programa. V mojem primeru smo se odločili za maketo avtomatskih vrat.

Cilj naloge je izdelati maketo in tiskano vezje, katerega bomo priključili direktno na mikrokrmilniški sistem. Maketa mora biti ustreznih dimenzij, da jo lahko postavimo na delovno površino, kjer opazujemo njeno obnašanje. Tiskano vezje pa mora biti sestavljeno tako, da bomo lahko programsko krmilili vse elemente makete. Naloga je sestavljena tudi iz kompletnega programa, ki krmili maketo.

Delovanje programa je podobno obnašanju realnih avtomatskih vrat. Z aplikacijo krmilimo vse sestavne dele makete: motor, senzor, tipke ter svetleče diode. Kadar so vrata v normalnem delovanju, program čaka na sprožitev senzorja. Ko se le to zgodi, se vrata odprejo, počakajo štiri sekunde in se zaprejo nazaj. Seveda čakajo odprta štiri sekunde tudi od trenutka, ko smo sprožili senzor ob že odprtih vratih. Kadar pa sprožimo senzor v trenutku, ko se vrata zapirajo, program takoj odpre vrata in počaka štiri sekunde. Imamo tudi možnost nastavitve dveh položajev: skozi zaprta vrata ali skozi odprta vrata. Le to storimo z dvema tipkama, ki se nahajata na tiskani plošči. Kadar pritisnemo tipko za vedno odprta vrata, se vrata odprejo, zasveti rdeča svetleča dioda, senzor pa ne vpliva na obnašanja vrat. Kadar pa pritisnemo tipko za vedno zaprta vrata, se le ta zaprejo, ali ostanejo zaprta, prižge se zelena svetleča dioda, senzor pa prav tako ne vpliva na delovanje vrat. Sistem vrnemo v normalno delovanje z ponovnim pritiskom na isto tipko, oziroma je v takem delovanju kadar ne sveti nobena svetleča dioda.

Podrobneje je delovanje in uporaba naprave opisana v jedru naloge, ki je smiselno razdeljeno na več poglavij. Najprej najdemo opis makete in njenih sestavnih delov. Ti so:

- ogrodje,
- vrata in vodila,
- motor,
- prenosni mehanizem in
- senzor.

Nato sledi opis tiskanega vezja, ki ga sestavljajo:

- analogni krmilnik motorja,
- vezje za krmiljenje tipk ob vedno zaprtih in odprtih vratih, kamor spadajo tudi svetleče diode,
- ter nenazadnje še detektor senzorja.

Po opisu strojniškega dela sledi še podroben pregled celotnega programa, ki krmili maketo in je napisan v zbirniškem jeziku. Izpis celotnega programa pa najdemo kot priloga v šestem poglavju.

2 Uvod

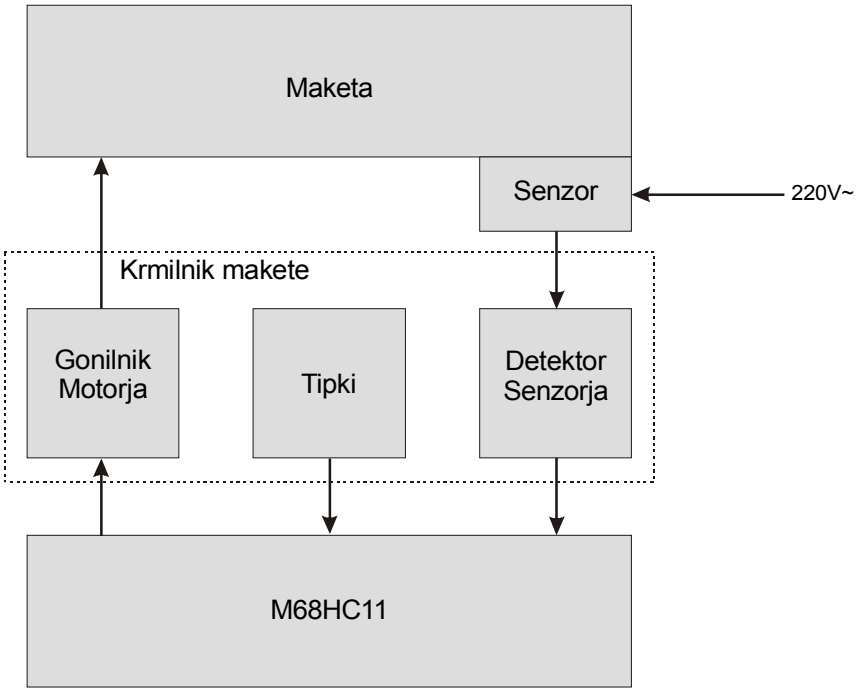
Danes smo priča zelo hitremu razvoju vseh vrst naprav iz skoraj vseh področij našega življenja. Tako ima že skoraj vsaka malo bolj zahtevna naprava, ki izkorišča elektronska vezja, vgrajen tak ali drugačen mikrokrmilniški sistem. Zato se njihovo področje uporabe vedno bolj razširja in tako so se mikrokrnilniki v zadnjih nekaj desetletjih razvijali zelo hitro. Prvi so bili relativno enostavni in počasni ter majhnih zmogljivosti. Novejši pa so kompleksni, izredno hitri in zelo zmogljivi. Vseeno pa imajo mnogo stvari še vedno skupnih. Čeprav so novejši mikrokrmilniki hitrejši, imajo močnejšo procesorsko enoto in veliko več vgrajenih enot, pa so načini izvajanja programov in razvoja aplikacij podobni. Če dobro poznamo lastnosti enostavnejših mikrokrmilnikov, lahko to v večini primerov uporabimo tudi pri zahtevnejših mikrokrmilnikih.

Dobro znanje programiranja mikrokrmilnikov je prav zato vsak dan pomembnejše. Programiranja pa se najlažje naučimo na konkretnem primeru. Iz teh idej sledi cilj te naloge, izdelati maketo, katero bomo lahko programirali preko mikrokrmilniškega sistema HC11. Maketa avtomatskih vrat je izdelana tako, da je dovolj praktična ter primernih dimenzij, za uporabo na delovni površini.

Cel sistem, ki služi pedagoškim namenom, je sestavljen iz strojnega dela. Ta zajema mehansko sestavljeno maketo z vsemi pripadajočimi deli, vrati, prenosnim mehanizmom, in tiskanim vezjem, ki ga priključimo na mikrokrmilniški sistem in preko njega programsko krmilimo maketo. Sistem pa sestavlja tudi aplikacija v kateri študent najde vse odgovore, ki nastanejo ob težavah programiranja makete.

3 Zgradba sistema

V tem poglavju je opisana celotna zgradba in delovanje sistema avtomatskih vrat. Vrata so sestavljena iz treh delov: makete vrat, krmilnika makete ter mikrokrmilniškega sistema HC11 [2]. Povezava vseh treh komponent prikazuje slika številka ena.

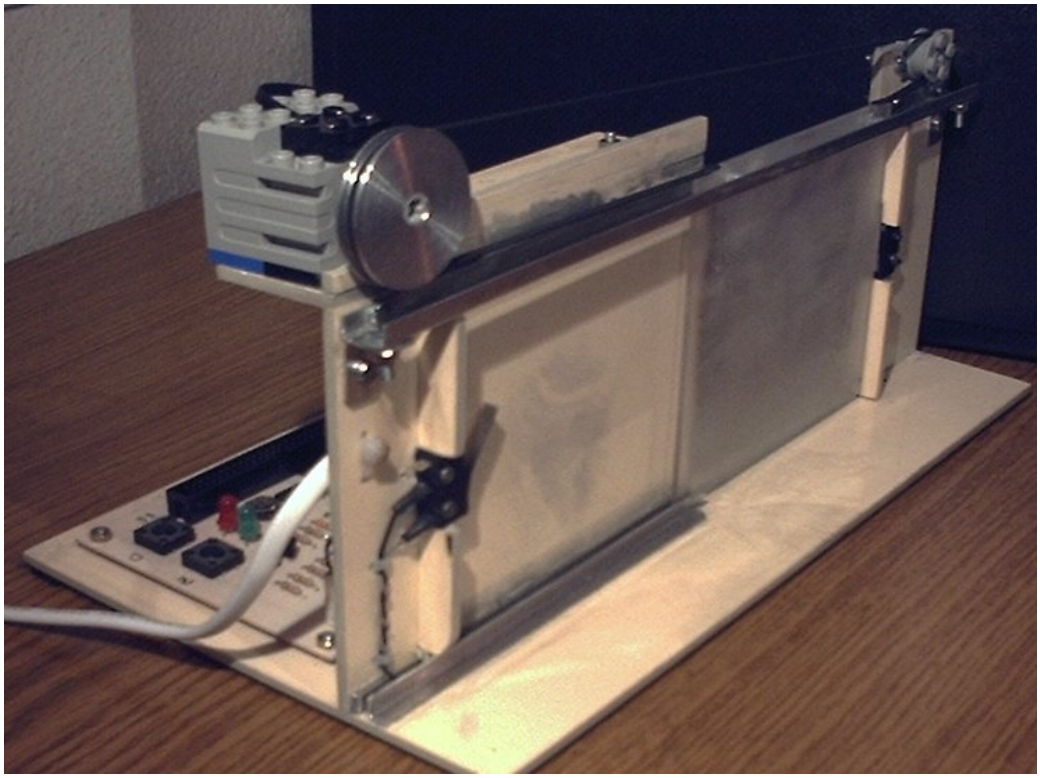


Slika št. 1: Blok shema sistema

Glavni del sistema je seveda maketa avtomatskih vrat, na kateri so vrata iz pleksi stekla, motor, ki poganja vrata, dve stikali končnih leg ter senzor.

Ker sem uporabil tak senzor kot ga dobimo pri realnih vratih, sem uporabil tudi elektroniko, ki spada zraven. Le ta pa se napaja z izmenično napetostjo 220 V. Na vezju sem uporabil le preklapljanje senzorja.

Poleg tega se na krmilniku makete nahaja še gonilnik motorja, ki pretvori digitalni signal v analogno napetost in z njo krmili motor. Na vezju sta še tipki za vedno odprta in vedno zaprta vrata ter svetleči diodi, ki povesta kdaj so vrata vedno zaprta ali odprta. Vse to pa je priključeno na mikrokrmilniški sistem HC11 [2], ki temelji na mikrokrmilniku Motorola M68HC11A1. Podrobneje mikrokrmilniškega sistema ne bom opisoval, saj sem sistem uporabil kot že delujočo celoto in je plod dela drugega avtorja. Celotni sistem prikazuje slika številka dve.



Slika št. 2: Slika makete avtomatskih vrat

3.1 Maketa

3.1.1 Ogrodje

Ogrodje makete avtomatskih vrat je sestavljeno iz materiala Akril Butadien Syrol ali krajše ABS. Material omogoča, da je maketa precej trdna z močnimi stebri, pa čeprav so stene tanke le tri milimetre. Material je tudi zelo primeren za obdelovanje in na maketi ni problema narediti manjših popravkov.

Maketa meri v dolžino 29 cm v višino pa 12 cm. Dimenzije ogrodja omogočajo, da imamo maketo postavljeno na delovni površini in tako v realnem času opazujemo delovanje programa in obnašanje vrat.

Ogrodje je sestavljeno iz spodnje plošče ter dveh stranskih stebrov. Centimeter od vrha prvega stebra se nahaja polica, na katero sta pritrjena motor in senzor. Na vsakem od stebrov se nagajata tudi stranska stebra, ki merita v širino en centimeter. Stebra imata nalogo vstaviti vrata, ko so le ta odprta ali zaprta. Na sredini stebrov sta vgrajeni stikali končnih leg, kateri sporočita programu, kdaj so vrata zaprta ali odprta.

Vsi deli makete, ki so pritrjeni na ogrodje so priviti z vijaki, kar daje maketi še posebno trdnost. Na spodnjem delu sem na vogale spodnje plošče pritrdil osem gumic dimenzij 1 x 1 cm. To omogoča, da maketa trdno stoji na delovni površini, hkrati pa sem lahko tudi potegnil žice, stikal končnih leg, pod maketo. Žice, ki niso pod maketo, sem trdno povezal z vezicami.

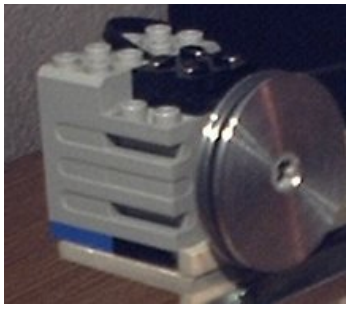
3.1.2 Vrata in vodila

Vrata na maketi so izdelana iz pleksi stekla, saj so le tako lahko dovolj trdna in tanka, a izgledajo podobno, kakor pravo steklo, iz katerega je v realnem svetu izdelano največ avtomatskih vrat. Dimenzije vrat so 110 x 110 cm. Pol centimetra od vrha vrat sta na vsakem koncu pritrjena po en kovinski distančnik, preko katerih vrata visijo na vodilih. Na distančnik je nameščen tudi majhen del zelo tanke plastične cevi. Cev se vrti na distančniku in močno zmanjša trenje med njim in vodilom, tako da vrata tečejo maksimalno gladko.

Vrata torej visijo na vodilih, ki so dolga 29 cm v obliki črke L. Notranje vodilo mora biti čim ožje, da vrata drsijo čim bližje steni. Za zunanje vodilo pa širina ni ključnega pomena. Zato meri notranje vodilo 5 x 3 mm, zunanje pa 5 x 5 mm. Na spodnjem delu vrat sta tudi dve vodili, ki imata nalogo držati vrata v pravi smeri, da le ta ne zanihajo. Tukaj je prav tako notranje vodilo dimenzij 5 x 3 mm, zunanje pa 5 x 5 mm. Vodila so izdelana iz aluminija in so na vsakem koncu pritrjena z vijaki na ogrodje.

3.1.3 Motor

Del makete je tudi enosmerni elektromotor, ki ga lahko priključimo na napetost 9 V, vendar v našem primeru na njega nikoli ne pride več kot 5 V. Od višine napetosti na motorju je odvisna moč in hitrost motorja. Smer vrtenja motorja določamo z različnim potencialom napetosti med sponkami motorja. Motor sem vzel iz otroškega kompleta LEGO Technic. Pri preizkušanju makete sem uporabil precej motorjev iz različnih otroških igrač, vendar sem prišel do zaključka, da noben prenosni mehanizem ne deluje tako natančno in tekoče kot mehanizem, ki je vdelan v motor iz ″LEGO kock″. Slika številka tri prikazuje kako izgleda tak motor.



Slika št. 3: Motor

3.1.4 Prenosni mehanizem

Eden večjih problemov pri izdelavi makete je predstavljal prenosni mehanizem iz motorja na vrata. Kot sem že omenil sem uporabil mehanizem, ki je vdelan v motorju iz ″LEGO kock″, saj se je le ta najbolje obnesel. Na os motorja sem pritrdil aluminijasto kolo premera 3,5 cm z utorom za gumico. Motor s kolesom se nahaja na levi strani makete. Na desni strani makete pa je pritrjeno plastično kolo premera 1,5 cm z utorom za gumico, ki je prosto tekoče. To kolo sem prav tako dobil v kompletu LEGO kock. Zato, da sem lahko nastavil idealno hitrost odpiranja in zapiranje vrat mora biti kolo na levi strani dovolj veliko.

Med obema kolesoma sem napel gumico širine 3 mm. Na to gumico pa so pritrjena vrata. Za prenosni mehanizem z gumico smo se odločili zato, ker ob napačnem reguliranju motorja gumica drsi na kolesu. Tako je lahko motor v aktivnem delovanju tudi takrat, kadar so vrata že zaprta ali odprta. Tak prenosni mehanizem še posebej obogati prvotni namen sistema, ta je, da se študentje učijo programiranja mikrokrmilniškega sistema HC11 [2]. Program za krmiljenje vrat je lahko napisan popolnima napačno in zaradi tega ne bo z maketo nič narobe.

3.1.5 Senzor

Že prej sem omenil, da na maketi nisem uporabil samo senzorja kot elementa, ampak celoten senzorski sistem s celotno elektroniko. To je skoraj enak sistem kakor ga uporabljajo realna avtomatska vrata. Uporabil sem senzor italijanskega proizvajalca AVE, model Sistema 45, ki je precej majhen in tako zelo primeren za na maketo. Senzor je pritrjen pod polico, ki je del levega stebra. Ker sem uporabil celotni senzorski sistem, ga moramo napajati z posebnim virom napetosti 220 V.

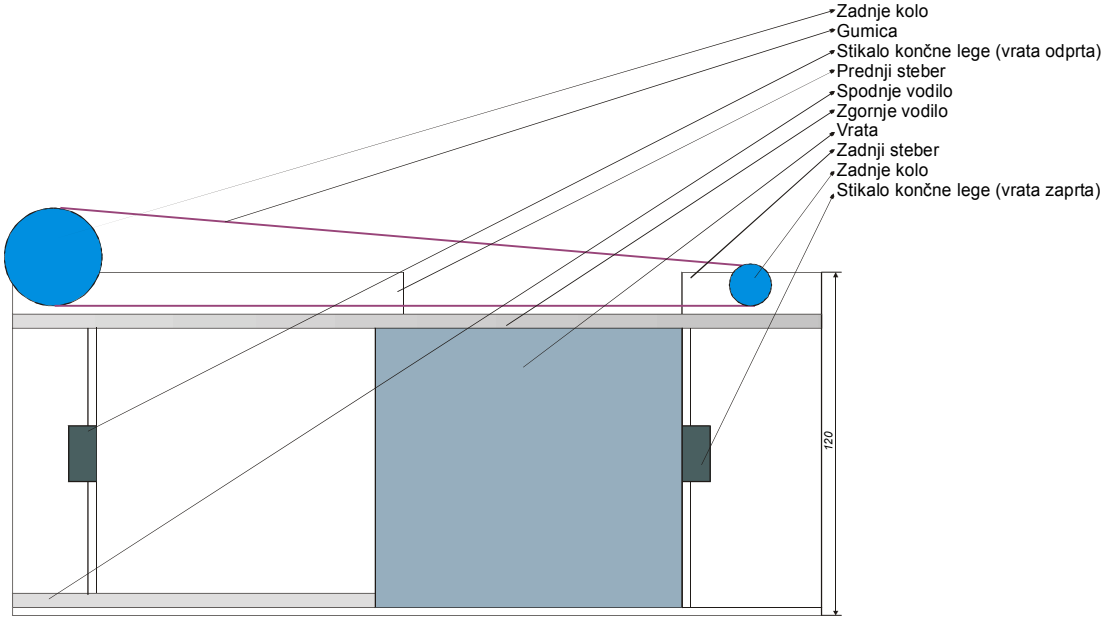
Prednje steklo senzorja ima nalogo razpršiti svetlobo. Originalno steklo pokriva zelo veliko področje, na katerem zaznava gibanje okoli senzorja. Ker pa tako veliko področje ni primerno za maketo, sem moral steklo oblepiti. Oblepil sem ga z črnim izolirnim trakom in tako zmanjšal območje, katerega zaznava senzor. Le tako se vrata odprejo samo takrat, kadar se jim dovolj približamo. Slika številka štiri prikazuje velikost senzorja in kako se le ta vključi v maketo.

Slika št. 4: Senzor

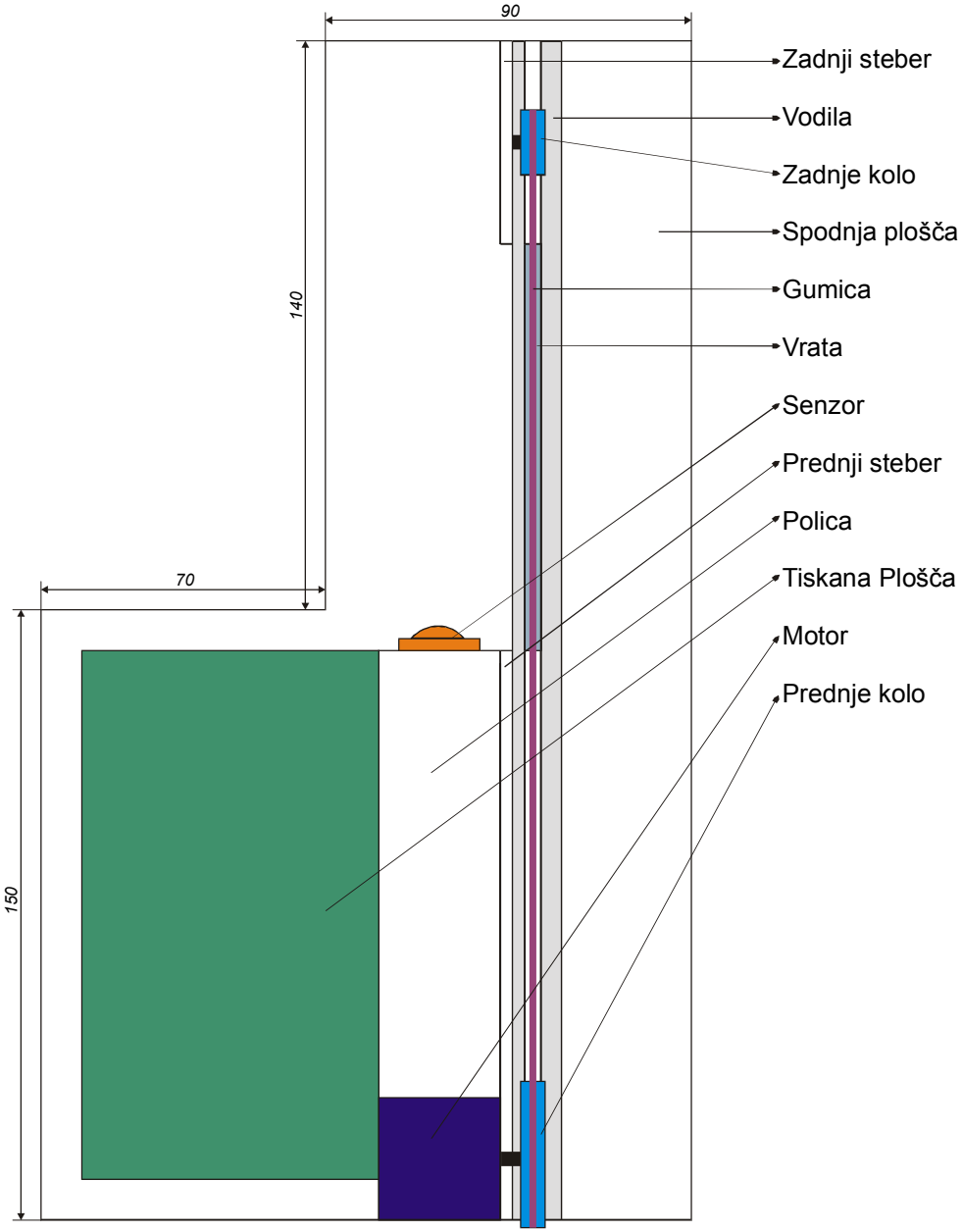
3.1.6 Tehnični načrt makete

Sliko številka pet in šest prikazujeta, kako so elementi na maketi sestavljeni skupaj. Na načrtu so navedene tudi mere nekaterih elementov, vseh mer pa nisem vključil, ker sem jih navedel že v prejšnjih odstavkih.

Prva slika prikazuje maketo od spredaj – naris. Narisana je iz strani kjer so vrata in oba kolesa na motorju. Druga slika pa prikazuje maketo od zgoraj – tloris.



Slika št. 5: Naris makete



Slika št. 6: Tloris makete

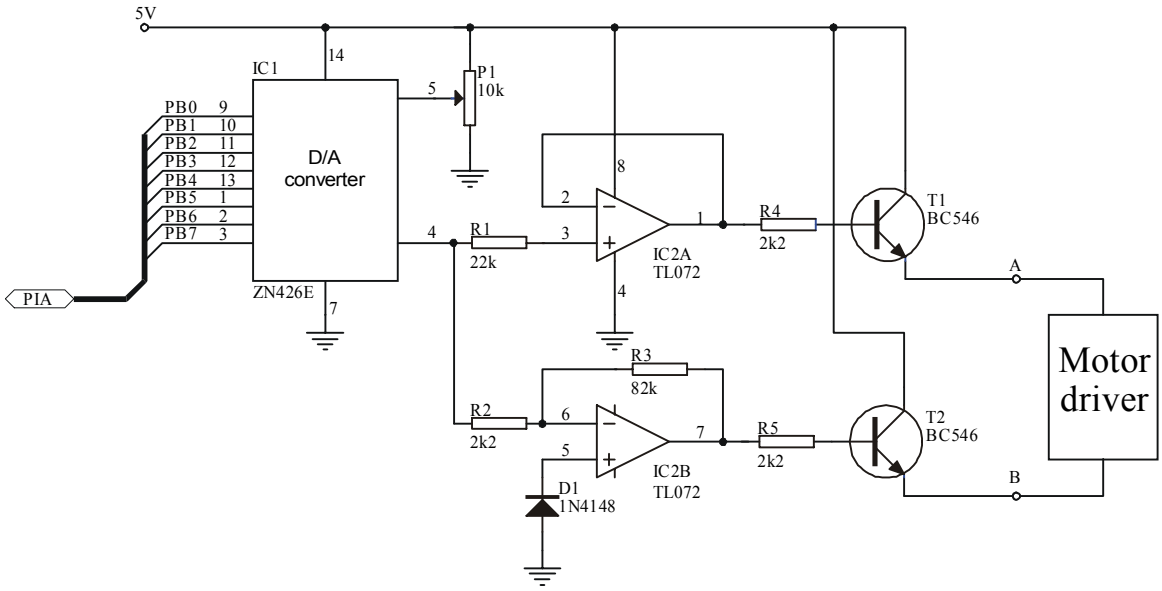
3.2 Krmilnik makete

Vezje, ki krmili maketo je sestavljeno iz štirih delov. Najprej najdemo vezje, ki krmili motor, nato tipkovnico (za vklapljanje dveh tipk in dveh stikal na maketi), vezje za prižig svetlečih diod ter detektor senzorja.

Celoten krmilnik je narejen na tiskani plošči, ki je pritrjena na maketo. Vezje se napaja s petimi volti, ki jih pripeljemo iz mikrokrmilniškega sistema HC11 in ne potrebuje svojega dodatnega napajalnika.

3.2.1 Krmilnik motorja

Krmilnik enosmernega elektromotorja sestoji iz dveh delov. Prvi del pretvarja digitalni signal, ki ga pripeljemo iz mikrokrmilniškega sistema v analognega ter prilagodi napetosti in tokova, ki jih peljemo na drugi del – regulator motorja. Drugi del je torej regulator motorja, ki ima nalogo spreminjati električni potencial med sponkama motorja. Slika številka sedem prikazuje prvi del – električno shemo digitalno analognega pretvornika z izhodno stopnjo.



Slika št. 7: Električna shema D/A pretvornika z izhodno stopnjo

Idejo za uporabo takšnega vezja sem našel v reviji Svet elektronike [4]. Vezje sestoji iz D/A pretvornika ZN426-E (IC1) [5], dveh operacijskih ojačevalnikov, ki se nahajata v integriranem vezju TL072 (IC2) [6], ter dveh izhodnih tranzistorjev BC546 (T1 in T2). Na priključek 5 D/A pretvornika, vodimo napetost preko trimer potenciometra P1. Na ta način nastavljamo maksimalno analogno napetost na izhodu D/A pretvornika

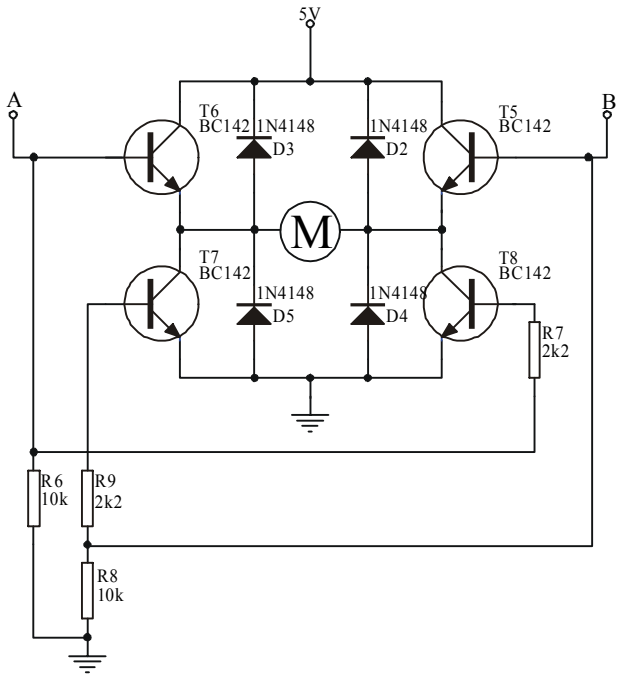
(priključek štiri), s tem pa tudi maksimalno izhodno napetost in maksimalno hitrost vrtenja elektromotorja. Na priključke ena do tri ter devet do 13 pripeljemo B port PIA-e iz mikrokrmilniškega sistema HC11. Krmiljenje hitrosti vrtenja elektromotorja dosežemo s spreminjanjem digitalnega stanja na vhodu D/A pretvornika. Motor se vrti z maksimalno napetostjo, ki jo nastavimo z trimerjem P1, kadar na vhod pripeljemo vrednost 256 oziroma binarno 11111111. Ko pa se na vhodu pojavi vrednost 0 oziroma 00000000, motor miruje. Seveda smo naš krmilnik z nadaljnjim vezjem priredili tako, da motor miruje pri vrednosti 67, pri vrednosti 255 se vrti maksimalno v desno in pri vrednosti 0 maksimalno v levo stran.

Iz izhoda D/A pretvornika vodimo signal preko uporov R1 in R2 na dva operacijska ojačevalnika. Prvi pripelje na izhod le tokovno ojačanje, napetost pa zaradi povratne zveze ostane ista. Tokovno ojačanje potrebujemo zaradi krmiljenja izhodnih tranzistorjev.

Pri drugem ojačevalniku pripeljemo signal na negativni vhod, kar pomeni, da invertira napetost - inverter. Torej, ko pripeljemo na vhod maksimalno napetost, dobimo na vhodu minimalno in obratno. Ta operacijski ojačevalnik ojači napetostni signal za približno 37 krat. Na pozitivni vhod pripeljemo diodo, ki je vezana na maso. Ta vhod bi lahko pustili tudi v zraku, vendar bi se zato lahko iz različnih zunanjih vzrokov napetost na izhodu operacijskega ojačevalnika spreminjala. Na vhodu je zato vedno dobro imeti neko točno določeno stanje.

Ker motor in njegov regulator porabita kar precej moči, imamo na vsakem izhodu iz operacijskega ojačevalnika dodan še po en tranzistor, ki nam tokovno ojači izhodni signal.

Drugi del vezja spreminja napetost med sponkama na motorju po načelu plus-minus in minus-plus. Električno shemo vezja prikazuje slika številka osem.



Slika št. 8: Električna shema regulatorja motorja

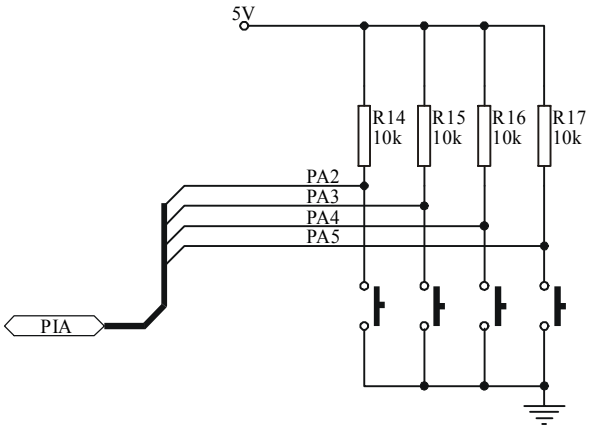
Vezje spreminja smer vrtenja motorja. V prvem primeru pripeljemo maksimalno napetost na točko A in minimalno na točko B (to naredimo z elektroniko v prejšnjem vezju). Torej maksimalna napetost pride na bazo tranzistorjev T6 in T7 zato se bazi teh dveh tranzistorjev odpreta in tok teče v eni smeri čez motor. V drugem primeru pa pripeljemo maksimalno napetost na točko B in minimalno na točko A. Torej odprti sta bazi tranzistorjev T5 in T7 in motor se vrti v drugi smeri. Z spreminjanjem napetosti med točkama A in B pa spreminjamo hitrost vrtenja motorja.

Vsak elektromotor je sestavljen iz tuljavic v katerih se inducirajo napetostne konice, katere pogosto motijo delovanje mikrokontrolerja, lahko pa tudi uničijo občutljivejše gradnike naprave. Te napake odpravimo z štirimi diodami D2, D3, D4 in D5, ki jih vežemo med kolektorje in emitorje tranzistorjev.

3.2.2 Tipkovnica

Na našem vezju imamo tudi dve tipki, s katerima programu oddamo informacijo, kdaj naj bodo vrata stalno odprta ali stalno zaprta. Stalno odprta ali stalno zaprta vrata pomeni, da program v takem primeru odpre ali zapre vrata, senzor pa ne vpliva na položaj vrat. Torej ne glede na gibanje okoli senzorja so vrata vedno zaprta ali vedno odprta vrata. Tipko za vedno zaprta vrata peljemo na četrti bit A porta PIA-e, za vedno odprta pa na peti bit A-porta PIA-e. Sistem vrnemo nazaj v prvotno delovanje (kjer reagira tudi senzor) tako, da tipko, ki smo jo pritisnili (za vedno odprta ali vedno zaprta vrata) pritisnemo še enkrat. Tipkovnica deluje z negativno logiko, zato imamo na PIA-i vedno logično ena, razen kadar imamo pritisnjeno tipko. Da ne pripeljemo na A port mikrokrmilniškega sistema prevelik tok ob pritisnjeni tipki, le tega omejimo z upori.

Prav tako imamo na maketi tudi dva stikala končnih leg. Ta dva stikala nam oddata informacijo, kdaj so vrata odprta ali kdaj so zaprta. Stikala fizično vklopijo vrata kadar so le ta odprta ali zaprta. Stikalo za zaprta vrata pripeljemo na trtji bit A porta PIA-e, za odprta vrata pa drugi bit A porta PIA-e. Tudi to vezje deluje z negativno logiko in peljemo na PIA-o logično ena, razen kadar je vkjučeno stikalo. Ob pritisnjenemu stikalu omejimo z upori. Električno shemo tipkovnice prikazuje slika številka devet.

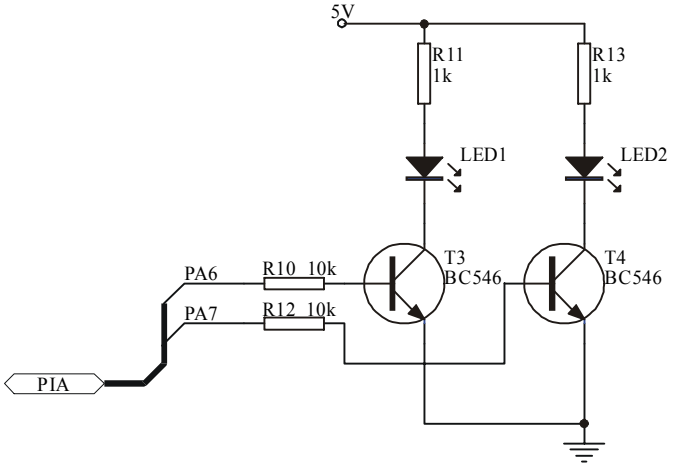


Slika št. 9: Električna shema tipkovnice

3.2.3 Vezje za prižig LED

Kot sem omenil v zgornjem odstavku imamo na tiskani plošči dve tipki – za vedno odprta in vedno zaprta vrata. Da vemo, kdaj imamo sistem v takem položaju, sta dodani tudi dve svetleči diodi. Kadar sveti zelena dioda pomeni, da so vrata vedno zaprta in nanj senzor ne vpliva. Kadar sveti rdeča dioda pomeni, da so vrata vedno odprta. Sistem deluje v normalnem položaju takrat, kadar ne sveti nobena dioda.

Zelena svetleča dioda je na vezju speljana na šesti bit A porta PIA-e, rdeča pa na sedmega. V vezje imamo dodana tudi dva tranzistorja BC546 (T3 in T4). Baze tranzistorjev programsko odpremo preko mikrokrmilniškega sistema. Ko je baza tranzistorja odprta teče tok iz petih voltov preko svetleče diode na maso in tako dioda zasveti. Zaradi tokovnih omejitev imamo dodanih še nekaj uporov. Električno shemo prikazuje slika številka 10.

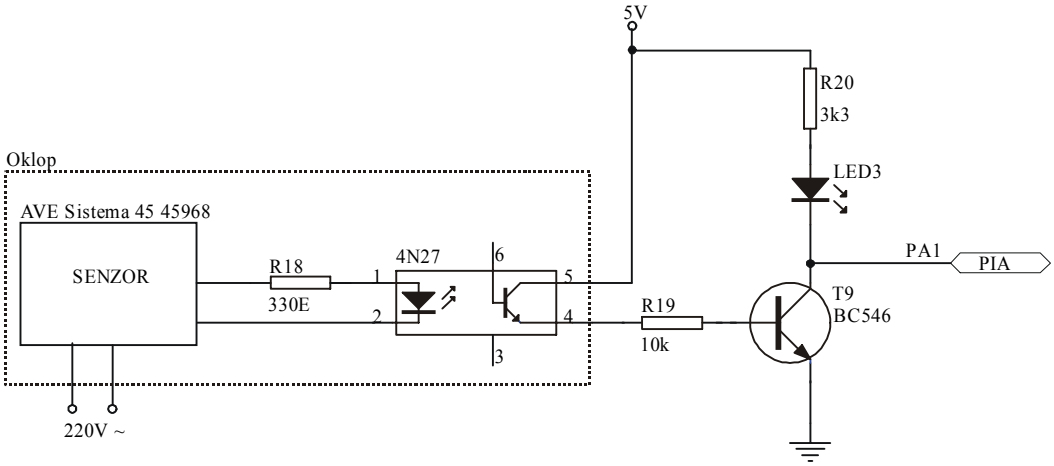


Slika št. 10: Električna shema vezja za prižiganje svetlečih diod

3.2.4 Detektor senzorja

Kot sem prikazal pri opisu makete, v sistemu uporabljamo senzor z vso pripadajočo elektroniko. Seveda pa je tak senzor narejen za primere realnih vrat, kar pomeni, da na njega kar precej vpliva jakost svetlobe in čas, ki nam pove, koliko časa naj senzor ″hrani″ signal, ki ga zazna ob gibanju. Prav zato imamo na njem dva trimerja s katerima lahko nastavljamo omenjena parametra. Trimer za nastavitev občutljivosti svetlobe sem nastavil na nič, kar pomeni, da svetloba na senzor praktično nima nobenega učinka. Trimer za nastavitev časa, pa sem prav tako nastavil na nič, vendar se je izkazalo, da senzor še vedno ″hrani″ informacijo za najmanj sedem sekund. S temi trimerji ni bilo mogoče nastaviti senzor, da bi preklopil takoj. Potrebno je bilo poiskati dokumentacijo vezja v senzorju [8] in ugotovil sem, da le to uporablja integrirano vezje UAA 4713. Tako sem lahko peljal izhod čipa direktno na vezje tiskane ploščice. S tem sem dosegal, da lahko v našem vezju uporabljamo le prekljaplanje senzorja.

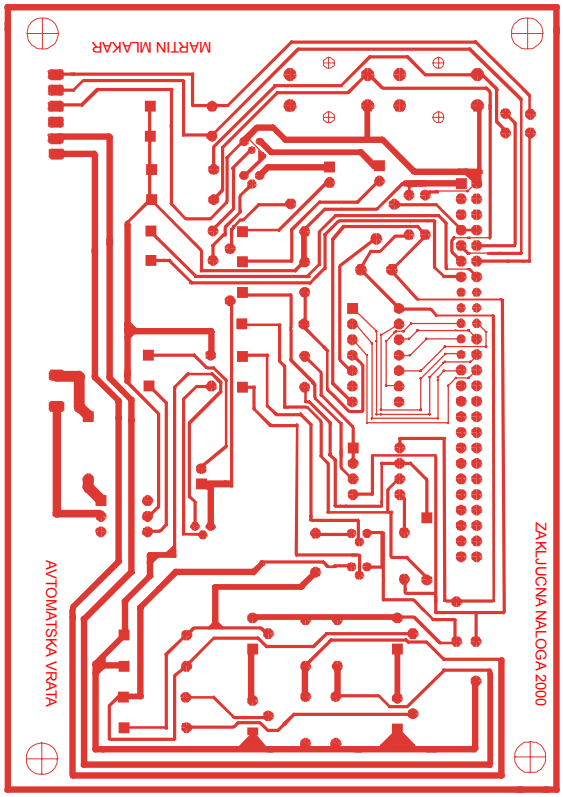
Senzor torej napajamo z izmenično napetostjo 220 V, kar pomeni, da jo kot tako dobimo tudi na izhodu. Zato je bilo v detektor potrebno vključiti nek element, ki bo galvansko ločil tokokroga. Odločil sem se za optični sklopnik 4N27 [7], ki ga vkljaplja senzor. Ta del torej deluje pod napetostjo 220 V in se nahaja na istem tiskanem vezju kakor ostala elektronika. Zato sem ga zaradi varnosti prekril z plastičnim oklopom. Izhod sklopnika sem peljal na bazo tranzistorja BC546 (T9), kateremu optični sklopnik odda informacijo, kdaj je senzor vključen in s tem odpre tranzistor. Preko tranzistorja sem zvezal tudi svetlečo diodo, saj le tako lahko nazorno opazujemo, kdaj je senzor sprožen. Izhod tranzistorja pa peljemo tudi na bit ena, PIA-e (A port) in tako tudi opravilo SENZOR, v našem programu, zazna informacijo, o vključenem senzorju. Vezje nazorneje prikazuje slika številka 11.



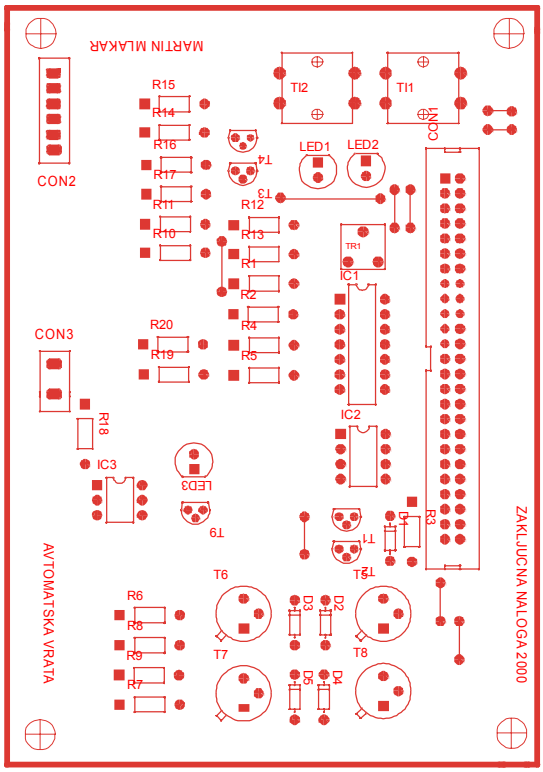
Slika št. 11: Detektor senzorja

3.2.5 Tiskano vezje

Tiskano vezje dimenzije 9 x 13 cm, ter debeline 1,5 milimetra, je narejeno z bakrom prevlečenim vitroplastom. Vezje sem narisal s programskim paketom Tango. Narisan izdelek sem nato prenesel na baker s postopkom sitotiska. Postopek jedkanja neuporabljenega bakra sem izvedel v mešanici vodikovega peroksida, solne kisline in vode. Ploščico sem nato z vijaki pritrdil direktno na maketo avtomatskih vrat. Na ploščico je z vijaki pritrjen tudi plastični oklop, ki preprečuje dotik osebe na del vezja, ki deluje z napetostjo 220 V. Slika številka 12 prikazuje vezje, ki sem ga s postopkom sitotiska prenesel na baker, slika številka 13 pa prikazuje prostorski razpored elementov na tiskanem vezju.



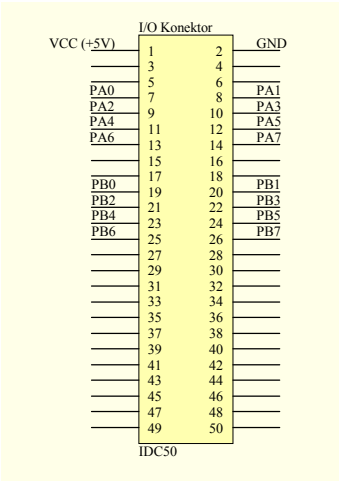
Slika št. 12: Tiskano vezje



Slika št. 13: Razpored elementov tiskane ploščice

3.2.6 I/O konektor (Povezava makete z mikrokrmilniškim sistemom)

Za povezavo mikrokrmilniškega sistema HC11 z zunanjimi enotami imamo na razpolago 40 linij, ki so vezane na vhodno/izhodni konektor tipa IDC50. Enak konektor je nameščen tudi na tiskanem vezju gonilnika makete, oba pa sta povezana s 50 žilnim ″flat″ kablom. Seveda izhodna naprava, ki je v našem primeru priključena na mikrokrmilnik, torej maketa, za svoje delovanje ne potrebuje vseh razpoložljivih linij. Tako so na sliki številka 14, ki prikazuje razpored linij vhodno/izhodnega konektorja, označene le uporabljene linije.



Slika št. 14: I/O konektor

Maketa uporablja naslednje linije:

- VCC, 5V napajanje,
- GND,
- PA0 .. PA7, 8-bitna dvosmerna vrata A, paralelnega vmesnika PIA6821,
    - PA0 (vhod), senzor,
    - PA1 (vhod), senzor,
    - PA2 (vhod), stikalo – odprta vrata,
    - PA3 (vhod), stikalo – zaprta vrata
    - PA4 (vhod), tipka – vrata stalno zaprta,
    - PA5 (vhod), tipka – vrata stalno odprta,
    - PA6 (izhod), zelena svetleča dioda – vrata zaprta;
    - PA7 (izhod), rdeča svetleča dioda – vrata odprta.
- PB0 .. PB7, 8-bitna dvosmerna vrata B, paralelnega vmesnika PIA6821. Vse linije so izhodne in so speljane na D/A pretvornik gonilnika motorja.

3.3 Programski del

Za pravilno delovanje makete avtomatskih vrat je programski del ključnega pomena. Seveda lahko ob narejeni maketi avtomatskih vrat program tudi spremenimo ali ga nadgradimo. Maketa je tako idealen učni pripomoček za učenje programiranja v zbirniškem jeziku. Program sem izdelal v programskem okolju HC11IDT, verzija 1.1.

3.3.1 Delovanje programa

V programu je zapisano, kako se bodo avtomatska vrata obnašala. Ko program zazna gibanje okrog senzorja, odpre vrata. Vrata se začno odpirati z maksimalno hitrostjo. Med tem ko se odpirajo, program šteje čas odpiranja in približno dva centimetra preden se vrata dokončno odprejo, jih program začne upočasnjevati. Tako se tik pred dokončnim odprtjem vrata gibljejo z minimalno hitrostjo. Nato program šteje čas, ko so vrata odprta. Po približno štirih sekundah jih program zapre. Postopek zapiranja vrat je enak postopku odpiranja. Zapirati se začnejo z maksimalno hitrostjo in približno dva centimetra pred dokončnim zaprtjem, program upočasni vrata tako, da se vrtijo z minimalno hitrostjo tik pred dokončnim zaprtjem. Seveda se vrata zaprejo le v primeru, da med odpiranjem, čakanjem na zapiranje ali zapiranjem, ni bil sprožen senzor. Kadar je senzor sprožen med čakanjem vrat na zapiranje, program začne šteti čas čakanja na zaprtje, od sprožitve senzorja. Kadar pa sprožimo senzor, ko se vrata že zapirajo, program prekine zapiranje in ponovno odpre vrata. Tako ne more priti do neljubega dogodka, ki bi se lahko zgodil v realnem svetu, da bi namreč oseba, ki bi hotela iti skozi vrata tik za osebo pred njo, ostala med vrati in bi jo le ta stisnila.

Program pa tudi opazuje tipki, ki sta pritrjeni na vezani plošči. Ko pritisnemo tipko za skozi zaprta vrata, program prižge zeleno svetlečo diodo in zapre vrata. Vrata so tako vedno zaprta ne glede na vklapljanje senzorja. Program vrnemo v prvotno delovanje z ponovnim pritiskom te iste tipke. Če pa hočemo imeti vrata vedno odprta, pritisnemo tipko za vedno odprta vrata in prižge se rdeča svetleča dioda. Tukaj prav tako vrnemo program v prvotno delovanje s ponovnim pritiskom te iste tipke. Seveda lahko med tema dvema funkcijama preklapljamo direktno iz vedno odprtih vrat v vedno zaprta in obratno.

3.3.2 Programiranje v realnem času

V zgornjem odstavku sem opisal delovanje naprave, ki je obenem tudi opis nalog, ki jih mora program opravljati. Na kratko lahko vse strnemo v dejavnosti, ki si sledijo v navedenem zaporedju. Program mora:

- preveriti, če je bila pritisnjena katera od tipk,
- prižgati svetlečo diodo ob pritisnjeni tipki,
- regulirati motor,
- zadržati odprta vrata nekaj sekund,
- bedeti nad senzorjem in
- nastaviti parametre ob resetu.

Mikrokrmilnik vseh nalog ne more opraviti istočasno. Lahko pa se posveti vsaki nalogi posebej, ter vsako opravi tako hitro, da izgleda, kot da bi vse delal hkrati. Mikrokrmilnik torej na račun svoje hitrosti hkrati in časovno usklajeno skrbi za vseh šest nalog. Časovna os mikrokrmilnika je torej razrezana na posamezne intervale, v katerih se procesor posveča različnim opravilom. Opisani postopek imenujemo časovno rezinjenje [1]. V mojem primeru sem uporabil operacijski sistem z že realiziranim urnikom opravil in razvrščevalnikom, torej rezinjenjem. Operacijski sistem je sestavni del uporabljenega mikrokrmilniškega sistema, zato na tem mestu nisem podrobneje opisoval delovanja t.i. jedra operacijskega sistema. Povzel sem le nekaj ključnih načel, ki bralcu pripomorejo k razumevanju delovanja programa.

Operacijski sistem za delo v realnem času vsebuje v svojem najožjem jedru razvrščevalni program, ki je odgovoren za logistiko časovnega rezinjenja, temelji pa na štirih poenostavitvah:

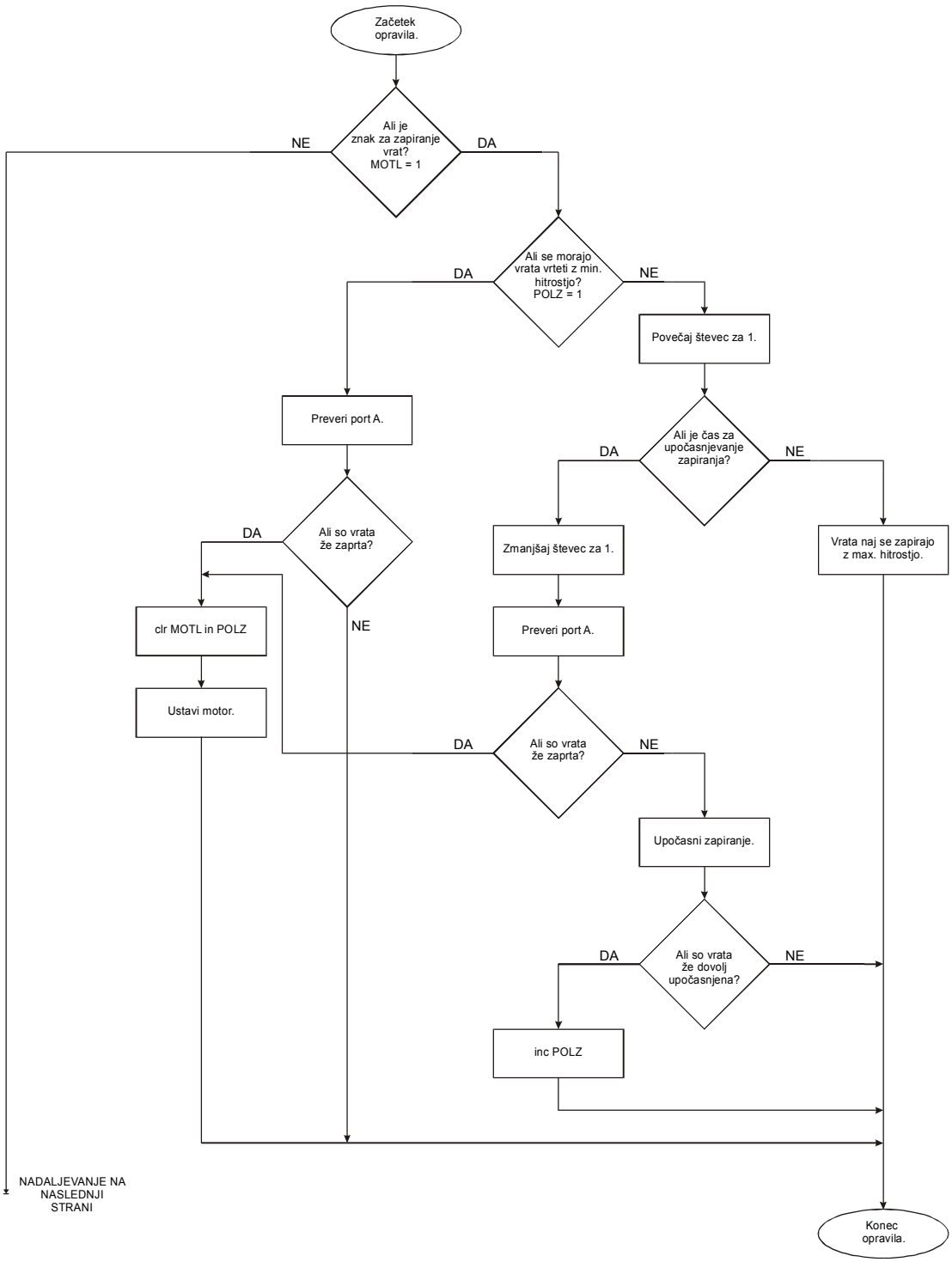
- vse časovne rezine so natanko enako velike,
- vsa opravila se vedno zaključijo še pred iztekom časovne rezine,
- urnik opravil obsega natanko 16 opravil in se izvaja ciklično ter
- zunanje enote ne povzročajo prekiniteve procesorja.

Na razpolago imamo torej šestnajst opravil, od katerih je vsako klicano natanko 64 krat v sekundi.

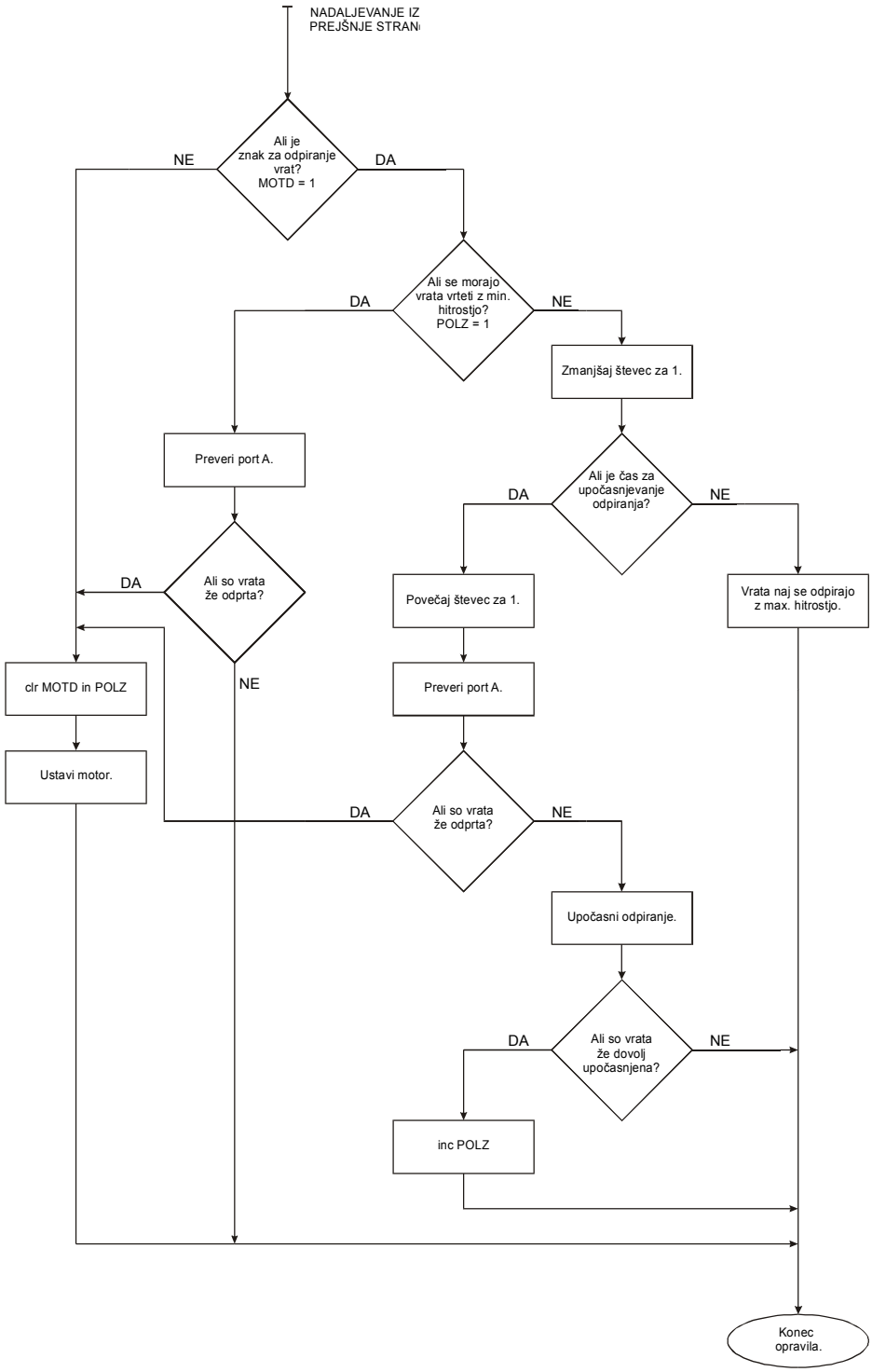
3.3.3 Opravilo MOT

Opravilo MOT krmili delovanje motorja. Razdeljeno je na dva dela: zapiranje in odpiranje vrat. Pri zapiranju opravilo najprej preveri, če se morajo vrata zapirati z minimalno hitrostjo. Nato glede na spremenljivko STEVEC pogleda ali se morajo vrata začeti upočasnjevati. Vsakič, ko se opravilo izvede, zmanjša spremenljivko STEVEC za ena, dokler se vrata ne začnejo upočasnjevati. STEVEC namreč ob resetu z opravilom default nastavimo na začetno vrednost. Torej dokler se vrata ne upočasnjujejo ali vrtijo z minimalno hitrostjo, se le ta zapirajo z maksimalno hitrostjo. Ko pa se začno upočasnjevati in so upočasnjena že dovolj, opravilo postavi spremenljivko POLZ na ena, kar pomeni, da se vrata zapirajo z minimalno hitrostjo. Takrat, ko opravilo dobi iz makete signal, da so vrata že zaprta, ustavi motor in izbriše spremenljivke. Popolnoma enak postopek poteka takrat, kadar se vrata odpirajo, le da se motor vrti v nasprotno smer.

Seveda se mora motor vrteti v obe smeri – odpiranje in zapiranje. Krmilnik motorja na tiskanem vezju je sestavljen tako, da motor miruje pri vrednosti B porta PIA-e 67. Pri vrednosti $EE se vrata zapirajo z maksimalno hitrostjo, pri vrednosti 24 pa odpirajo z maksimalno hitrostjo. Tako lahko s spreminjanjem vrednosti na B portu PIA-e reguliramo zapiranje ali odpiranje vrat. Podrobnejši opis opravila prikazuje slika številka 15.



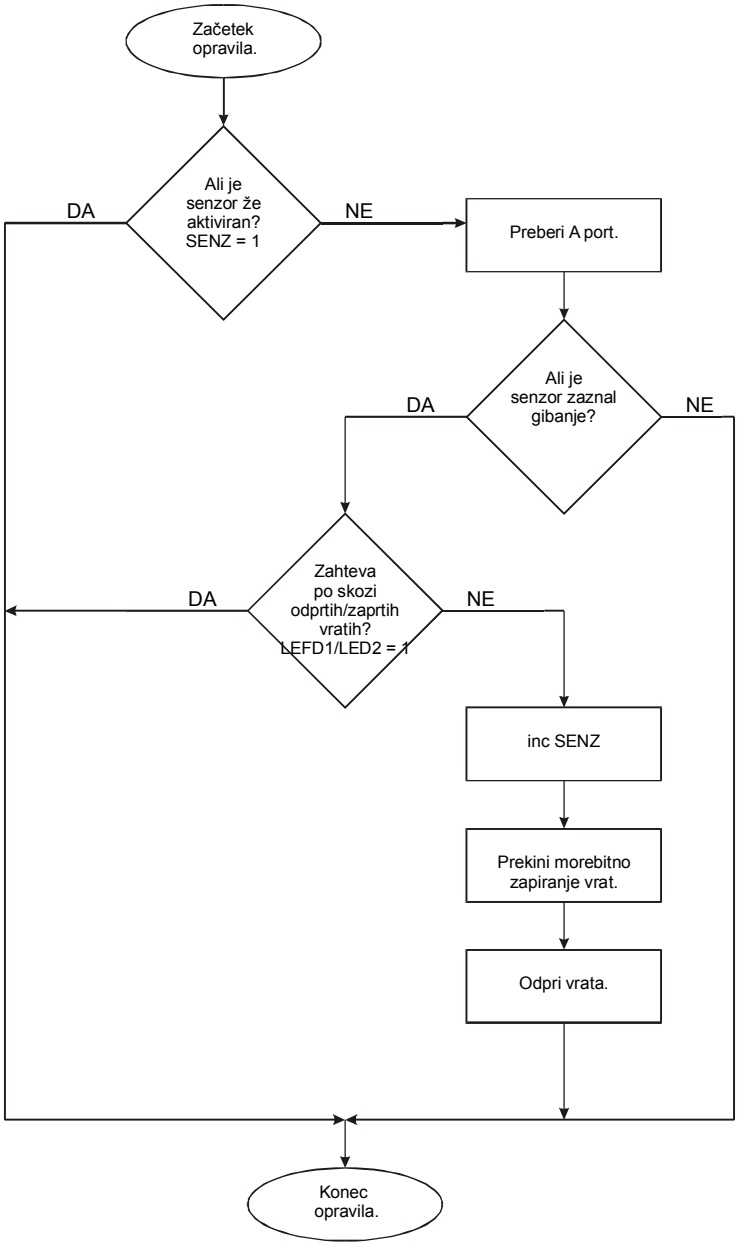
Slika št. 15a: Diagram poteka opravila MOT



Slika št. 15b: Diagram poteka opravila MOT

3.3.4 Opravilo SENZOR

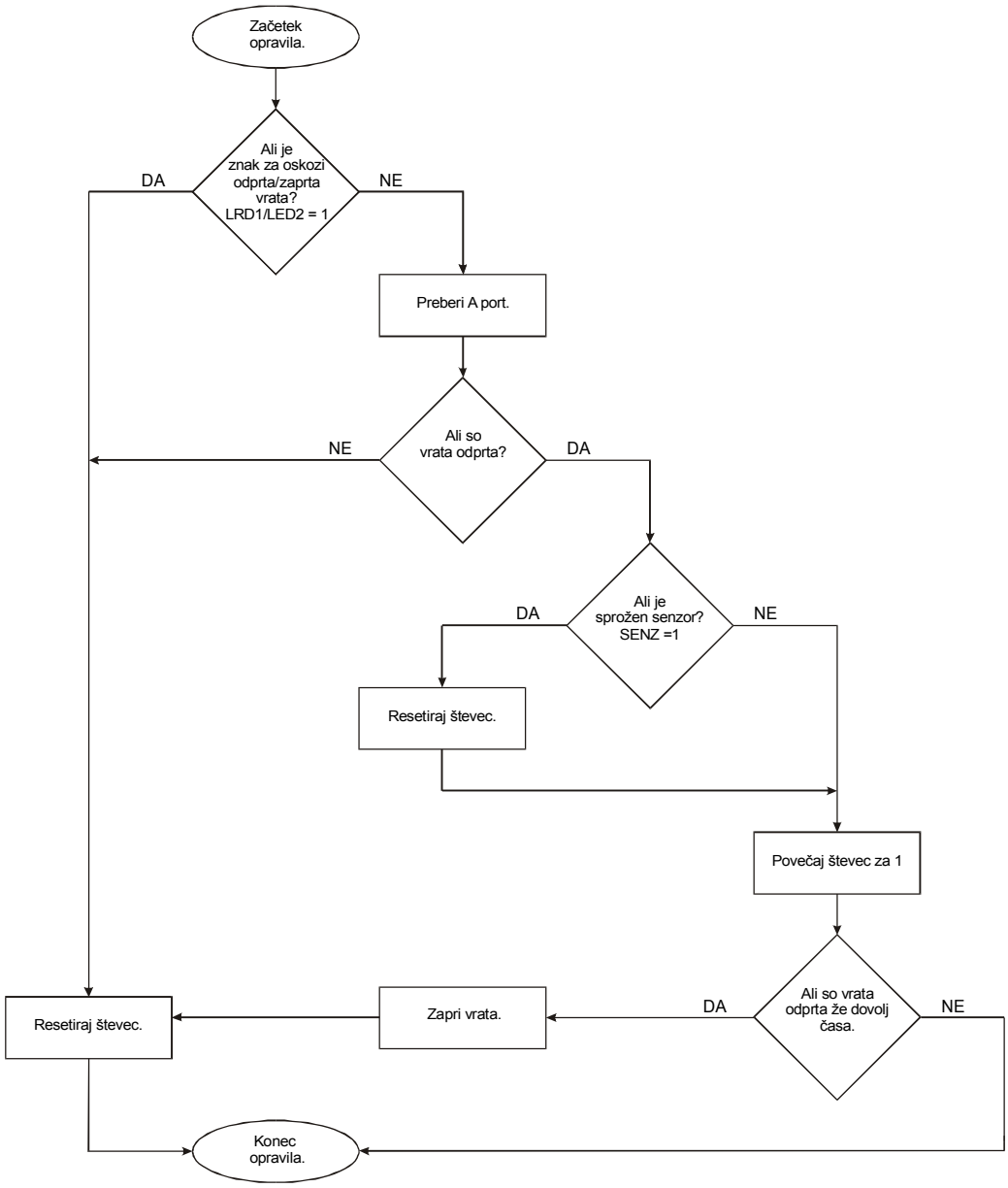
Kot pove že ime, opravilo opazuje senzor. Ko se opravilo prepriča o tem, ali je pritisnjena kakšna od tipk za stalno odprta ali stalno zaprta vrata, le ta postavi spremenljivko SENZ na ena, prekine morebitno zapiranje vrat in odpre vrata. Kadar opravilo istočasno prejme več informacij o aktiviranem senzorju, upošteva le prvo. To je zato, ker se opravilo izvaja tako hitro (64 krat v sekundi), da bi ob drugačnem ravnanju prišlo do napak. Motor bi namreč hotel odpreti vrata ogromno krat (64 krat toliko, kolikor sekund je vključen senzor). Slika številka 16 podrobneje opisuje opravilo.



Slika št. 16: Diagram poteka opravila SENZOR

3.3.5 Opravilo LOGIC

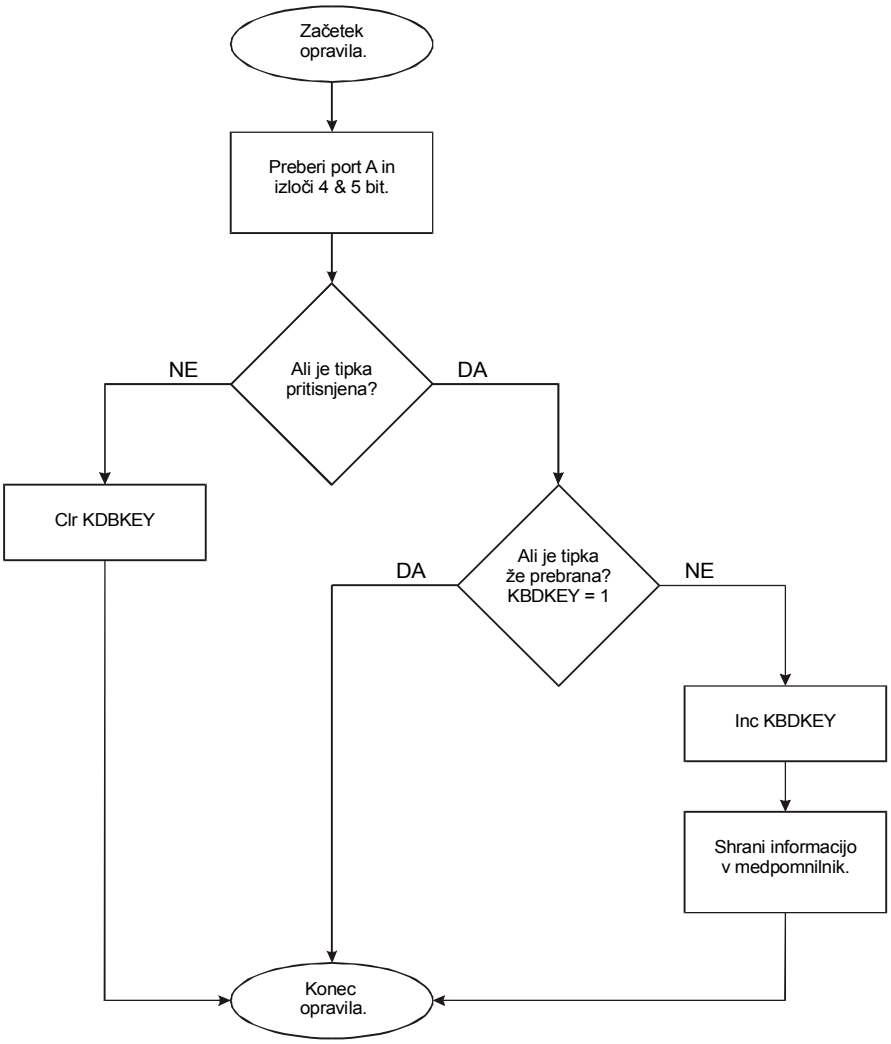
Opravilo opazuje vrata, ko so le ta odprta. Vsakič, ko se opravilo izvede se pri odprtih vratih poveča števec URA za ena in ga primerja z vrednostjo FF. Ko števec doseže to vrednost, torej štiri sekunde, so vrata odprta že dovolj časa in jih opravilo zapre. Pri vsaki sprožitvi senzorja opravilo zbriše števec URA. Seveda opravilo tudi preveri, če je mogoče bila izvedena zahteva po vedno odprtih ali zaprtih vratih. Opravilo natančneje opisuje slika številka 17.



Slika št. 17: Diagram poteka opravila LOGIC

3.3.6 Opravilo KBD

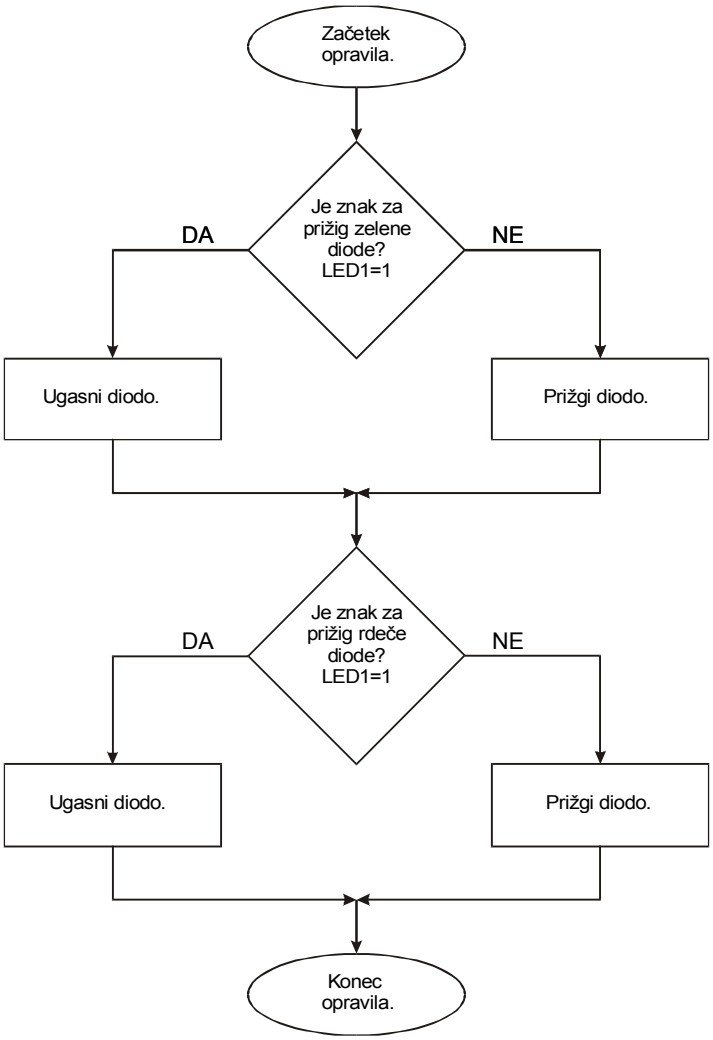
Opravilo KBD je gonilnik tipk za vedno odprta in vedno zaprta vrata, ki se nahajata na tiskani plošči. Opravilo zbira informacije od tipk in jih shrani v medpolnilnik ali ″buffer″. Podatki, ki so shranjeni v medpolnilniku KBDB do KBDE so sedaj na voljo kateremukoli delu programa. Večkratno branje iste tipke je onemogočeno z KBDKEY. Natančnejši opis najdete na sliki številka 18.



Slika št. 18: Diagram poteka opravila KBD

3.3.7 Opravilo LED

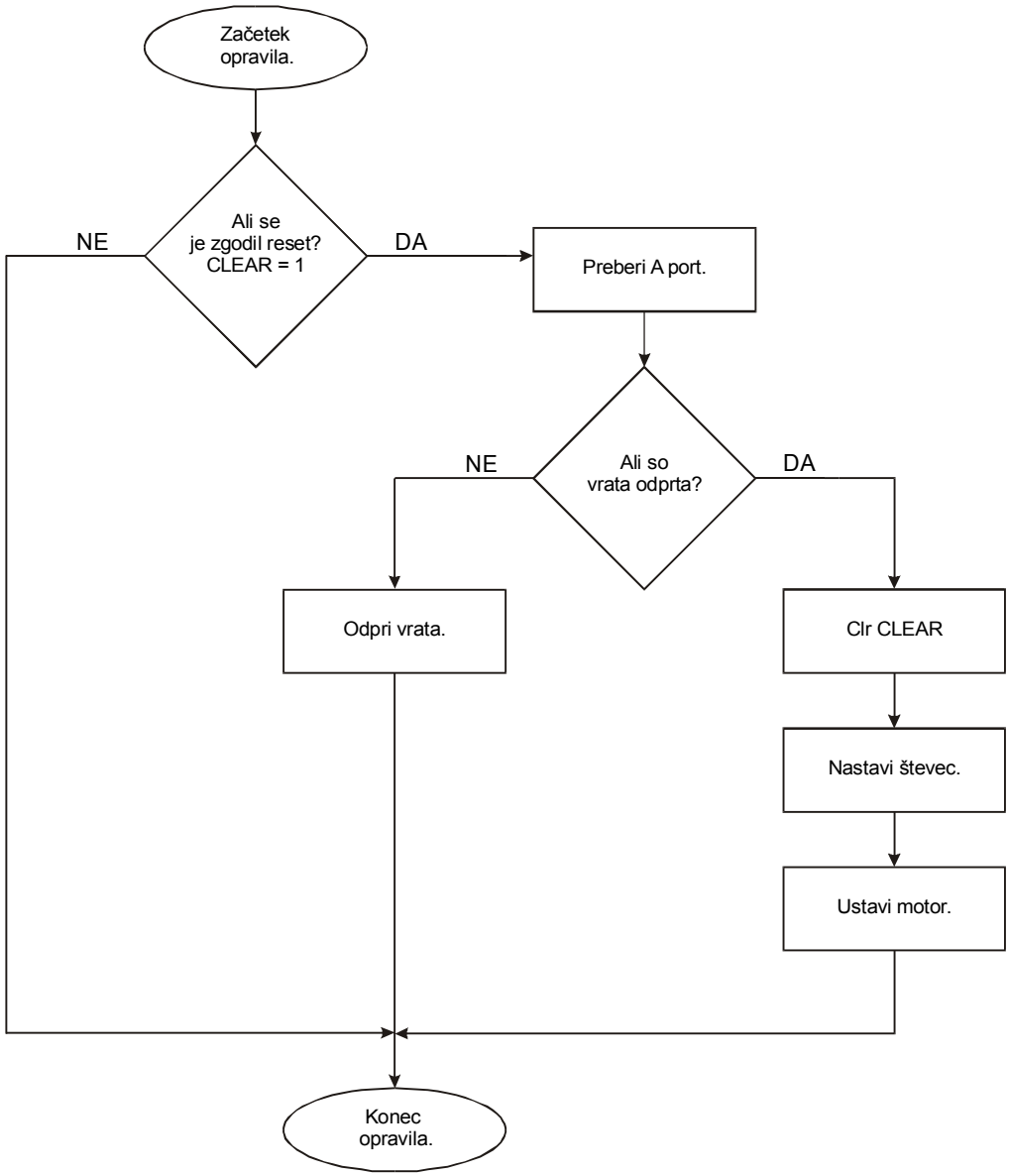
Opravilo LED je gonilnik svetlečih diod za vedno odprta in vedno zaprta vrata. Gonilnik preverja stanje tipk in glede na to prižge ali ugasne svetlečo diodo. Če je zahteva po vedno odprtih vratih, gonilnik prižge rdečo svetlečo diodo, kadar pa je zahteva po vedno zaprtih vratih, pa gonilnik prižge zeleno svetlečo diodo. Torej, če ne sveti nobena svetleča dioda pomeni, da je sistem v normalnem delovanju. Slika številka 19 natančneje opisuje opravilo.



Slika št. 19: Diagram poteka opravila LED

3.3.8 Opravilo DEFAULT

Opravilo DEFAULT v primeru reseta odpre vrata, nato ustavi motor in nastavi spremenljivko STEVEC na začetno vrednost. Ta postopek natančneje opisuje slika številka 20.



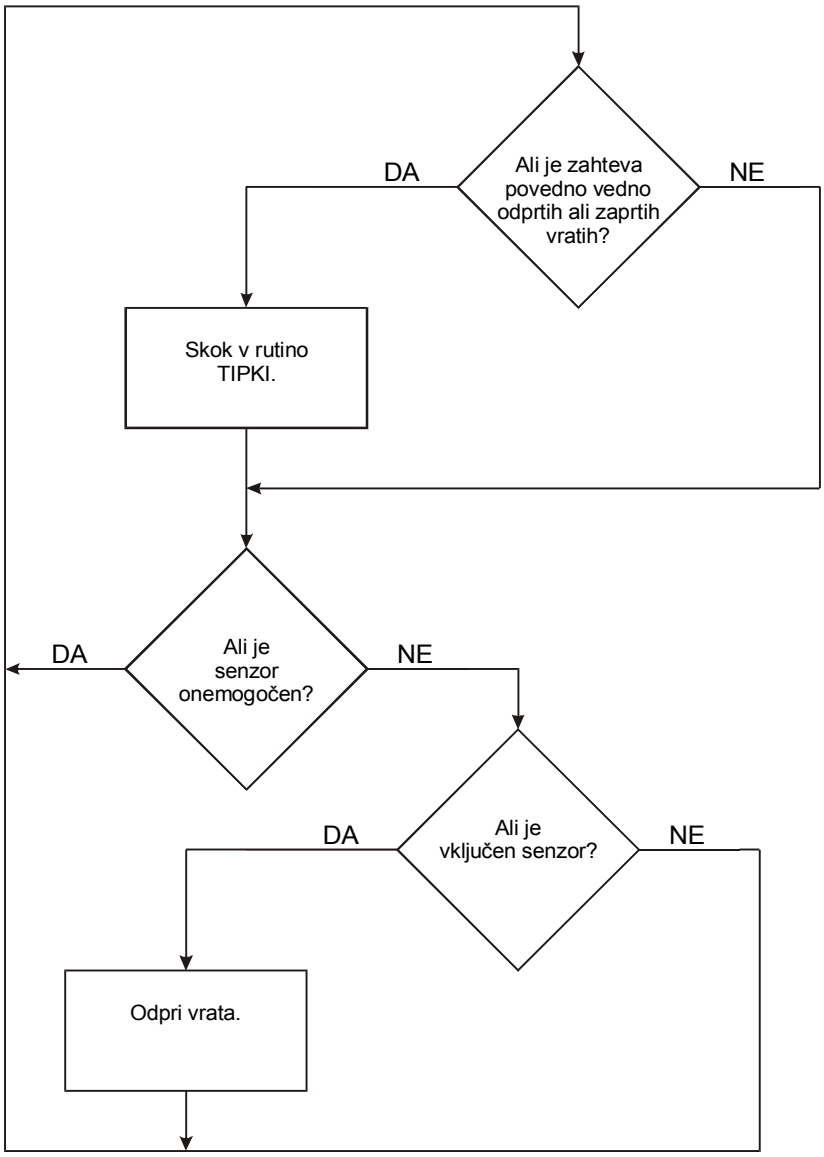
Slika št. 20: Diagram poteka opravila DEFAULT

3.3.9 Nastavitev paralelnega vmesnika PIA6821

Uporabljeni mikrokrmilniški sistem vsebuje tudi paralelni vmesnik PIA6821, ki omogoča priključitev najrazličnejših zunanjih enot na sistem. Paralelni vmesnik vsebuje dvoja osem bitna vrata, katerih vsako linijo je mogoče programirati kot vhodno oziroma izhodno linijo. Za krmiljenje maketa vrat potrebujemo 10 izhodnih in 6 vhodnih linij. Nastavitev vmesnika je zapisana v nastavitveni rutini INIT, ki ob zagonu programa poleg vmesnika nastavi tudi začetne vrednosti spremenljivk.

3.3.10 Glavni program (MAIN)

Vsa opisana opravila predstavljajo gonilnike posameznih naprav, z izjemo opravila LOGIC in DEFULT. Opravila skrbijo za prenos podatkov, nekatera prenašajo podatke v program, druga pa podatke prenašajo na zunanje naprave. Seveda potrebujemo nek program, ki bo nadziral, oziroma usmerjal podatke med posameznimi opravili. Program, t.i. glavni program mora biti zasnovan kot neskončna zanka, iz katere ni izhoda. Zanka pobira podatke iz medpomnilnikov, jih ustrezno obdeluje ter rezultate preko določenih spremenljivk posreduje opravilom. Vsekakor pa glavni program vsega dela ne opravi sam. Večino dela opravijo podprogrami, imenovani tudi rutine, katere ″main″ kliče po določenem zaporedju. Podprogram torej opravlja točno določeno nalogo, ki je sestavni del celotnega problema in katere rezultati narekujejo potek glavnega programa. Večino časa se glavni program brez učinka vrti v glavni zanki. Sprememba poteka nastane ob pritisku tipke ali sprožitvi senzorja. V primeru pritisnjene tipke za vedno odprta ali vedno zaprta vrata program požene rutino TIPKI, ki odpre ali zapre vrata in onemogoči delovanje senzorja. Kadar pa ni zahteve po vedno odprtih ali zaprtih vratih in je sprožen senzor, program odda informacijo opravilu MOT, da naj odpre vrata.



Slika št. 21: Diagram poteka glavnega programa

Program kot celoto torej sestavljajo opravila, ki posredujejo podatke, podprogrami, ki omenjene podatke obdelujejo ter glavni program, ki predpisuje protokol izvajanja rutin. Tako oblikovan program odlikuje preglednost, ki se nam obrestuje zlasti pri odpravljanju morebitnih nepravilnosti oziroma pri dopolnjevanju programa. Zbirniško kodo glavnega programa, ki je napisana po opisanem postopku prikazuje slika številka 22.

```
********************* MAIN PROGRAM ******************************

MAIN        jsr   KBDCHR     Zahteva po vedno odprtih/zaprtih vratih.
            beq   NAPREJ     Ne, nadaljuj.
            jsr   TIPKI      Da, zapri/odpri vrata.
NAPREJ      ldaa  LED1       Ali je senzor
            adda  LED2       onemogočen?
            bne   MAIN       Da, vrni se na začetek.
            ldaa  SENZ       Ne, preveri, če je senzor aktivira.
            beq   SENAPREJ   Ne, vrni se na začetek.
            inc   MOTD       Da, odpri vrata.
SENAPREJ    bra   MAIN       Vrni se na začetek.
```

Slika št. 22: Zbirniška koda glavnega programa

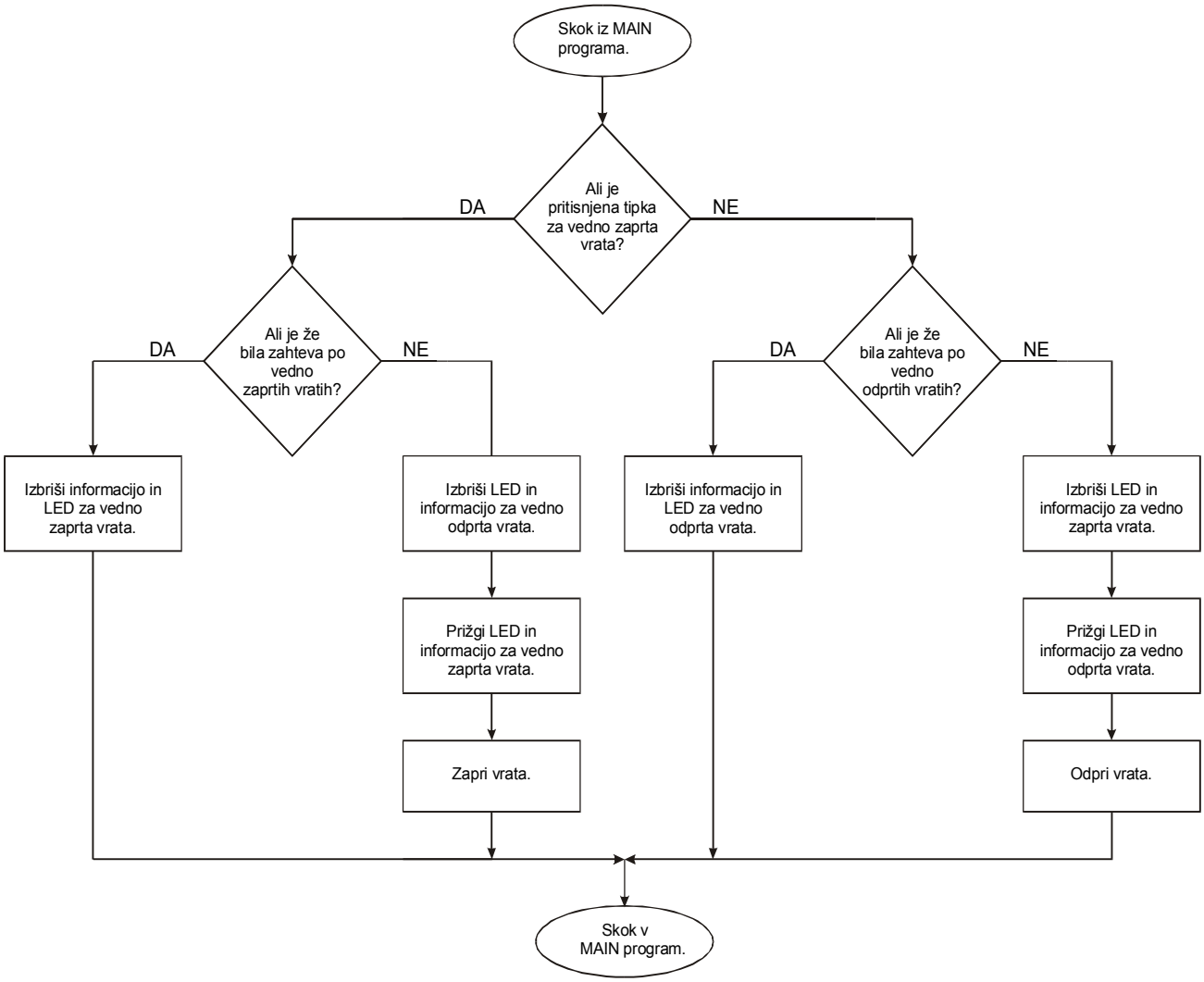
V širšem smislu operacijski sistem sestavlja tudi knjižnica podprogramov. Programske knjižnice operacijskih sistemov prirejenih za učne namene oziroma za razvojne namene, so običajno obsežne in vsebujejo veliko število splošno uporabnih podprogramov. V primeru realizacije konkretnega programa, kot je v mojem primeru, pa je knjižnica skrčena zgolj na podprograme, ki so v programu tudi uporabljeni. V mojem primeru imam v knjižnici podprogramov le dve rutini ali podprograma: KBDCHR in TIPKI.

3.3.11 Podprogram KBDCHR

Podprogram KBDCHR skrbi za pobiranje podatkov iz medpomnilnika. Rutina v akumulatorju A vrne znak iz medpomnilnika. V primeru praznega medpomnilnika rutina neuspešno branje označi ničelni bit na Z=1, v primeru polnega pa Z=0. Vsi ostali registri ostanejo nedotaknjeni.

3.3.12 Podprogram TIPKI

Rutina glede na informacije v medpolnilniku odpre ali zapre vrata in onemogoči senzor. Naprej preveri ali mora vrata odpreti ali zapreti. Nato glede na prejšnjo informacijo preveri, če so vrata morda že odprta ali zaprta. Ko konča vsa ta preverjanja rutina odpre ali zapre vrata in opravilu LED odda informacijo za prižig rdeče ali zelene diode. Podrobnejši potek rutine prikazuje slika številka 23.



Slika št. 23: Diagram poteka rutine TIPKI

4 Sklep

Izdelana maketa avtomatskih vrat omogoča enostavno učenje programiranja konkretnih primerov v zbirniškem jeziku. Prav to pa je bila poglavitna zahteva moje zaključne naloge. Maketo enostavno priključimo na mikrokrmilniški sistem, ki mu dodamo še napajanje, ter ga povežemo z osebnim računalnikom in že lahko pričnemo s programiranjem. Potek našega programa lahko realno opazujemo na maketi, kar močno poenostavi ter obogati učenje sestavljanja zbirniških programov. Zaradi prenosnega mehanizma iz motorja na vrata, ki vsebuje gumico, pa lahko motor krmilimo tudi napačno a s tem ne ogrozimo delovanje motorja. Gumica namreč drsi na kolesu, ki je pritrjen na motor, kar še obogati prvotni namen naloge.

Na prve težave sem naletel že pri samemu sestavljanju makete. Treba je bilo poiskati najprimernejši elektromotor in ga preko prenosnih mehanizmov povezati z gumico, ki poganja vrata. Preizkusil sem kar nekaj motorjev iz najrazličnejših otroških igrač. Na maketi sem imel kar precej vrst prenosnih mehanizmov, ki pa niso tekli dovolj tekoče. Izkazalo se je, da najboljšo rešitev predstavlja elektromotor iz kompleta igrač ″LEGO kocke″, ki že vsebuje enostaven prenosni mehanizem. Os motorja sem direktno povezal s kolesom, na katerem drsi gumica. Rešitev je sicer precej draga a je tudi najboljša.

Na naslednjo težavo sem naletel pri senzorju. Hotel sem namreč narediti posebno vezje, ki bo napajano direktno iz mikrokrmilniškega sistema in bo nadziralo delovanje samega senzorja kot elementa. Preizkusil sem kar nekaj vrst senzorjev in aplikacij. Kasneje sem se odločili, da zaradi nazornejšega prikaza delovanja senzorja pri realnih vratih in stabilnejšega delovanja uporabim senzor s pripadajočo elektroniko, ki se napaja z izmenično napetostjo 220 V.

Pri izdelavi tiskanega vezja sem uporabil srednješolske izkušnje za delo s programskim paketom Tango in jedkanjem tiskane ploščice. Veliko uporabnih podatkov (vezave operacijskih ojačevalnikov, ohišja elementov,…) pa sem našel v priročniku za elektrotehniko in elektroniko [3].

5 Zahvala

Iskreno se zahvaljujem vsem, ki so mi pomagali pri izdelavi moje zaključne naloge. Še posebej pa se zahvaljujem:

- mentorju doc. dr. Tadeju Tumi, ki mi je omogočil izdelavo zaključne naloge,
- Robertu Šubicu in ing. Sandiju Zabukovcu, za pomoč pri izdelavi makete avtomatskih vrat,
- podjetju Žustal Žiri, Romanu Žustu, za izdelavo aluminijastih sestavnih delov makete in
- podjetju LOG d.o.o. Srednje Bitnje, Dragu Logarju, za pomoč pri izbiri senzorja ter izročitvi le tega.

6 Priloga: Izpis celotnega programa

Priloga vsebuje izpis celotnega zbirniškega programa. Program vsebuje komentarje v slovenskem in angleškem jeziku. Ukaze poleg slovenskih komentarjev sem napisal pri programiranju makete, ukazi poleg angleških komentarjev pa so sestavni del uporabljenega operacijskega sistema.

```
************************* GLOBAL LABELS ****************************************
_IRAM   EQU   $0000       Internal RAM $0000..$00FF (256byte).
_PORTA  EQU   $1000       Port A Data register.
_PIOC   EQU   $1002       Parallel I/O Control Register.
_PORTC  EQU   $1003       Port C Data register.
_PORTB  EQU   $1004       Port B Data register.
_PORTCL EQU   $1005       PPort C Alternate Latched.
_DDRC   EQU   $1007       Port C Data Direction Register.
_PORTD  EQU   $1008       Port D Data Register.
_DDRD   EQU   $1009       Port D Data Direction Register.
_PORTE  EQU   $100A       Port E Data Register.
_CFROC  EQU   $100B       Timer: Compare Force.
_OC1M   EQU   $100C       Timer: Output Compare 1 Mask.
_OC1D   EQU   $100D       Timer: Output Compare 2 Data.
_TCNT   EQU   $100E       Timer: 16 bit free running counter.
_TIC1   EQU   $1010       Timer: Input Capture Register 1.
_TIC2   EQU   $1012       Timer: Input Capture Register 2.
_TIC3   EQU   $1014       Timer: Input Capture Register 3.
_TOC1   EQU   $1016       Timer: Output Compare Register 1.
_TOC2   EQU   $1018       Timer: Output Compare Register 2.
_TOC3   EQU   $101A       Timer: Output Compare Register 3.
_TOC4   EQU   $101C       Timer: Output Compare Register 4.
_TOC5   EQU   $101E       Timer: Output Compare Register 5.
_TCTL1  EQU   $1020       Timer: Control Register 1.
_TCTL2  EQU   $1021       Timer: Control Register 2.
_TMSK1  EQU   $1022       Timer: Interrupt Mask Register 1.
_TFLG1  EQU   $1023       Timer: Interrupt Flag Register 1.
_TMSK2  EQU   $1024       Timer: Interrupt Mask Register 2.
_TFLG2  EQU   $1025       Timer: Interrupt Flag Register 2.
_PACTL  EQU   $1026       Timer: Pulse Accumulator Control.
_PACNT  EQU   $1027       Timer: Pulse Accumulator Count Register.
_SPCR   EQU   $1028       SPI: Control Register.
_SPSR   EQU   $1029       SPI: Status Register.
_SPDR   EQU   $102A       SPI: Data Register.
_BAUD   EQU   $102B       SCI: Baud Rate Register.
_SCCR1  EQU   $102C       SCI: Control Register 1.
_SCCR2  EQU   $102D       SCI: Control Register 2.
_SCSR   EQU   $102E       SCI: Status Register.
_SCDR   EQU   $102F       SCI: Data Register.
_ADCTL  EQU   $1030       A/D: Control Register.
_ADR1   EQU   $1031       A/D: Result Register 1.
_ADR2   EQU   $1032       A/D: Result Register 2.
_ADR3   EQU   $1033       A/D: Result Register 3.
_ADR4   EQU   $1034       A/D: Result Register 4.
_OPTION EQU   $1039       System Configuration Option.
_COPRST EQU   $103A       Arm/Reset COP Timer Circuitry.
_PPROG  EQU   $103B       EEPROM Programming Register.
_HPRIO  EQU   $103C       Highest Priority I bit.
_INIT   EQU   $103D       RAM and I/O Mapping Register.
_TEST1  EQU   $103E       Testing Functions Control Register.
_CONFIG EQU   $103F       System Configuration.
_DPRA   EQU   $1800       PIA: Data direction / Periferial register A.
_CRA    EQU   $1801       PIA: Control register A.
_DPRB   EQU   $1802       PIA: Data direction / Periferial register B.
_CRB    EQU   $1803       PIA: Control register B.
_XRAM   EQU   $2000       External RAM $2000..$3FFF (8192byte).
_EEPROM EQU   $B600       Internal EEPROM $B600..$B7FF (512byte).
_EPROM  EQU   $E000       User EPROM $E000..$FFFF (8192byte).
```

```
************************* HC11 INTERRUPT VECTORS ******************************
_ISCI   EQU   $FFD6       Serial Communication Interface (SCI)
_ISPI   EQU   $FFD8       Serial Peripherial Interface (SPI)
_IPAIE  EQU   $FFDA       Pulse Accumulator Input Edge
_IPAO   EQU   $FFDC       Pulse Accumulator Overflow
_ITOF   EQU   $FFDE       Timer Overflow
_ITOC5  EQU   $FFE0       Timer Output Compare 5
_ITOC4  EQU   $FFE2       Timer Output Compare 4
_ITOC3  EQU   $FFE4       Timer Output Compare 3
_ITOC2  EQU   $FFE6       Timer Output Compare 2
_ITOC1  EQU   $FFE8       Timer Output Compare 1
_ITIC3  EQU   $FFEA       Timer Input Capture 3
_ITIC2  EQU   $FFEC       Timer Input Capture 2
_ITIC1  EQU   $FFEE       Timer Input Capture 1
_IRTI   EQU   $FFF0       Real Time Interrupt (RTI)
_IIRQ   EQU   $FFF2       IRQ Pin
_IXIRQ  EQU   $FFF4       XIRQ Pin (Pseudo Non-Maskable Interrupt)
_ISWI   EQU   $FFF6       Software Interrupt (SWI)
_IILLOP EQU   $FFF8       Illegal Opcode Trap
_ICOP   EQU   $FFFA       COP Failure (Reset)
_ICLKM  EQU   $FFFC       Clock Monitor Fail (Reset)
_IRESET EQU   $FFFE       Reset


************************* INTERRUPT VECTOR TABLE ******************************
        ORG   _ITOC1      Set Output Compare 1 interrupt
        FDB   _SCHINT        to the scheduler interrupt rutine.
        ORG   _IRESET     Upon reset
        FDB   _START         go to START.


        ORG   _EPROM
************************* TASK SCHEDULE ***************************************
SCHTAB  FDB   KBD        Gonilnik tipk (stalno odprta/zaprta).
        FDB   LED        Gonilnik led diod (indikatorja zahtev).
        FDB   MOT        Gonilnik motorja.
        FDB   LOGIC      Koliko časa naj bodo vrata odprta?
        FDB   SENZOR     Gonilnik senzorja.
        FDB   DEFAULT    Nastavitev parametrov ob resetu.
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
        FDB   SCHRTS     Void task.            ~ =  5
SCHRTS  rts              Void task.


************************* THE SCHEDULER INTERRUPT RUTINE (~=80) ***************
_SCHINT  ldaa  SCHTST        Preberi testno spremenljivko.
        beq    SCHOK         Skoci, ce je predhodna prekinitev koncana,
SCHERR  bra    SCHERR        sicer ostani  -- prekoracitev rezine!
SCHOK   inc    SCHTST        S testno spremenljivko oznaci tekoco prekinitev.
        ldd    _TOC1         Load timer output compare register 1,
        addd   #1200         increment it by time slice
        std    _TOC1         and restore it to TOC1
        ldx    SCHPTR        Get pointer to current 1/64s period task
        ldx    0,x           Get the task entry address.
        ldaa   #$80          Clear currentlypending
        staa   _TFLG1        OC1F interrupt and
        cli                  allow interrupts.
        jsr    0,x           EXECUTE THE TASK
        ldaa   SCHPTR+1      Get high byte of SCHPTR,
        adda   #2            increment it,
        anda   #%00011110    overlay 0's
        staa   SCHPTR+1      and restore it to SCHPTR.
        clr    SCHTST        Reset test bit to indicate end of interrupt.
        rti                  Return from OC1 interrupt
```

```
************************* ENABLE THE SCHEDULER ********************************
SCHON  clr   SCHTST      6   Reset test bit to indicate no interrupt running.
       ldx   #SCHTAB     3   Initialize pointer
       stx   SCHPTR      5    to first 1/64s period tasks.
       ldd   _TCNT       5   Load free running counter
       addd  #1200       4    increment it by time slice
       std   _TOC1       5    and store it to output compare register 1.
       ldaa  #$80        2   Select OC1,
       staa  _TFLG1      4    clear any pending OC1F,
       staa  _TMSK1      4    enable OC1 interrupt and
       cli               2    start interrupts.
       rts               5   Return(Scheduler running).


************************* SYNCHRONIZE WITH SCHEDULER **************************
*  Call this subroutine from any non-realtime routine to request D<1100
*  uninterrupted subsequent maschine cycles. Upon return, you have the
*  requested number of uninterrupted cycles, plus D extra cycles. Use D=0 to
*  find out the number of cycles left until next scheduler interrupt.
*******************************************************************************
SCHSYN   pshx              4  Push X.
         addd  #$0020      4  Add the system overhead of 32 machine cycles.
         pshb              3  Push number of machine cycles
         psha              3     from D to stack
         tsx               3     and make X point to it.
SCHWAI   ldd   _TOC1       4  Load output compare register,
         subd  _TCNT       5     subtract free running counter
         subd  0,X         6     and number of cycles.
         bmi   SCHWAI      3  Loop back if not enough cycles until interrupt.
         pulx              5  Dummy pull X to clear D from stack.
         pulx              5  Restore X.
         rts               5  Return(D=extra cycles left).


************************* RESET ***********************************************
_START   lds   #$3FFF      Define stack segment.
         jsr   INIT        Initialize RAM variables.
         jsr   SCHON       Start scheduler.
         jsr   MAIN        Call MAIN -- there should be no return.
HALT     bra   HALT        Dead loop!


************************* INITIALIZE RAM VARIABLES ****************************
INIT     ldx   #$0000         Clear.
       stx   LED1             Vrata vedno odprta.
       stx   LED2             Vrata vedno zaprta.
       stx   POLZ             Minimalna hitrost pri zapiranju vrat.
       stx   POLZ2            Minimalna hitrost pri odpiranju vrat.
       stx   MOTL             Spremenljivka -> zapiri vrata.
       stx   MOTD             Spremenljivka -> odpiri vrata.
       stx   SENZ             Spremenljivka senzorja.
       stx   URA              Koliko casa naj bodo vrata odprta, pri sprozitvi senzorja?
         ldx   #KBDB+2        Clear the
         stx   KBDB            keyboard
         stx   KBDE            buffer.
         clr   KBDKEY         Indicate no key pressed.
         ldx   #TRAB+2        Clear the
         stx   TRAB            SCI transmitting
       stx   TRAE             buffer.
         ldx   #RECB+2        Clear the
         stx   RECB            SCI receiveing
         stx   RECE            buffer.
         clr   _CRA           Enable Data direction register A and
         clr   _CRB            Data direction register B.
         ldaa  #%11000000         Zg. bita sta izhodna, ostali vhodni!
         staa  _DPRA
         ldaa  #$FF           Set all bits of port B
         staa  _DPRB           to output.
         ldaa  #$04           Enable Data registers
         staa  _CRA            of port A and
         staa  _CRB            port B.
       clr   _DPRA
       ldaa   #47             Nastavi vrednost pri kateri
         staa   _DPRB         motor miruje!!
         ldaa  _OPTION        Get system options,
         oraa  #%10000000      set A/D power on
         staa  _OPTION         and store options.
```

```
        inc   CLEAR           Oznaci Reset ->>ponastavi stevec na prvotno vrednost
          rts                 Return().


************************* MAIN PROGRAM ****************************************
MAIN    jsr     KBDCHR        Zahteva po vedno odprtih/zaprtih vratih.
        beq     NAPREJ        Ne, nadaljuj.
        jsr     TIPKI         Da, zapri/odpri vrata.
NAPREJ  ldaa    LED1          Ali je senzor
        adda    LED2          onemogocen?
        bne     MAIN          Da, vrni se na zacetek programa.
        ldaa    SENZ          Ne, preveri, ce je senzor aktiviran.
        beq     SENAPREJ      Ne, vrni se na zacetek.
        inc     MOTD          Da, odpri vrata.
SENAPREJ        bra     MAIN  Vrni se na zacetek.


************************* KBD - Gonilnik tipkovnice ***************************
* Opravilo "skenira" tipkovnico in prispele znake shranjuje v medpomnilnik
* KBDB do KBDE. Veckratno branje iste tipke je onemogoceno z KBDKEY.
*******************************************************************************
KBD     ldaa    _DPRA         Poglej PIA-o, port A
        oraa    #%11001111    Izloci tipki za stalno odprta/zaprta vrata.
        coma                  Tipka pritisnjena?
        beq     NEVELJ        Ne, izbrisi KBDKEY.
        ldab    KBDKEY        Da, preveri, ce je tipka ze prebrana.
        bne     KONEC         Da, zakluci.
        inc     KBDKEY        Oznaci KBDKEY.
******************* Vpis v KBD medpolnilnik ***********************************
        ldx     KBDE
        sta     0,x
        inx
        cpx     #KBDE
        bne     SKOK
        ldx     #KBDB+2
SKOK    cpx     KBDB
        beq     NEVELJ
        stx     KBDE
        rts
NEVELJ  clr     KBDKEY
KONEC   rts


********************* Branje iz KBD medpolnilnika *****************************
* Rutina prebere znak iz medpolnilnika in ga spravi v akomuletor A. V primeru
* praznega medpolnilnika rutina vrne vrednost Z=1, uspesno branje pa
* oznaci z Z=0.
*******************************************************************************
KBDCHR   pshx
        ldx     KBDB          Logicni zacetek bufferja ->mesto praznjenja.
        cpx     KBDE          Preveri, ce je buffer prazen:
        beq     KONEC2        Da, koncaj rutino Z=1
        ldaa    0,x           Ne, preberi znak z mesta praznjenja.
        inx
        cpx     #KBDE         Preveri, ce smo na fiz. dnu bufferja:
        bne     SKOK2         Ne, zapisi nov naslov praznjenja.
        ldx     #KBDB+2       Da, novo mesto praznjenja je fiz. zacetek bufferja.
SKOK2   stx     KBDB          nov naslov praznjenja.
KONEC2  pulx
        rts


************************ LED - Gonilnik svetlecih diod ************************
* Gonilnik preverja informacije o vedno odprtih ali zaprtih vratih. Glede
* na te ingormacije prizge ali ugasne diodi.
*******************************************************************************
LED     ldaa    LED1          Preveri, ali je znak ze prizig zelene diode
        beq     OFFLED1       Ne, ugasni diodo.
        ldaa    _DPRA         Da, poglej A port PIA-e in
        oraa    #%10000000      postavi prvi bit na 1, da prizgez diodo ter
        staa    _DPRA           shrani na A port PIA-e.
        bra     DRUGA         Preveri rdeco diodo.
OFFLED1 ldaa    _DPRA         Poglej A port PIa-e in
        anda    #%01111111      postavi prvi bit na 0, da ugasnes diodo ter
        staa    _DPRA           shrani na A port PIA-e.
DRUGA   ldaa    LED2          Poglej ali je znak za prizig rdece diode
```

```
        beq     OFFLED2         Ne, ugasni diodo.
        ldaa    _DPRA           Da, poglej A port PIA-e in
        oraa    #%01000000        postavi drugi bit na 1, da prizgez diodo ter
        staa    _DPRA             shrani na A port PIA-e.
        rts
OFFLED2 ldaa    _DPRA           Poglej A port PIa-e in
        anda    #%10111111        postavi drugi bit na 0, da ugasnes diodo ter
        staa    _DPRA             shrani na A port PIA-e.
        rts


************************* MOT - Gonilnik Motorja ****************************
* Gonilnik zapira in odpira vrata. Pri zapiranju ali odpiranju gonilnik vsakic,
* ko se le ta izvede, poveca STEVEC za ena in nato preverja, kdaj se morajo vrata
* upocasniti ali pa se vrteti z min. hitrostjo - POLZ. Motor miruje pri vrednosti
* 67, pri $EE se vrata zapirajo z max. hitrostjo, pri $24 pa se odpirajo z min.
* hitrostjo.
*******************************************************************************
MOT     ldaa    MOTL            Preveri ali je znak za zapiranje?
        beq     MOT2            Ne, nadaljuj.
        ldaa    POLZ            Da, ali naj se vrata zapirajo z minimalno hitrostjo.
        bne     CAKAJ           Da, vrata se zapirajo z min. hotrostjo.
        inc     STEVEC          Ne, povecaj stevec
        ldaa    STEVEC            in poglej ce je ze cas za
        cmpa    #$24                upocasnjevanje.
        beq     POCASI          Da, zacni upocasnjevati motor.
        ldaa    #$EE            Ne, vrata naj se zapirajo z max. hitrosjo (FF je max).
        staa    _DPRB
        rts
POCASI  dec     STEVEC          Zmanjsaj stevec za 1,
        ldaa    _DPRA             nalozi PIA-o,
        oraa    #%11110111        in preveri, ce so vrata
        coma                      ze zaprta.
        bne     ODPRTA                  Da, miruj.
        ldaa    _DPRB           Ne, nalozi B port PIA-e in
        deca                      upocasni zapiranje.
        deca
        deca
        deca
        deca
        deca
        staa    _DPRB
        cmpa    #142            Poglej ce so vrata ze dovolj upocasnjena.
        beq     POCASI2         Da, oznaci POLZ (min. hitrost).
        rts                     Ne, koncaj.
POCASI2 inc     POLZ            Oznaci POLZ (min. hitrost).
        rts
CAKAJ   ldaa    _DPRA           Poglej ali je zaprl
        anda    #%00001000        vrata do konca (stikalo-vrata zaprta).
        beq     ODPRTA          Da, ustavi motor.
        rts
ODPRTA  clr     MOTL            Resetiraj zapiranje,
        clr     POLZ              min. hitrost ter
        ldaa    #67               ustavi motor.
        staa    _DPRB
        rts

MOT2    ldaa    MOTD            Znak za odpiranje vrat?
        beq     MIRUJ           Ne, resetiraj parametre.
        ldaa    POLZ2           Da, ali naj se vrata odpirajo z min. hitrostjo.
        bne     CAKAJ2          Da, vrata naj se odpirajo z min. hotrostjo.
        dec     STEVEC          Ne, zmanjsaj stevec
        beq     POCASIZ           in poglej ce je cas za upocasnjevanje.
        ldaa    #24             Ne, vrata naj se odpirajo z max. hirtostjo (00 je max.).
        staa    _DPRB
        rts
POCASIZ inc     STEVEC          Povecaj stevec za 1,
        ldaa    _DPRA             nalozi PIA-o,
        oraa    #%11111011        in preveri, ce so vrata ze
        coma                      odprta.
        bne     MIRUJ           Da, miruj.
        ldaa    _DPRB           Ne, nalizi B port PIA-e in
        inca                      upocasni odpiranje.
        inca
        staa    _DPRB
        cmpa    #44             Poglej ce so vrata ze dovolj upocasnjena.
        beq     POCASIZ2                Da, oznaci POLZ2 (min. hitrost).
```

```
       rts                    Ne, koncaj.
POCASIZ2inc    POLZ2          Oznaci POLZ2 (min. hitrost).
       rts
CAKAJ2 ldaa    _DPRA          Poglej ali je odpru vrata
       anda    #%00000100       vrata do konca (stikalo-vrata odprta).
       beq     MIRUJ          Da, ustavi motor.
       rts
MIRUJ  ldaa    #67            Ustavi motor,
       staa    _DPRB
       clr     MOTD             resetiraj odpiranje in
       clr     POLZ2            resetiraj min. hitrost.
       rts


********************* SENZOR - Gonilnik senzorja ******************************
* Opravilo spremlja senzor in odpre vrata, ce je senzor aktiviran in ni
* zahtev po stalno zaprtih ali odprtih vratih. Opravilo ne reagira, ce je senzor
* prebran 2x zapored.
*******************************************************************************
SENZOR ldaa    SENZ           Poglej ali je senzor ze bil aktiviran.
       bne     JEZE           Da, koncaj opravilo.
       ldaa    _DPRA          Ne, poglej ali je senzor zaznal gibanje.
       oraa    #%11111100     Maskiraj zadnja bita,
       coma
       bne     PREMIK         Da, zaznano je gibanje.
       rts
PREMIK ldaa    LED1           Preveri ali je aktivna zahteva po
       adda    LED2           stalno odprtih/zaprtih vratih.
       bne     JEZE           Je, koncaj opravilo.
       inc     SENZ           Oznaci spremenljivki SENZ,
       clr     MOTL             prekini morebitno zapiranje vrat,
       clr     POLZ             prekini gibanje z minimalno hitrostjo in
       inc     MOTD             povzrovi odpiranje vrat!
JEZE   rts


******************************** TIPKI ****************************************
* Ob pritisku tipke rutina prizge diodo (za vedno odprta ali vedno zaprta
* vrata), ter odpre ali zapre vrata. Rutina pricakuje v akom. A vrednost iz
* medpolnilnika!
*******************************************************************************
TIPKI  cmpa    #$10           Preveri, ce je zahteva po vedno odprtih vratih.
       bne     ZAPRTA         Da, prizgi LED za vedno odprta vrata in odpri vrata.

       ldaa    LED2           Ne, preveri ce je ze bila zahteva po vedno zaprtih vratih.

       bne     SOZAPRTA              Da, izbrisi informacijo za vedno zaprta vrata.
       clr     LED1           Ne, izbrisi informacijo za vedno odprta vrata,
       inc     LED2             oznaci informacijo za vedno zaprta vrata ter
       inc     MOTL             zapri vrata.
       rts
SOZAPRTAclr    LED2           Izbrisi informacijo za vedno zaprta vrata ter
       rts                      koncaj rutino.
ZAPRTA ldaa    LED1           Preveri ce je ze bila zahteva po vedno odprtih vratih.
       bne     SOODPRTA              Da, izbrisi informacijo za vedno odprta vrata.
       clr     LED2           Ne, izbrisi informacijo za vedno zaprta vrata,
       inc     LED1             oznaci informacijo za vedno odprta vrata ter
       inc     MOTD             odpri vrata.
       rts
SOODPRTAclr    LED1           Izbrisi informacijo za vedno odprta vrata ter
       rts                      koncaj rutino.


*************** LOGIC - koliko casa so vrata odprta ***************************
* Opravilo opazuje vrata, ko so le ta odprta. Vsakic, ko se opravilo izvede se
* pri odprtih vratih poveca stevec URA za ena in ga primerja z vrednostjo FF.
* Ko stevec doseze to vrednost so vrata odprta ze dovolj casa in jih
* opravilo zapre. Pri vsaki sprozitvi senzorja opravilo zbrise stevec URA.
*******************************************************************************
LOGIC  ldaa    LED1           Preveri ali je zahteva po vedno odprtih ali
       adda    LED2             zaprtih vratih.
       bne     NESTEJ         Da, zbrisi stetje in koncaj opravilo.
       ldaa    _DPRA          Ne, nalozi port A PIA-e,
       oraa    #%11110111       izloci 4-ti bit, ter
       coma                     preveri ce so vrata odprta.
       bne     NESTEJ         Ne, zbrisi stetje in koncaj opravilo.
       ldaa    SENZ           Poglej, ce je aktiviran senzor.
```

```
       beq    STEJ           Ne, pricni steti cas, ko so vrata odprta.
       clr    URA            Da, resetiraj stetje
       clr    SENZ             in informacijo o senzorju.
STEJ   inc    URA            Povecaj stetje za 1,
       ldaa   URA              nalozi vrednost v akomulator in
       cmpa   #$FF             poglej ali je caz za zapreti vtata.
       beq    ZJEPRI         Da, zapri vrata.
       rts                   Ne, koncaj rutino.
ZJEPRI inc    MOTL           Zapri vrata.
NESTEJ clr    URA            Zbrisi stetje ter
       rts                     koncaj opravilo.


****************** DEFAULT - nastavitev parametrov ob resetu *****************
* Opravilo v primeru reseta, odpre vrata ter nastavi vrednost stevca na prvotno
* vrednost.
*******************************************************************************
DEFAULTldaa   CLEAR          Ali se je zgodil reset?
       beq    DELOV          Ne, koncaj opravilo.
       ldaa   _DPRA          Nalozi A port PIA-e,
       oraa   #%11111011       izloci 3. bit ter
       coma                    preveri ali so vrata odprta.
       bne    APERTA         Da, nastavi stevec.
       ldaa   #20            Ne, odpri vrata.
       staa   _DPRB
       rts
APERTA clr    CLEAR
       ldaa   #$00           Nastavi stevec na
       staa   STEVEC           zacetno vrednost in
       ldaa   #67              ustavi motor.
       staa   _DPRB
DELOV  rts

         ORG   _XRAM


************************* KERNEL VARIABLES (7 bytes) *************************
SCHPTR   RMB   2           16 bit pointer to current task in SCHTAH.
SCHTST   RMB   1           Test task on exceeding time slice.
************************* DRIVER VARIABLES (100 bytes) ************************
KBDB     RMB   3+8           KBD buffer
KBDE     RMB   2              with a capacity of 8 characters.
KBDKEY   RMB   1             KBD status variable: keypressed?
TRAB     RMB   3+64          SCI transmitting buffer
TRAE     RMB   2              with a capacity of 64 characters.
RECB     RMB   3+8           SCI receiveing buffer
RECE     RMB   2              with a capacity of 8 characters.
LED1   RMB   1               Vrata vedno odprta.
LED2   RMB   1               Vrata vedno zaprta.
STEVEC RMB   1               Kdaj se naj vrata zacnejo upocasnjevati?
MOTL   RMB   1               Spremenljivka -> zapri vrata.
MOTD   RMB   1               Spremenljivka -> odpiri vrata.
POLZ   RMB   1               Minimalna hitrost pri zapiranju vrat.
POLZ2  RMB   1               Minimalna hitrost pri odpiranju vrat.
SENZ   RMB   1               Spremenljivka senzorja.
URA    RMB   1               Koliko casa naj bodo vrata odprta, pri sprozitvi senzorja?
CLEAR  RMB   1               Clear.
************************* VARIABLE HEAP ***************************************
HEAP     RMB   1

       END
```

7 Seznam uporabljenih virov

1. Tadej Tuma, Mikrokrmilniški sistemi programiranje za družino HC11, Založba FE in FRI, Ljubljana, 2000
2. Robert Rostohar, Razvojno mikrokrmilniško okolje M68HC11 za pedagoške namene, Magistrska naloga, Uni. v Ljubljani, Fak. za elektrotehniko, 2000
3. Horst Rohlfing, Harry Schmidt, Priročnik za elektrotehniko in elektroniko, Tehniška založba Slovenije, 2000
4. Svet Elektronike, številka 10, strani 59-60, 1999
5. Spletne strani, http://www.scintilla.utwente.nl/~arjan/Leisure/Electronics/DAConverters/ZN426.pdf
6. Spletne strani, http://www.ce.chalmers.se/undergraduate/D/EDA230/datablad/tl072.pdf
7. Spletne strani, http://webserver.qtopto.com/oi/4n2x.pdf
8. Spletne stranu, http://www.st.com/stonline/books/pdf/docs/1530.pdf

8 Izjava

Izjavljam, da sem zaključno nalogo samostojno izdelal pod vodstvom mentorja doc. dr. Tadeja Tume. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Martin Mlakar