Cuprins

Introducere

Capitolul I- ***Studiu privind stadiul roboților mobile***

1.1 Roboți mobili

1.2 Clasificarea roboților mobili

1.3 Utilizări ale roboților mobili

1.4 Structura unui robot mobil

1.5 Acționarea roboților mobili

1.6 Sistemul senzorial

1.7 Tipuri de roboti mobili

Capitolul II- ***Studiul dinamicii robotului mobil cu roți***

2.1 Model cinematic

Capitolul III- ***Proiectarea elementelor constructive***

3.1 Elemente componente ale minirobotului

3.2 Placa minirobotului

3.3 Motoarele de curent continuu

3.4 Senzorul

Capitolul IV- ***Proiectarea interfețelor de comunicare***

4.1 PX-400 Cutie Programată In-Sistem cu interfață serială

4.2 Interfața SPI

4.3 Cabluri de transfer a datelor

Capitolul V- ***Programarea robotului***

5.1 Instalarea AVR Studio

5.2 Crearea programului C în AVR Studio

Bibliografie

Introducere

Unul din cele mai importante aspecte în evoluția ființei umane este folosirea uneltelor care să simplifice munca fizică. În aceasta categorie se înscriu şi roboții, ei ocupând totuşi o poziție privilegiată datorită complexității lor.

Noțiunea de robot datează de peste 4 mii de ani. Omul şi-a imaginat dispozitive mecanizate inteligente care să preia o parte însemnata din efortul fizic depus. Astfel a construit jucării automate si mecanisme inteligente sau şi-a imaginat roboții in desene, carti, filme "SF" etc.

Revoluția informatică a marcat saltul de la societatea industrializată la societatea avansat informatizată generând un val de înnoiri în tehnologie şi în educație.Acest lucru a dus şi la apariția roboților

Termenul "robot" a fost folosit in 1920 de cehul Karel Capek într-o piesa numită "Robotul universal al lui Kossum". Ideea era simplă: omul face robotul după care robotul ucide omul. Multe filme au continuat sa arate că roboții sunt maşinării dăunătoare si distrugătoare.

Revoluția informatică a marcat saltul de la societatea industrializată la societatea avansat informatizată generând un val de înnoiri în tehnologie şi în educație permițând realizarea de roboți.

Roboții oferă beneficii substantiale muncitorilor, industriilor si implicit țărilor. In situatia folosirii în scopuri paşnice, roboții industriali pot influența pozitiv calitatea vieții oamenilor prin înlocuirea acestora in spații periculoase, cu conditii de mediu daunatoare omului, cu conditii necunoscute de exploatare etc.

Domeniile de aplicare a tehnicii roboților se lărgesc mereu, ei putând fi utilizați în industrie, transporturi şi agricultură, în sfera serviciilor, în cunoasterea oceanului şi a spatiului cosmic, în cercetarea ştiințifică etc.

Lucrarea prezintă modul de proiectare şi realizare a unui minirobot echipat cu microcontroler ATmega8-16PI şi diferiți senzori aducând o contribuție la dezvoltarea bazei teoretice şi practice

de studiu a microcontrollerelor Atmel şi a posibilităților nelimitate de dezvoltare de aplicații în domeniul roboticii.

CAPITOLUL I
STUDIU PRIVIND STADIUL ROBOȚILOR MOBILI

1.1 Roboți mobili

Robotul mobil este un sistem complex care poate efectua diferite activități într-o varietate de situații specifice lumii reale. El este o combinație de dispozitive echipate cu servomotoare şi senzori (aflate sub controlul unui sistem ierarhic de calcul) ce operează într-un spațiu real, marcat de o serie de proprietăți fizice (de exemplu gravitația care influențează mişcarea tuturor roboților care funcționează pe pământ) şi care trebuie să planifice mişcările astfel încât robotul să poată realiza o sarcină în funcție de starea inițială a sistemului şi în funcție de informația apriori existentă, legată de mediul de lucru.

Succesul în îndeplinirea acestor sarcini depinde atât de cunoştințele pe care robotul le are asupra configurației inițiale a spațiului de lucru, cât şi de cele obținute pe parcursul evoluției sale.

Problemele specifice ce apar la roboții mobili ar fi următoarele: evitarea impactului cu obiectele staționare sau în mişcare, determinarea poziției şi orientării robotului pe teren, planificarea unei traiectorii optime de mişcare.

În cazul unui sistem robotic automat distribuit pozițiile spațiale sunt de o extremă importanță şi de ele depinde îndeplinirea scopurilor dorite şi funcționarea întregului sistem. Cu alte cuvinte, robotul trebuie să fie capabil să-şi planifice mişcările, să decidă automat ce mişcări să execute pentru a îndeplini o sarcină, în funcție de aranjamentul momentan al obiectelor din spațiul de lucru.

Planificarea mişcărilor nu constă dintr-o problemă unică şi bine determinată, ci dintr-un ansamblu de probleme dintre care unele sunt mai mult sau mai puțin variante ale celorlalte.

Evitare coliziunii cu obstacole fixe sau mobile (de exemplu alți roboți mobili) aflate în spațiul de lucru al robotului se poate face prin mai multe metode:realizarea unei apărători mecanice care prin deformare opreşte robotul, folosireasenzorilor care măsoară distanța până la obstacolele de pe direcția dedeplasare, folosirea senzorilor de proximitate, folosirea informațiilor corelate de la mai multe tipuri de senzori.

Localizarea obiectelor se poate realiza şi prin contact fizic, dar acesta impune restricții asupra vitezei de mişcare a structurii manipulate. Contactul fizic dintre robot şi obiectele din mediu generează forțe de reacțiune care modifică starea robotului. Vitezele mari de lucru fac ca efectele dinamice ale unui contact fizic cu obstacole sau obiecte manipulate să fie riscante (pot duce la deteriorarea obiectelor sau a robotului).

Navigarea robotului este posibilă şi fără o determinare a poziției şi orientării față de un sistem de coordonate fix, dar această informație este utilă pentru sisteme de comandă a mişcării. Dintre metodele de navigație mai des utilizate se pot menționa: măsurarea numărului de rotații făcute de roțile motoare, folosirea de acceleratoare şi giroscoape, geamanduri electromagnetice instalate în teren, semnalizatoare pasive sau semipasive de tip optic sau magnetic.

1.2 Clasificarea roboților mobili

Roboții mobili se clasifică astfel

- În funcție de dimensiuni: macro, micro şi nano-roboți.

• În funcție de mediul în care acționează: roboți tereştri – se deplasează pe sol, roboți subacvatici – în apă, roboți zburători – în aer, roboți extratereştri – pe solul altor planete sau în spațiul cosmic;

• În funcție de sistemul care le permite deplasarea în mediul în care acționează există de exemplu pentru deplasarea pe sol

1. roboți pe roți sau şenile
2. roboți păşitori: bipezi, patrupezi, hexapozi, miriapozi;
3. roboți târâtori: care imită mişcarea unui şarpe, care imită mişcarea unei râme etc.;
4. roboți săritori, care imită deplasarea broaştelor, cangurilor etc.;
5. roboți de formă sferică (se deplasează prin rostogolire) etc.

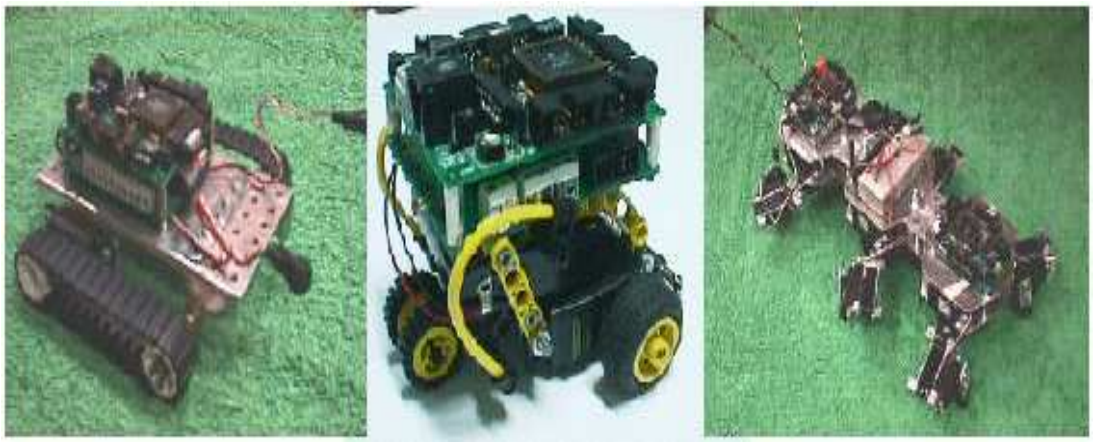


Figura 1- Diferite tipuri de miniroboți.

1.3 Utilizări ale roboților mobili.

Utilizările pentru care au fost, sunt şi vor fi concepuți roboții mobili sunt dintre cele mai diverse . Mulți roboți din zona micro îşi găsesc utilizarea în medicină, fiind capabili să se deplaseze de-a lungul vaselor şi tuburilor corpului omenesc, în scopul investigațiilor, intervențiilor chirurgicale, dozării şi distribuirii de medicamente etc. La fel de spectaculoase sunt şi multe utilizări ale macro-roboților:

• În domeniul industrial, agricol, forestier: în domeniul industrial roboţii mobili sunt reprezentaţi de AGV-uri (Automated-Guided Vehicles), vehicule pe roţi, cu ghidare automată, care transportă şi manipulează piese, constituind o alternativă flexibilă la benzile de montaj; în agricultură există tractoare şi maşini agricole fără pilot, capabile să execute singure lucrările pe suprafeţele pentru care au fost programate; în domeniul forestier roboţii mobili pot escalada copacii înalţi

• În domeniul militar: este luată în considerare de către armata americană perspective înlocuirii soldaţilor combatanţi cu roboţi, pentru a reduce riscul pierderilor umane în luptă; roboţi mobili de cele mai ingenioase şi robuste configuraţii sunt aruncaţi în clădi şi incinte din zone de conflict, în scopuri de investigare şi chiar anihilare a inamicului;

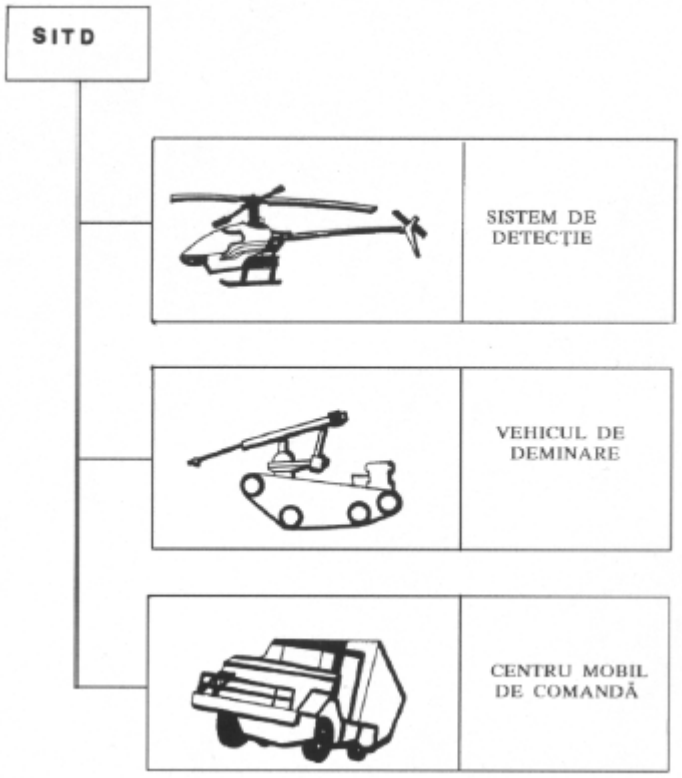


Figura 2-Sistem integrat telecomandat pentru deminare.

- În domeniul utilităților publice: una dintre cele mai utile şi economice utilizări ale roboților mobili o reprezintă inspectarea conductelor de combustibili gazoşi şi lichizi şi a canalelor de canalizare. De exemplu rețeaua de canalizare a Germaniei însumează 400.000 km, iar inspectarea şi curățirea acesteia presupune costuri de 3□6 Euro pe metru. Numai 20% din conducte sunt accesibile, iar utilizarea roboților poate reduce costurile cu un sfert.
- În domeniul distractiv şi recreativ: sunt roboții-jucării, roboții pentru competiții
- În domeniul serviciilor: Există posibilități deosebit de largi de implementare. Sunt roboți pentru: deservirea bolnavilor în spitale; ajutorarea persoanelor bătrâne sau cu diferite handicapuri; ghidarea şi informarea publicului în muzee aspirarea şi curățirea încăperilor; spălarea geamurilor şi a pereților clădirilor;
- În domeniul securității: Multe operații de inspectare şi dezamorsare a unor obiecte şi bagaje suspecte sunt executate de roboți;
- În domeniul operațiilor de salvare: Roboții salvatori (Rescue robots) sunt utilizați în operațiile de salvare a victimelor unor calamități: cutremure, incendii, inundații.

Roboții mobili au următoarele caracteristici comune:

1. structura mecanică este un lanț cinematic serie sau paralel respectiv tip "master-slave";
2. sistemul de acționare utilizat este electric pentru sarcini mici şi medii şi hidraulic pentru sarcini mari;
3. sistemul senzorial utilizează senzori interni (de turație, poziție, efort) la nivelul articulațiilor, senzori externi(camere TV) pentru scanarea mediului şi senzori de securitate(de proximitate, de prezență cu ultrasunete);
4. sistemul de comandă este ierarhizat, de obicei multiprocesor;
5. limbajele de programare utilizate sunt preluate de la roboții staționari.

1.4 Strucura unui robot mobil

Structura roboților mobili (RM) corespunde arhitecturii generale a roboților, având două părți:

- Structura mecanică, respectiv manipulatorul, care determină performanțele tehnice;
- Structura electronică, respectiv de comandă-control, care condiționează calitatea performanțelor.

Indiferent de generația robotului, probleme complexe apar la realizarea structurii mecanice de volum, greutate şi cost reduse, la transmiterea mişcării şi adaptarea la structura mecanică a motoarelor electrice şi hidraulice, la proiectarea mâinilor mecanice pentru a apuca obiecte de diferite forme. Referitor la structura electronică, posibilitățile actuale permit folosirea a câte unui microprocesor pentru comanda fiecărui grad de mobilitate, precum şi a altor microprocesoare specializate pentru tratarea semnalului senzorial. Robotul mobil interacționează cu mediul înconjurător prin structura sa mecanică, asigurând astfel deplasarea, poziționarea şi orientarea organului de execuție.

1.4.1 Structura mecanică a roboților mobili este formată din:

- sistemul de locomoție (pe şenile sau roți), prin care se asigură deplasarea robotului pe o suprafață de lucru (în cadrul unei autonomii sporite);
- sistemul de manipulare, care asigură poziționarea şi orientarea organului de lucru.

Robotul mobil în procesul de deplasare pe o anumuită traiectorie este caracterizat prin 3 funcți:

1. funcția de locomoție;
2. funcția de percepție-decizie;

3. funcția de localizare;

Funcția de locomoție cuprinde sistemul de acționare electric(de propulsie) şi sistemul de sprijinire(suspensie).Modalitățile de propulsare sunt dintre cele mai diverse,cum ar fii:pe roți, cu jet de apă, cu aer etc;

Roboții mobili pot fii dotați cu camera video sau alți senzori de percepere al mediului în care activează. Memoria robotului conținută in microcontroler înmagazinează cunoştințele necesare localizării tuturor segmentelor de traseu posibile

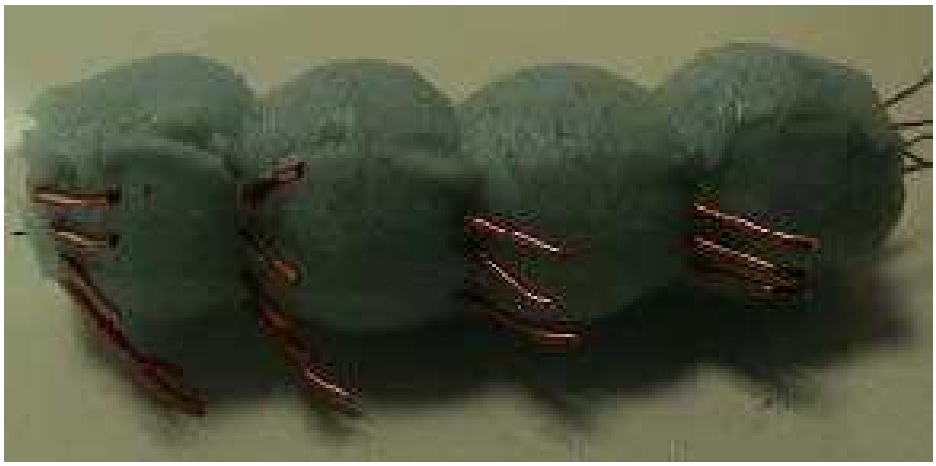


Figura 3- Locomoția viermelui de pământ.

- Locomoția viermelui de pământ este influențată de factori precum frecarea dintre module şi suprafață, de tipul perilor artificiali dispuşi pe module şi posibilitatea lor de penetrare a suprafețelor şi forțe inerțiale.
- Prototipul în starea de până acum are o deplasare neglijabilă fapt pentru care la realizarea mişcării au fost urmate două metode:

1. Fabricarea de picioruşe direcționale ce vor mima perii cheratinoşi ai râmei, fapt ce duce la o avansare prin agățare a acestora de suprafață

2.Propulsarea micro-robotului pe o suprafață direcțională precum ar fi catifeaua.

1.5 Acționarea roboților mobili

Se face cu motoare electrice de putere mică, cu moment de inerție redus,cu capacitate de suprasarcină, cu reductoare de raport mare (i>100) şi moment de inerție redus de tip procesional sau armonic.Se pot folosi şi unități integrate motor-reductor

Motoare electrice cu inerție redusă utilizate:

- motoare de curent continuu cu pahar sau indus disc;
- motoare sincrone cu magneți permanenți;
- motoare pas cu pas cu reluctanță variabilă cu indus pahar şi intrefier radial sau cu indus tip disc şi intrefier radial;
- motoare sincrone cu magneți permanenți;

Analiza performanțelor dinamice a acționărilor pe baza funcției de transfer $H_d(s)$ presupune determinarea parametrilor:

Precizia care se determină cu ajutorul funcției de transfer prin calculul erorii staționare:

$$g_{st} = lim_{s\to\infty}(s) = lim_{s\to\infty} s\frac{1}{1+Hd(s)}Ui(s) \quad (1)$$

$U_i(s)$= mărime de intrare.

Promtitudinea, care se calculează cu ajutorul locului de transfer considerând un compromis optim precizie-stabilitate pentru o margine de fază MΦ=25÷45^0.

Stabilitatea care se determină cu ajutorul locului de transfer.

Capacitatea de urmărire se poate aprecia cu ajutorul caracteristicii Bode.

Pentru a se ameliora performanțele dinamice ale acționărilor în anumite domenii de frecvență se utilizează elemente de corecție serie sau paralel având funcția de transfer

$$H_c = \frac{Hd(s)}{H1(s) \cdot H2(s)} \quad (2)$$

1.6 Sistemul senzorial

Robotul mobil este pus în situația de a desfăşura acțiuni similare cu cele ale operatorului uman. Acest lucru determină existența unor anumite dispozitive prin care să se culeagă informații din mediul de lucru, care să realizeze interacțiunea robot . mediu cu ajutorul unor caracteristici ale mediului sau ale obiectelor din mediu şi o unitate centrală care să prelucreze în timp real informația senzorială, să o transforme într-o formă utilă pentru sistemul de comandă.

Sistemul senzorial mai este numit şi sistem de măsurare. El asigură măsurarea unor mărimi fizice şi eventual perceperea unor modificări semnificative a acestor mărimi.

Datorită sistemului senzorial se pot pune în evidență şi caracteristicile geometrice şi chimice ale obiectelor din mediul de lucru. Senzorii datorită caracteristicilor pe care le au pot explora zona de lucru, zona de contact, cea apropiată, cea îndepărtată, iar senzorii foarte puternici chiar şi zone foarte îndepărtate. Informațiile culese cu ajutorul sistemului senzorial servesc la construirea unui model al lumii în care evoluează robotul, model funcție de care aceasta îşi va genera planurile de acțiuni viitoare.

În funcție de soluția constructivă, senzorii sunt pasivi sau activi. Senzorii activi folosesc caracteristicile intrinseci ale mediului, iar cei pasivi generează

modificări în mediu pentru a măsura unele caracteristici.

Alte două categorii de senzori cu care poate fi dotat un robot mobil sunt: senzori de mărimi interne ai robotului (poziție, viteză sau accelerația unor componente mecanice proprii) şi senzori de mărimi externe (greutate, formă, poziție, temperatură, culoare, etc ale obiectelor asupra cărora acționează robotul).

Caracteristicile senzoriale ale unui robot depind foarte mult de gradul său de autonomie, de aplicațiile pentru care a fost proiectat şi de tipul mediului de lucru.

În general percepția se realizează în două etape:

- conversia proprietăților fizice într-un semnal, de obicei electric;
- prelucrarea acestui semnal în vederea extragerii informației care interesează.

Se preferă o prelucrare preliminară în care senzorii robotului elimină zgomotele care perturbă semnalul util.

Există mai multe criterii de clasificare a senzorilor utilizați în sistemele de comandă ale robotului industrial:

1. după cum vine sau nu în contact cu obiectul a cărui proprietate fizică o măsoară, distingem:

• senzori cu contact;

• senzori fără contact;

2. după proprietățile pe care le pun în evidență:

• senzori pentru determinarea formelor şi dimensiunilor (pentru evaluarea în mediu de lucru);

• senzori pentru determinarea proprietăților fizice ale obiectelor (de forță, de cuplu, de densitate şi elastici);

• senzori pentru proprietăți chimice (de compoziție, de concentrație, analizatoare complexe);

• după mediul de culegere a informației:

a) senzorii pentru mediul extern;

b) senzorii pentru funcția internă;

• după distanța la care sunt culese informațiile avem senzori de contact.

1.7 Tipuri de roboți mobili:

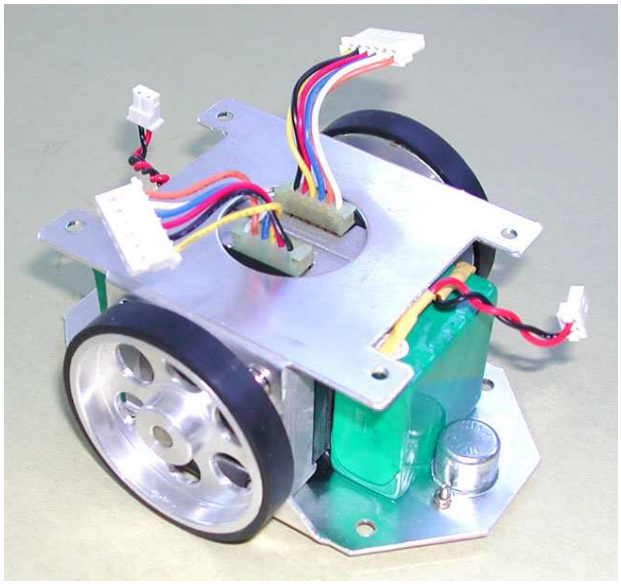


Figura 4-Minirobot AIRAT 2.

AIRAT 2 este un robot micromouse care foloseşte un procesor CPU 8051.AIRAT 2 foloseşte senzori pentru a o recepta când se întoarce înapoi. Placa CPU folosete o placă JS8051-A2. Placa JS8051-A2 este foarte bine construită.Foloseşte resurse externe de putere cum ar fii LCD, ADC, două ceasuri externe, auto-Flash scriere şi altele.

AIRAT 2 utilizeaza şase senzori oferindu-i astfel posibilitatea de a se deaplasa pe diagonală.Un simulator PC este prevăzut,oferindu-i posibilitatea utilizatorului de a întelege mai bine nivelul inalt de căutare algoritmică a mouseului.Codul sursa C este implementat astfel încat programatorul poate dezvolta mai usor altgoritmi care pot fi testati cu ajutorul unui simulator si apoi implementat mouseului.

În plus,LCD,comunicatie seriala,controlul mouse-ului precum si alte functii sunt furnizate sub forma de librarie si fisiere sursa.Pentru cei ce vor sa invete mouse-ul la un nivel inalt,AIRAT2 furnizeaza un mediu excelent de dezvoltare,teste algoritmice,precum si multe altele.

AIRAT 2 a aparut pe coperta publicatiei franceze »MICROS&ROBOTS ».

CARACTERISTICI AIRAT 2

- capabil de reglare proprie.Invață din mers.
- Foloseşte 6 senzori dindu-i posibilitatea de a se deplasa pe diagonala
- Uşor de asamblat/dezasamblat
- Port de reîncarcare
- Instructii de asamblare si manual al utilizatorului
- Include un simulator PC pt. accelerarea dezvoltarii
- Librarii,coduri sursa C
- AIRAT2 baterie(NiMh-450) Descriere

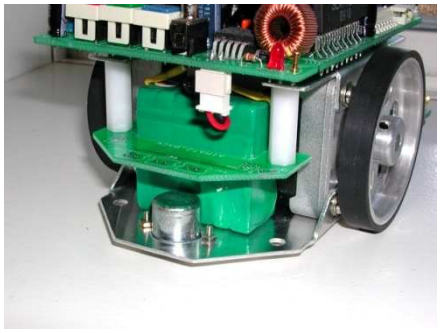


Figura 5-Bateria AIRAT 2.

Baterie NiMh.Marime 35X35X16mm

Putere 7.2 V(1.2X6), 450mAh

RCX 1.0

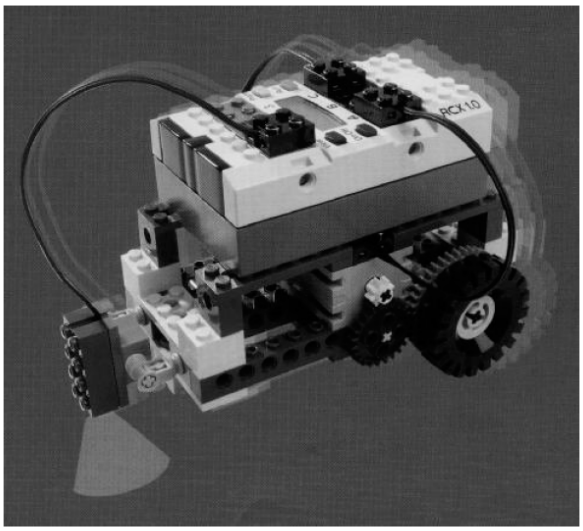


Figura 6-Minirobot RCX 1.0.

Una dintre principalele structuri de robot mobil, construită şi utilizată în cadrul testelor, este prezentată în figura . Include două roți active în partea din spate, acționată fiecare de câte un motor de curent continuu montat într-o piesă LEGO, şi o roată pasivă în partea din față. În vârful construcției este amplasată unitatea de comandă, RCX, conectată la cele două motoare (A şi C) în programele prezentate la senzorul de lumină din frunte (senzor 1 în programe). Acest senzor este constituit dintr-un LED (diodă luminiscentă), care generează un fascicul de lumină roşie şi un fotodetector care recepționeză lumina reflectată de podea şi generează la ieşire o tensiune electrică, proporțională cu intensitatea luminii reflectate. Roțile active sunt acționate prin intermediul a două trepte de angrenaje cu roți dințate.

Prima include un pinion cu 8 dinți, montat pe axul motorului (pinionul de culoare

deschisă din fig) şi o roată dințată cu 24 de dinți, iar a doua, este realizată prin angrenarea roții intermediare de 24 de dinți cu o roată de 40 de dinți, care antrenează axul roții active. Raportul de transmitere poate fi calculat cu formula:

$i = (24/8) * (40/24) = 5,$

Asigură o reducere de 5 la 1 a vitezei unghiulare a motorului şi o amplificare de 1 la 5 a momentului dezvoltat de motor. Direcția de deplasare a robotului este controlată,fie prin mişcarea roților active în sensuri opuse, fie prin deplasarea acestora cu viteze unghiulare diferite.

Epson a creat cel mai mic microrobot zburator din lume

Figura 7-Micro Flying Robot.

Tokio,Japonia,Noiembrie 17 Seiko Epson Corporation(“Epson”) au creat µFR(“Micro Flying Robot”),cel mai mic prototip zburator microrobot. Epson au creat µFR pentru a demonstra tehnologia micromecatronica care s-a dezvoltat in laboratoare in ultimii ani si pentru a explora posibititati pentru microroboti si dezvoltarea aplicatiilor pentru componentele tehnologice. Compania a prezentat ultimele oferte la Expozitia Internationala Robotica din 2003, care a avut loc la Tokyo Big Sight între 19 - 22, 2003.

Bazîndu-se pe propria tehnologie mecatronica,care este una din tehnologiile 100% apartinând companiei, Epson a dezvoltat si a vândut o familie de roboti cunoscută sub numele de EMRoS,incepand cu Monsieur, care a fost pus in vanzare in 1993 şi este prezent in cartea Recodrurilor ca fiind cel mai mic microrobot din lume.In Aprilie Epson a creat Monsieur – II- P,un prototip de microrobot care opereaza pe un motor ultra-subtire si

ultrasonic si un modul Bluetooth de tipo reductor care permite multiplelor unitati sa fie comandate prin telecomanda simultan.

Folosind acesti roboti, Epson deasemenea a realizat un robot tip teatru de balet..In acest sens Epson,a jucat un rol de pionerat in cercetarea si dezvoltarea microrobotilor precum şi în aplicatiile componentelor tehnologice.

µFR prezentat la expozitie, a provoacat levitarea prin folosirea unor elice ce au o mişcare tip contra-rotaţie propulsate de un motor foarte mic si ultrasonic care totodata are si ca mai mare proporţie putere-greutate din lume si care este echilibrat la semi-înaltime de primul mecanism stabilizator din lume ce foloseşte un actuator liniar.În plus,esenta micromecatronicii a fost asamblata intr-un montaj tehnologic foarte compact pentru minimizarea mărimii si a greutăţii unităţii circuitului de control.

Prin dezvoltarea µFR, Epson a demonstrat posibilitatea de extindere a razei de activitate a roboţilor din spaţiul bi-dimensional (pământul) în spaţiul tridimensional(aer).

Caracteristici ale µFR (Micro Flying Robot)

- mic,usor,motor ultrasonic; tinand cont de cercetarile companiei, actuatorul µFR are cea mai mare proporţie greutate-putere din lume.
- modul wireless cu consum mic; ţinând cont de carcetările companiei, modulul wireless are cel mai mic consum din lume.
- Tehnologie de control la semi-inaltime;

Concluzii:

În lumea zilei de azi,roboţii au un rol important în mai toate domeniile vieţii noastre.

Domeniile de utilizare sunt foarte variate incepand de la mijloace de transport şi terminând cu explorarea spatiala.Odata cu dezvoltarea tehnologiei, cu atât mai mulţi roboţi

sofisticați îşi vor face apariția în aşa fel incât şi viața umană va cunoaşte o imbunătățire semnificativă.

CAPITOLUL II

STUDIUL DINAMICII ROBOTULUI MOBIL CU ROȚI

2.1 Model cinematic robot

În cadrul acestei analize,considerăm robotul ca un corp rigid cu roți,deplasându-se pe un teren plat.Sunt necesari 3 parametri pentru a poziționa corpul robotului în planul respectiv: doi pentru a determina poziția acestuia în raport cu originea sistemului de axe ataşat planului, iar al treilea pentru determinarea orientării acestui corp în raport cu axa verticală, perpendiculară pe plan.Există mişcări suplimentare datorate rotației roților în jurul axelor proprii, respective cuplelor cinematice de orientare.Prin corpul robotului întelegem doar corpul rigid al acestuia, ignorând cuplele cinematice dintre acest corp şi roți.

Un vehicul aflat pe un teren plat are 3 grade de libertate: 2(x,z) ce descriu poziția şi unul referitor la orentarea θ(rotația în jurul axei verticale).În general roboții

mobili nu au control complet independent al celor 3 parametii ce definesc situația lui şi trebuie să efectueze manevre complexe pentru a atinge o anumită situare.

Unele vehicule au roți suplimentare sau puncte de contact, cu rolul de suport dar fără a contribui la orientare sau propulsie.Cunoscute ca roți pasive(ex roți de tip castor), acestea sunt de obicei ignorate în modelarea cinematică a unui robot mobil.

2.1.1 Model cinematic robot acționare diferențială

Pentru a specifica situarea instantanee a robotului în mediul în care se deplasează, stabilim o relație de legătură între sistemul de referință f şi sistemul de axe curent,ataşat corpului robotului

- Planului I se ataşează sistemul de referință fix $X_0 0_0 Y_0$.
- Corpului robotului i se ataşează sistemul de axe $X_R O_R Y_R$, în punctul P
- Corespunzător poziției inițiale a robotului, se consideră notația $X_{R0} O_{R0} Y_{R0}$

$$^0P = Tr_{+r0} {}^0R(\theta_{R0}) {}^{R0}p$$

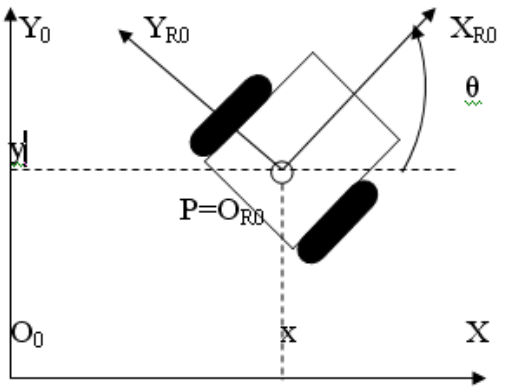


Figura8-Model cinematic robot cu 2 roți

Coordonatele punctului P, determinate în raport cu originea O_0 a sistemului de referință scrise sub formă matriceală sunt:

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \cos\theta_{R0} & -\sin\theta_{R0} & 0 \\ \sin\theta_{R0} & \cos\theta_{R0} & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{R0} \\ y_{R0} \\ z_{R0} \end{bmatrix} \quad (3)$$

- x_0, y_0, θ_0 sunt parametrii ce exprimă poziția şi orientarea corpului robotului în raport cu sistemul de referință;

- $x_{R0}, y_{R0}, \theta_{R0}$ sunt parametrii ce exprimă poziția şi orientarea corpului robotului în raport cu sistemul de axe ataşat acestuia, sistemul corespunzător poziției inițiale

- x, y, θ sunt parametrii ce exprimă poziția şi orientarea poziției inițiale a corpului robotului(poziției inițiale a sistemului de axe ataşat corpului) în raport cu sistemul de axe de referință.

Robotul care face obiectul lucrării este un robot cu acționare diferențială.Acest tip de acționare este una dintre cele mai simple soluții de roboți mobili cu roți. Robotul are 2 roți convenționale fixe,ataşate pe şasiu.Utilizarea de viteze unghiulare diferite pentru cele 2 roți conduce la schimbarea direcției de mers.

$$\begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_{R0} & -\sin\theta_{R0} & 0 & x \\ \sin\theta_{R0} & \cos\theta_{R0} & 0 & y \\ 0 & 0 & 1 & \theta \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{R0} \\ y_{R0} \\ \theta_{R0} \\ 1 \end{bmatrix} \quad (4)$$

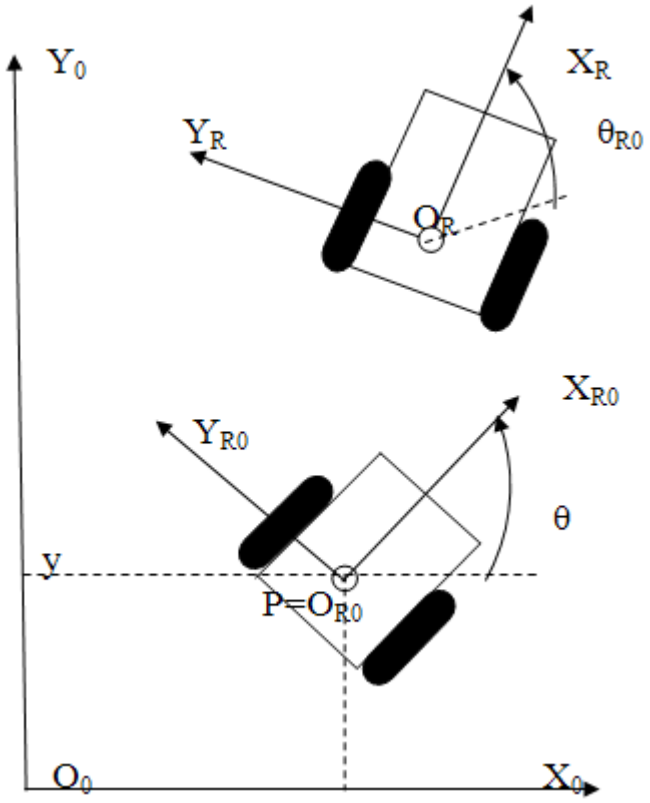


Figura 9- Deplasarea robotului.

Cunoscând vitezele/pozițiile roților se cere viteza/poziția robotului.

Se cunosc : $v1=\omega_1 \times r$; $v2=\omega_2 \times r$

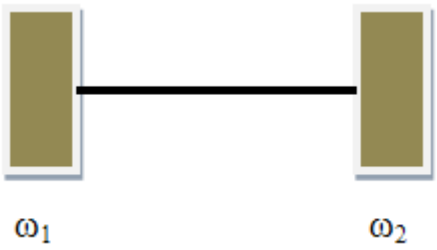


Figura10-Reprezentarea vitezelor unghiulare a 2 roți.

unde r=raza roților şi ω_1, ω_2= vitezele lor unghiulare

Pentru efectuarea analizei, se va proceda în felul următor:

1. Se specifică mărimile necesare efectuării calculului(dimensiuni ale robotului, unghiul de orientare al robotului,θ)

2. Se determină punctul în jurul carua se roteşte robotul(centrul instantaneu de rotație), respectiv raza traiectoriei descrise de robot.

3. Se detrmină viteza unghiulară a robotului.

4. Se integrează relația obținută, pentru a obține poziția robotului.

2.1.2 Determinarea C.I.R.

1.Determinarea mărimilor necesare efectuării calculului(dimensiuni ale robotului; unghiul de orientare al robotului).

2 Se ataşează robotului un sistem de axe şi se măsoară unghiul de rotație al sistemului curent în raport cu sistemul inițial de referință.

Centrul instantaneu de rotație este la intersecția axelor roților pentru cele 2 poziții.

Punctul de contact al fiecărei roți trebuie să se rotească cu aceaşi viteză unghiulară Ω în jurul centrului instantaneu de rotație.

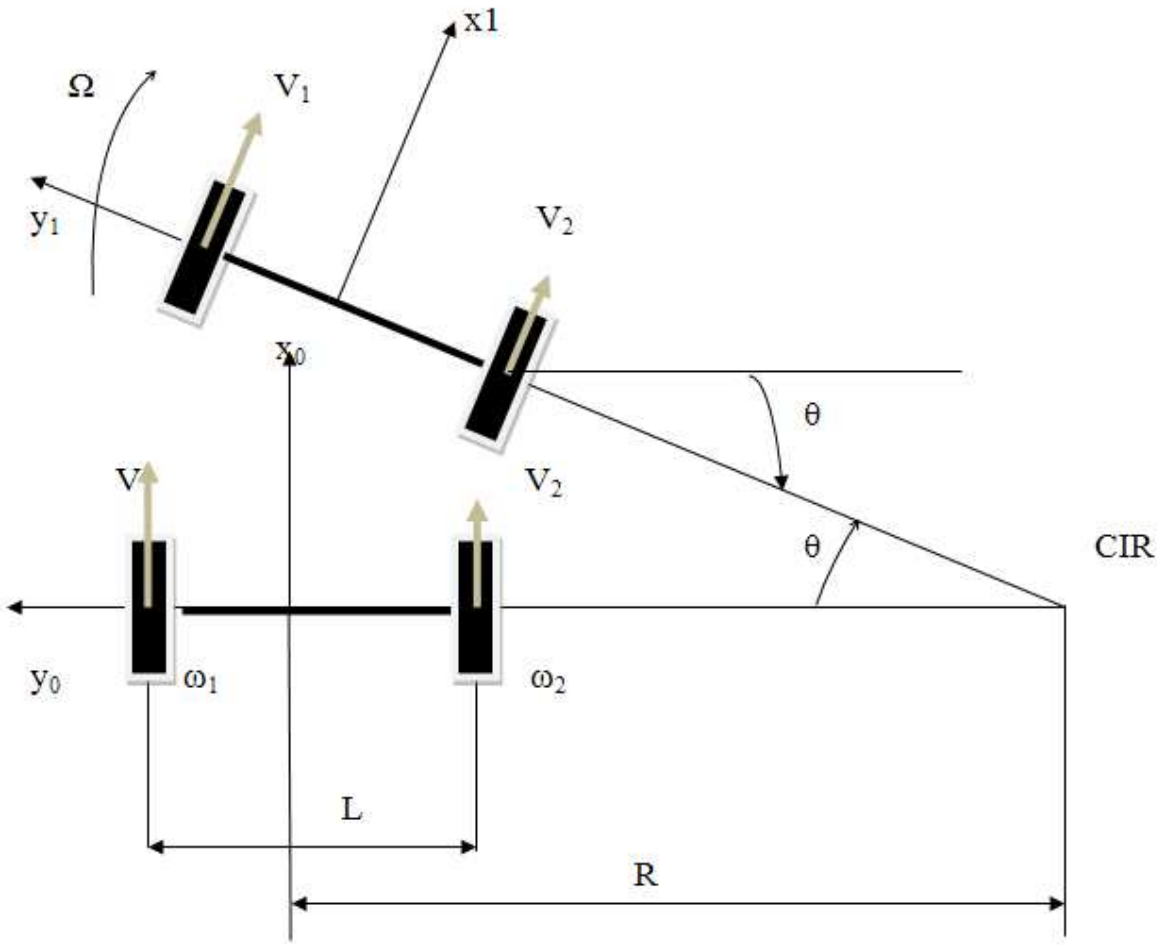


Figura 11- Determinarea C.I.R

.

3.Determinarea vitezei unghiulare ,Ω, a robotului

v1=ω1r=Ω(R+L/2)

v2=ω2R=Ω(R-L/2)

Rezultă

Ω=(v1-v2)/L

R=L(v1+v2)/(v1-v2)

• Pentru traiectoria rectilinie

v1=v2 $\Rightarrow$ Ω=0 şi R=∞

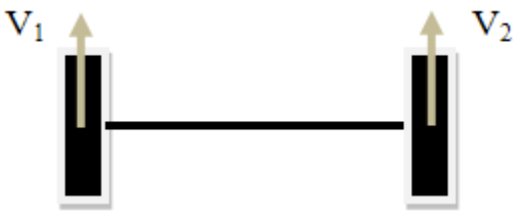


Figura 12-Traiectoria rectilinie.

• Pentru traiectorie circulară cu raza R- relațiile sunt cele prezentate anterior.

Se determină experimental că:

L=10 cm

v1=0.5m/s şi v2=0.3 m/s$\Rightarrow$Ω=2 rad/s^2 şi R=16 cm

• Pentru pivotarea în jurul roții

v1=v2$\Rightarrow$Ω=0

R=∞

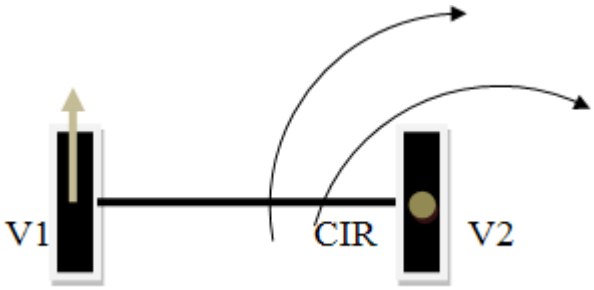


Figura13-Pivotarea în jurul roții 2.

- Pentru pivotare(rotație în jurul axei verticale centrale)

v2=-v1=>Ω=2v1/L=-2v2/L

R=0

v1=ω1r=Ω(R+L/2)

v2=ω2r=Ω(R-L/2)

Rezultă

Ω=(v1-v2)/L

R=L(v1+v2)/(v1-v2)

Viteza liniară a robotului este:

v=Ω×R=(v1+v2)/2=0.4 m/s

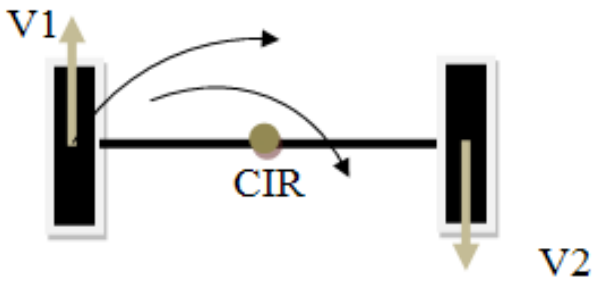


Figura14-Pivotarea în jurul axei verticale.

Integrarea ultimei ecuații pentru obținerea poziției robotului

$dx=vx(t)dt;$

$dy=vy(t)dt;$

$d\theta=\Omega(t)dt;$

$vx(t)=v(t)\cos(\theta(t));$

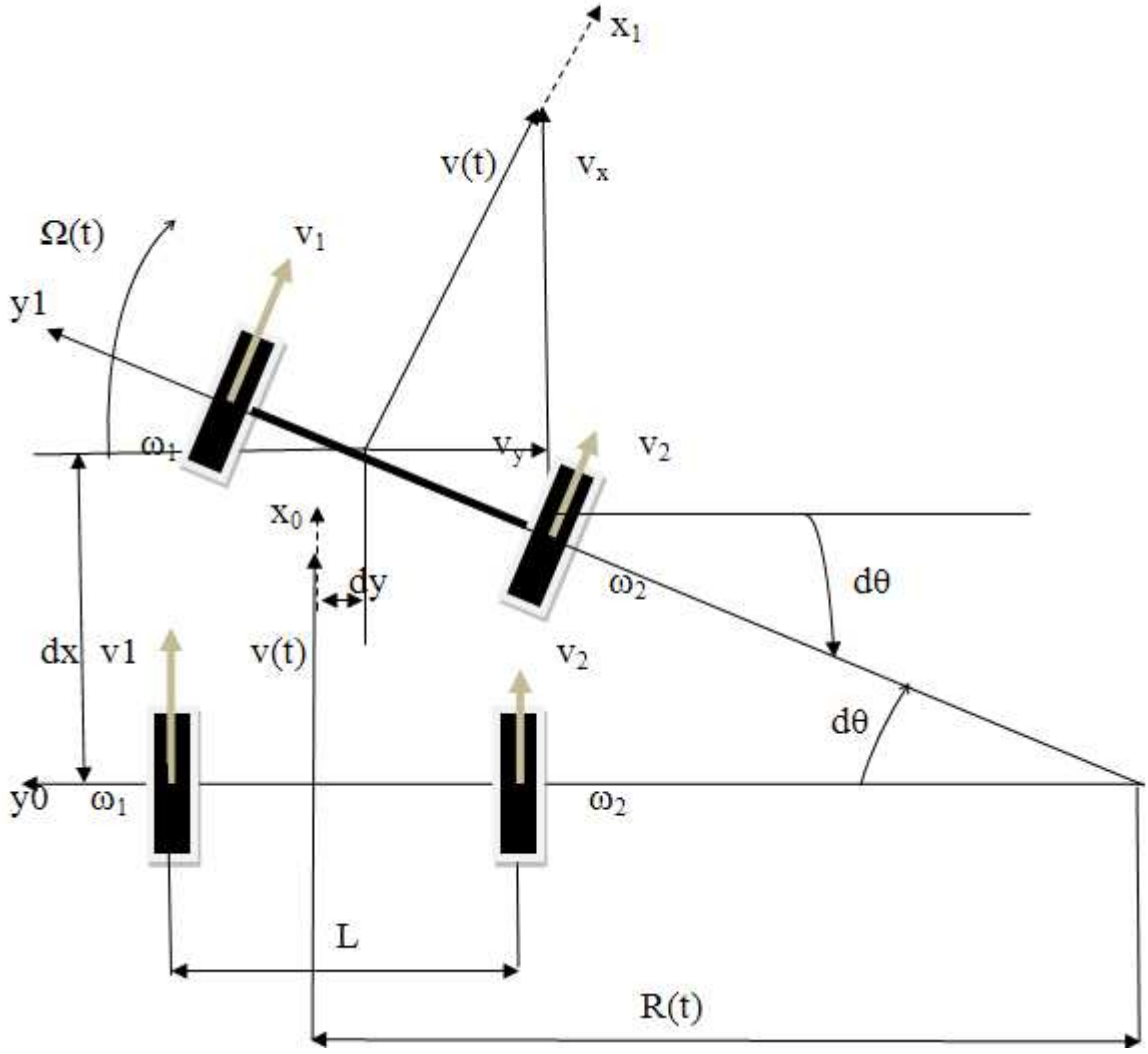


Figura 15-Deplasarea robotului.

vy(t)=v(t)sin(θ(t));

Prin integrare se obține

x(t)=$\int v(t)\cos(\theta(t))$dt+x0;

y(t)= $\int v(t)\sin(\theta(t))$dt+y0;

θ(t)= $\int \Omega(t)\cos(\theta(t))$dt+x0;

Ω=(v1-v2)/L

R=L(v1+v2)/(v1-v2)

v=ΩR=(v1+v2)/2

Ecuațiile cinematicii directe se scriu sub formă matriceală în raport cu sistemul de referință:

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ \Omega(t) \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r/2 & r/2 \\ 0 & 0 \\ r/L & r/L \end{bmatrix} \times \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} v_x(t) \\ v_y(t) \\ \Omega(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} v(t) \\ \Omega(t) \end{bmatrix} \quad (6)$$

Contribuția roții 1 la viteza robotului este:

$v_{x1}=(\omega_1 \times r)/2=0.25$ m/s

$v_{y1}=0$

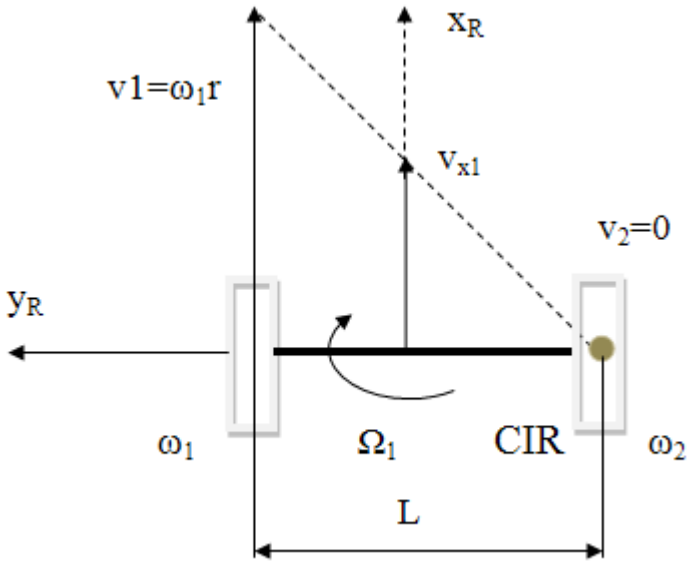


Figura 16- Contribuția roții 1 la deplasarea robotului.

$\Omega_1=(\omega_1 \times r)/L=5$ rad/s

Contribuția roții 2 la viteza robotului

$v_{x2}=(\omega_2 \times r)/2=0.15$ m/s

$v_{y2}=0$

$\Omega_2=-(\omega_2 \times r)/L=-1$ rad/s

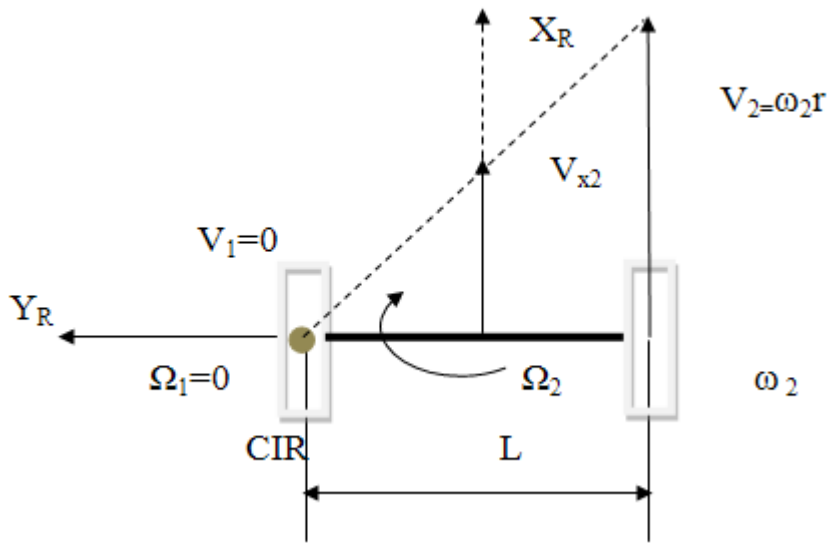


Figura 17- Contribuția roții 2 la deplasarea robotului.

Ecuațiile cinematicii directe scrise în raport cu sistemul de axe ataşat robotului sunt:

$v_{x(}(t)=v_{x1}+v_{x2}=0.4$ m/s

$v_{y(}(t)=v_{y1}+v_{y2}$

$\Omega(t)=\Omega_1+\Omega_2$

Se face ipoteza că roțile nu pot aluneca lateral=> componenta după axa y este 0.Dacă robotul descrie o traiectorie circulară cu raza R, lungimea arcului descris de acesta este:

$A_R=(A_1-A_2)/2$

unde

$A_1=(R+L/2)\times\theta$

$A_2=(R-L/2)\times\theta$

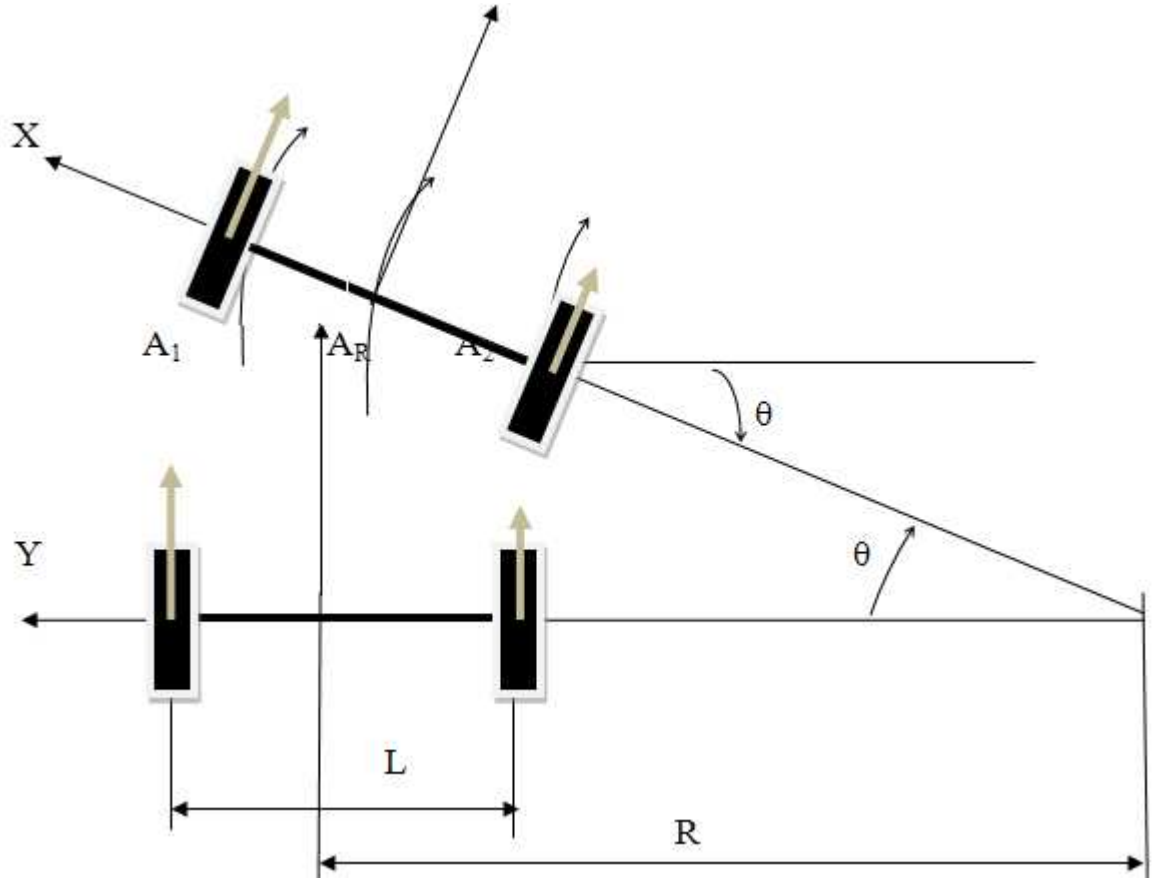


Figura 18- Traiectoria circulară a robotului.

Lungimea cercului descris de punctul de contact al roții 1 este dată de relația

$C_1=2\times\pi\times(R+L/2)$

$$\frac{A_1}{C_1}=\frac{\theta}{2\times\pi}$$

$\theta=A_1/(R+L/2)$

Similar pentru roata 2

$C_2=2\times\pi\times(R+L/2)$

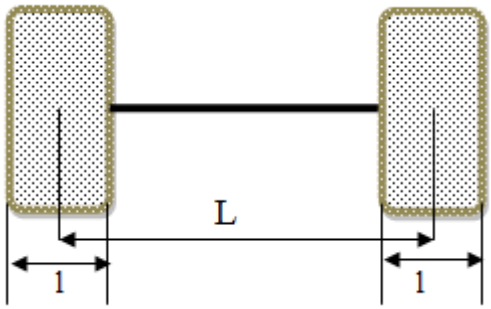


Figura 19-Reprezentare a 2 roți.

$$\frac{A_2}{C_2} = \frac{\theta}{2 \times \pi}$$

De unde rezultă:

$R - (L/2) = A_2/\theta$

Înlocuind această relație în

$\theta = A_1/(R+L/2)$

Prin înlocuire se obține:

$\theta = (A_1 - A_2)/L$

De aici se deduce următoarele concluzii:

- orientarea θ a robotului este complet independentă de raza traiectoriei descrise de acesta
- Dimensiunea L este o sursă de erori din cauza uzurii şi complianței pneurilor

Lungimea traiectoriei descrise de punctul de contact al roții poate fi calculată cu relația:

$A_1=(2\cdot\pi\cdot N_1\cdot R_{ef\,1})/T_1$

N_1=impulsurile numărate de traductorul de poziție al roții 1

T_1=numărul de impulsuri pe o rotație a roții 1

R_{ef1}=raza efectivă a roții 1

Similar pentru roata 2

$A_2=(2\cdot\pi\cdot N_2\cdot R_{ef\,2})/T_2$

Pentru a se deplasa după o traiectorie rectilinie ,robotul va păstra $N_1=N_2$, dar dacă cele 2 pneuri au presiuni diferite ,razele lor for fii diferite.

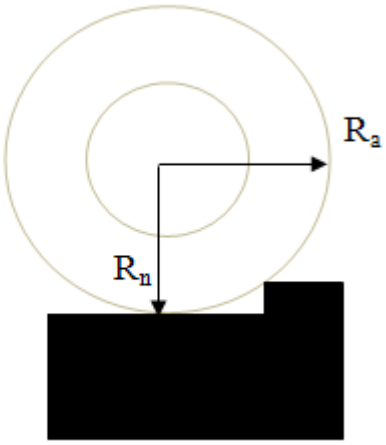


Figura 20- Deplasarea unei roți peste un obstacol.

Considerăm o roată teoretică, rigidă(necompliantă),având raza R şi presupunem

că aceasta trebuie să treacă peste un obstacol cu înălțime h.

Roata se va deplasa pe suprafața C până când întâlneşte obstacolul $N_1=N_2$

Distanța percepută de traductor ca fiind parcursă de roată este Am

Distanța reală parcursă este Ah. Eroarea de poziționare unghiulară a robotului va fii:

$\Delta\theta = 2^{\times}(A_m-A_h)$

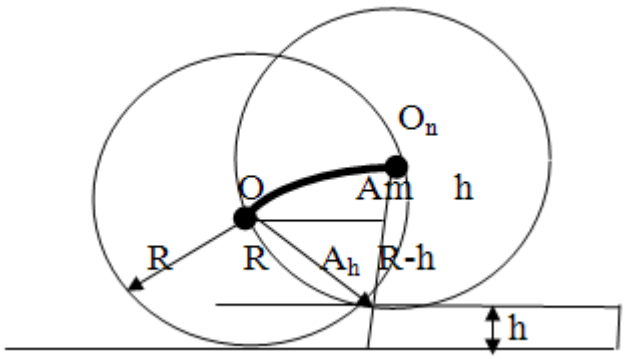


Figura 21- Deplasarea unei roți rigide peste un obstacol

Atunci când roata traversează o groapă aflată în terenul în care se deplasează,obstacolele şi gropile în teren produc erori privind determinarea corectă a poziției robotului.Erorile sunt cauzate şi de alunecarea roților ca urmare a existenței unor pete de unsoare pe suprafața respectivă, de impactul cu obstacole sau de accelerarea şi decelerarea excesivă sau de impactul cu obstacole.

CAPITOLUL III

PROIECTAREA ELEMENTELOR CONSTRUCTIVE

Tema tehnică a lucrării

Să se proiecteze şi să se realizeze platforma mobilă a unui minirobot care să îndeplinească următoarele condiţii tehnice

Parte mecanică

- Masa totală:450 grame
- Viteză maximă:0.5 m/s
- Roți motoare: 2 cu acționare independentă
- 2 sfere montate pe şasiu cu rol de menținere echilibru
- Garda la sol(distanța dintre cel mai jos element mecanic şi sol):minim 5 mm
- Cadrul de susținere(şasiu)

Parte electrică

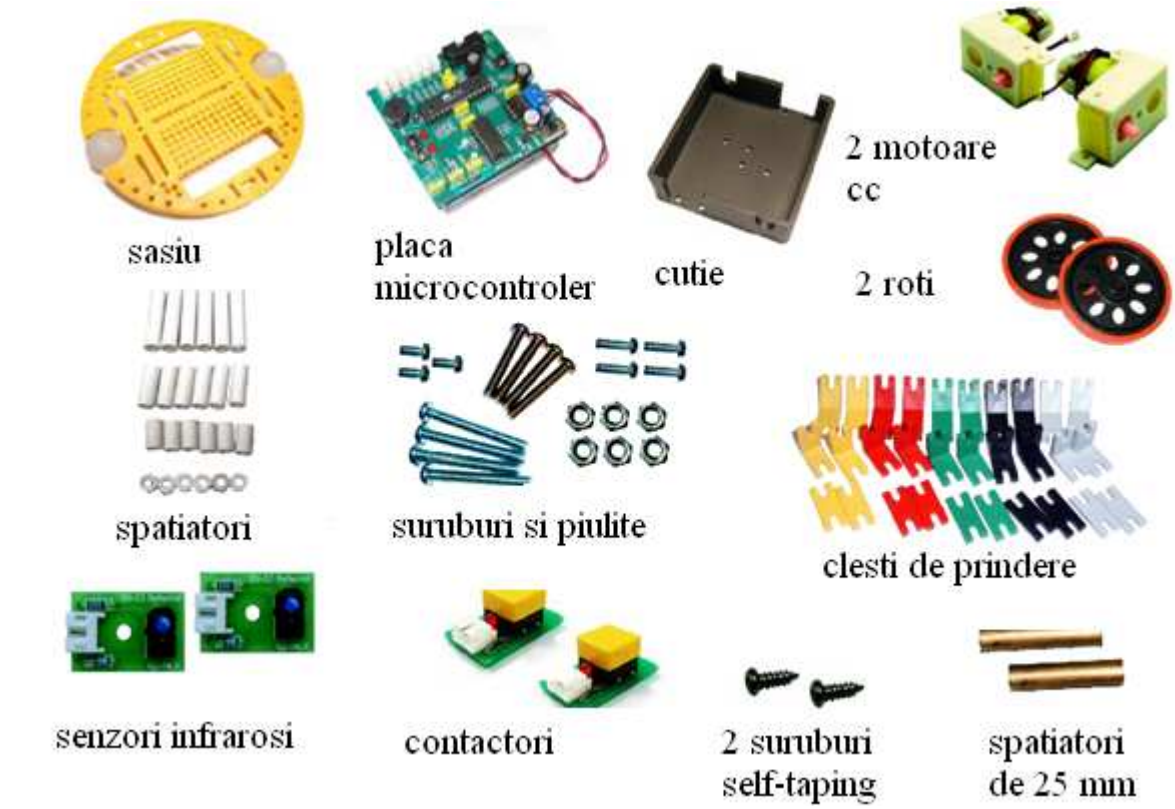
- Tip motor de acționare curent continuu(cu excitație paralel)
- Baterii: 4×1.5V(curent continuu)
- 1 senzor infraroşu Sharp
- 2 LED-uri
- 5 porturi analogice pentru conectarea senzorilor şi un port de comunicație serial
- 2 senzori de contact
- 2 reflectori infraroşii
- 2 leduri semnalizare
- 1 telecomandă cu emisie infraroşu
- 1 receiver infraroşu,1 afişor cu cristale lichide,1 senzor de temperatură

Parte informatică

- Comandă motoare: programare microcontroler
- Dezvoltarea programelor respective

PROIECTAREA ŞI REALIZAREA MINIROBOTULUI.

3.1 Elementele componente ale minirobotului



senzor infraroşu sharp senzor de temperatură afişor cu cristale lichide

Figura8-Elementele componente ale robotului

Componenta de bază a robotului o constituie placa microcontrolerului pe care este prezent microcontrolerul Atmel ATMEGA8-16PI împreună cu diferite circuite auxiliare de interfață cu diferite medii printre care enumerăm circuitul integrat L293D cu rol de punte H ce îndeplineşte rolul de amplificare al semnalului de la pinii microcontrolerului şi acționare de putere a motoarelor de curent continuu şi circuitul integrat TL499 care are rol de stabilizator de tensiune. Pe lângă aceasta mai avem două motoare de curent continuu doi senzori de contact un senzor infraroşu Sharp şi alte componente mecanice care utilizează ca mediu de programare limbajul C.

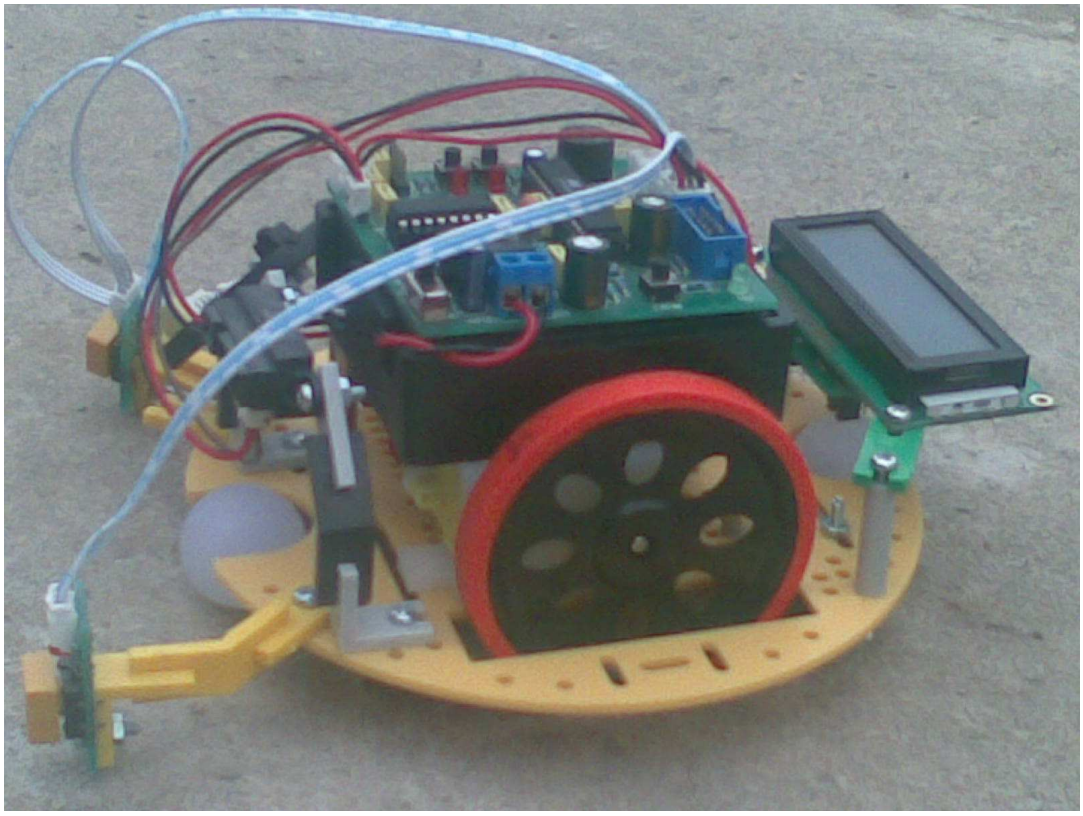


Figura9-Vedere laterală minirobot

Robotul este conceput pe două roți motrice din plastic cu membrană de cauciuc iar pe şasiu se mai găseşte 2 sfere cu rol de echilbrare .Este alimentat cu 4 baterii de 1.5 volți tip AA. Pentru a înnobila partea electronică robotul a fost dotat cu senzori infraroşii pentru urmărire o linie neagră, 2 contactori pentru atins obstacole si evitarea lor ,un senzor de temperatura pentru determinarea temperaturii obiectelor cu care vine in contact, un afişor cU cristale lichide pentru scriere diferite mesaje si un senzor infrarosu Sharp care ocoleşte obstacole si care deasemenea poate determina distanța până la un anumit obiect. Robotul poate fii manipulat cu ajutorul unei telecomenzi IR prin intermediul unui receiver IR.

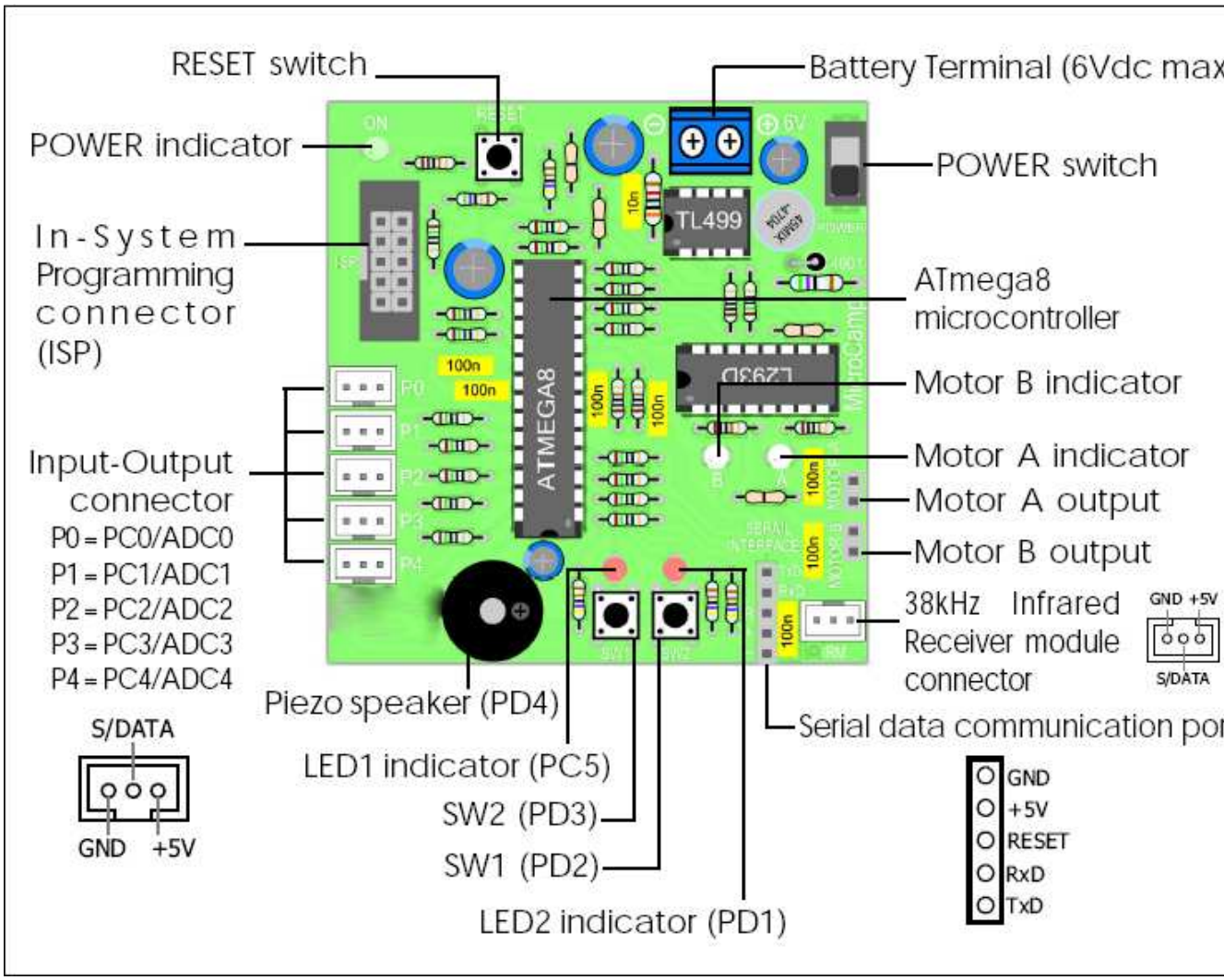


Figura 9- Planul general al plăcii microcontrolerului.

3.2 Placa miniribotului.

- Principalul microcontrolor este de tip AVR pe 8 biți de la Atmel; ATmega8.

Conține multe trăsături caractersitice microcontrolorului modern cum ar fii: Convertizorul Analog Digital(ADC) pe 10 biti, programul Flash de memorie 8Kb cu 10000 cilcluri ştergere-scriere, Data EEPROM de 512 bytes si RAM 512 bytes.

- Frecvența internă a ceasului de 16MHz.

- 5 Porturi Intrare/Ieşire pe 3 pini cu 5 canale programabile.Acestea se poate intrebuin□a ca port digital de intrare, port digital de ieşire şi port analog de intrare.Alimentarea se face la o tensiune normal de 5V

Conține un port pentru conectarea Receptorului Infraroşu cu frecvența de 38KHz.Acesta este desemnat să participe cu Semnalul Serial de primire(RxD) către dispozitivul exterior de comunicare.

- Piezodifuzor pentru semnale sonore.

- 2 comutatoare pentru apasare(switch-uri).

- Comutator de restartare.

- 2 indicatoare LED active atunci cand logic este “1”.

- 2 canale pentru motoare in curent continuu alimentate la o tensiune de 4.5 V şi intensitate 600 mA cu indicatoare LED.

- Alimentarea se face cu 4 baterii tip AA.

- Circuit integrat pentru a menține o tensiune constanta de 5 volți atunci când motoarele funcționează.

- Se mai găsesc de asemenea grafuri(rezistențe care au rol de a limita trecerea curentului şi condensatori care înmagazinează curent şi apoi îl descarcă)

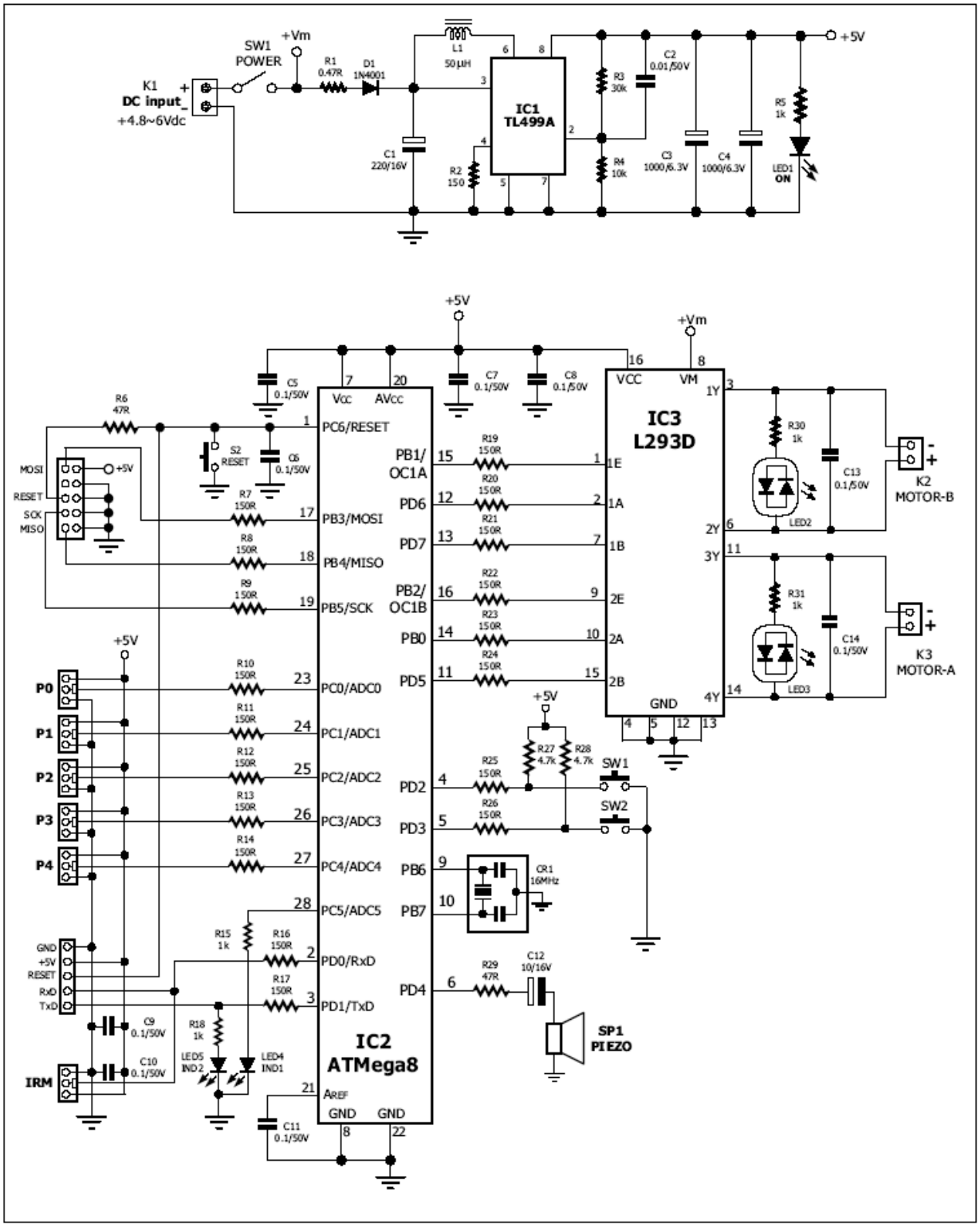


Figura 10- Schema electrică a plăcii de bază a minirobotului

3.2.1 Descrierea funcționării plăcii de baza a minirobotului

Piesa principală o constituie microcontrolorul ATmega8. Functionează la o frecvență de 16MHz din crytal care este conectat la pinii PB6 şi PB7.

Toate porturile pot fii programabile pentru intrare/ieşire analogă sau digitală.Semnalele analoge din porturile PC0 -PC4 trec prin Convertorul Analog Digital . Rezoluția conversiei este de 10 biți.

Porturile PB3,PB4 şi PB5 sunt porturi Programare In Sistem.

Pinul PC6/Reset este conectat cu comutatorul RESET pentru restartarea operațiilor microcontrolorului de catre utilizator.

Pinul PD0/RxD este receptor serial.Participă cu conectorul IRM(Senzorul Infraroşu) şi portul de comunicare pe 5 pini Serial.

Pinul PD1/TxD transmite serial.Participă la activarea LED5(etichetat IND2) şi a pinului TxD

Pentru LED4(IND1) este conectat direct la PC5 cu rezistor de limitare a curentului

Placa de baza a minirobotului conține şi 2 comutatoare conectate la PD2 şi PD3 activând un rezistor de 4.7kΩ care seteaza nivelul logic"High" într-o operație normală şi schimbă la nivelul logic "Low" sau "0" cand este apăsat comutatorul.

Pinul PD4 este conectat la un Piezo difuzor de capacitate 10μF.

De asemenea pe placă se mai găseşte dispozitivul DC circuit motor current continu.Dispzitivul IC este L2932D H-Punte. Un dispozitiv circuit motor curent continu necesită 3 pini semnal pentru funcționare.A(pinii PD6 şi PD0) şi B(pinii PD7 şi PD5) sunt intrări pentru aplicarea semnalului şi selectarea sensului de rotație a motorului. E(pinii PB1 şi PB2) sunt folosiți pentru activarea şi dezactivarea funcționării circuitului dispozitiv.Prin aplicarea semnalului PWM la aceşti pini se

poate controla şi viteza motorului.Daca semnalul PWM(Pulse Width Modulation) are lărgime înseamna ca nivelul"high" a fost trimis către ieşirea motorului.

Circuitul de alimentare este de tip întrerupător.TL499A are funcția de stabilizator pentru alimentarea circuitului microcontrolerului la o tensiune de +5V.

La ieşirea L2932D, un led bi-color este conectat pentru a indica polul voltajului.Culoarea verde înseamna înainte,culoarea roşu înseamna înapoi.

3.2.2 Microcontrolerele AVR

Față de microprocesor, microcontrolerul are integrat pe acelaşi chip oscilatorul,

memoria program, o memorie RAM, numărătoare, interfețe de comunicație şi porturi de

intrare ieşire. De asemenea în structura unor microcontrolere mai sunt integrate blocuri

analogice cum ar fi : convertoare analog numerice de 8, 10 sau 12 biți, comparatoare

analogice, modulatoare PWM, etc.

Tipurile mai noi de microcontrolere au integrata în structura şi o memorie EEPROM

necesară pentru memorarea setărilor sau parametrilor aplicației la căderea tensiunii de

alimentare.

La fel ca şi microprocesoarele, unele tipuri de microcontrolere pot adresa o memorie

program şi o memorie RAM, externe. Datorita preţului redus de cost şi simplităţii de utilizare,

au cunoscut o dezvoltare spectaculoasă microcontrolere cu memorie flash (memorie

EEPROM). Memoria poate fi ştearsă şi înscrisă de mai mult de 1000 de ori, număr suficient

de mare pentru punerea la punct a oricărei aplicaţii. preţul unui microcontroler cu memorie flash incorporată este de câţiva dolari, în funcţie de caracteristici şi de gradul de complexitate.

Acelaşi microcontroler înscris cu un program de aplicaţie, se poate vinde cu un preţ de

la 2 până la 100 de ori mai mare. pe piaţa de componente electronice există o diversitate

foarte mare de familii de microcontrolere. Firmele producătoare de microcontrolere au în

producţie zeci de variante de microcontrolere cu o gamă variată de caracteristici.

Una din cele mai cunoscute familii de microcontrolere este seria ’51 a firmei Intel.

Această familie de microcontrolere este produsă de mai multe firme (Intel, Siemens, Philips,

Atmel, MHS etc.) care, pornind de la o structura de baza, au creat o varietate foarte mare de tipuri de microcontrolere.

Primul tip de microcontroler din seria ‘51 apărut pe piaţă a fost microcontrolerul

I8048. Acest microcontroler este folosit pentru comunicația cu tastatura în calculatoarele PC.

O altă clasă de familii de microcontrolere ce s-a impus pe piață în ultimul timp este

cea a microcontrolerelor cu structura RISC (reduced instruction set computer). Spre deosebire de microcontrolerele cu un set complex de instrucț iuni de tip CISC (complex instruction set computer), microcontrolerele cu arhitectura RISC au un set redus de instrucțiuni care, însă, în cele mai multe cazuri, se execută într-un singur ciclu de ceas.

Pentru microcontrolerele RISC memoria program este organizată pe 12, 14 sau 16 biți.

Codul de program este citit intr-un singur ciclu de ceas iar procesarea informațiilor se face paralel. Marea majoritate a microcontrolerelor RISC au posibilitatea de înscriere a

programului de aplicație direct în circuit. Înscrierea în circuit ISP (în system programming) se face pe interfața serie SPI.

Dintre familiile de microcontrolere cu arhitectura RISC amintim seria demicrocontrolere PIC micro ale firmei Microchip (PIC 12C508, PIC 12C509, PIC 12CE519,

PIC 12C672, PIC 12CE674 etc.) şi seria AVR a firmei Atmel.(AT90S1200, AT90S2313,

AT90S2343, AT90S4433, AT90S8515, AT90S8535, AT90C8534, ATTINY22,

ATMEGA103 etc.).

Microcontrolere AVR au o structura RISC nouă, realizată în anii 90. Arhitectura

acestor microcontrolere a fost elaborată în colaborare cu experţi în limbajul C, rezultând un produs ce are un set eficient de instrucţ iuni, care permite scrierea unor programe de aplicaţie performante, cu un cod optim.

Majoritatea instrucţiunilor implementate în microcontrolerele AVR se execută într-o

singură perioadă de ceas. La aceeaşi frecvenţă de lucru, microcontrolerele AVR au o viteză de lucru de peste 10 ori mai mare faţă de microcontrolerele cu structura clasică din seria ’51, la care o instrucţiune se execută în 12 perioade de ceas.

Un alt avantaj al microcontrolerelor din seria AVR este acela că au implementat un set

de 32 de regiştrii de lucru cu funcţie de acumulator, faţă de unul singur în cazul celorlalte

microcontrolere. Toate microcontrolerele din seria AVR au implementat un numărător

watchdog şi majoritatea au funcţia ISP pentru programare în circuit.

3.2.3 Microcontrolerul Atmel ATmega8

ATmega8 este un microcontroller CMOS low-power pe 8 biti, bazat pe arhitectura AVR RISC. Prin executarea unor instructiuni puternice într-un singur moment de tact, ATmega8 ajunge la performanţa de 1 MIPS pe MHz permiţându-i desingnerului de sistem să optimizeze consumul de putere contra vitezei de procesare. ATmega8 are urmatoarele caracteristici :

- microcontroller low-power pe 8 biţi, bazat pe arhitectura AVR RISC.
- 8K biţi de Flash Programmable In-System cu capacitate citire in timpul scrierii.Este folosită pentru memorarea unui program scris. Pentru că memoria ce este

făcută în tehnologia FLASH poate fi programată şi ştearsă mai mult decât odată, aceasta face microcontrolerul potrivit pentru dezvoltarea de componentă

- 512 biti de EEPROM-memorie de date ce trebuie sa fie salvate când nu mai este alimentare.Este în mod uzual folosită pentru înmagazinarea de date importante ce nu trebuie pierdute dacă sursa de alimentare se întrerupe dintr-o dată.De exemplu o astfel de dată este o temperatură prestabilită în regulatoarele de temperatură.Dacă în timpul întreruperii alimentării această dată se pierde, v-a trebui sa facem ajustarea încă o dată la revenirea alimentării.Astfel componenta pierde în privința auto-menținerii;

- 1K bit de SRAM-memorie de date folosită de un program în timpul executării sale.În SRAM sunt înmagazinate toate rezultatele intermediare sau datele temporare ce nu sunt cruciale la întreruperea sursei de alimentare.

- 32 de registri cu scop general;

- 23 de linii generale I/O grupate în 3 porturi. PORTUL B , PORTUL C şi portul D care sunt conexiuni fizice între microcontroler şi lumea de afară.

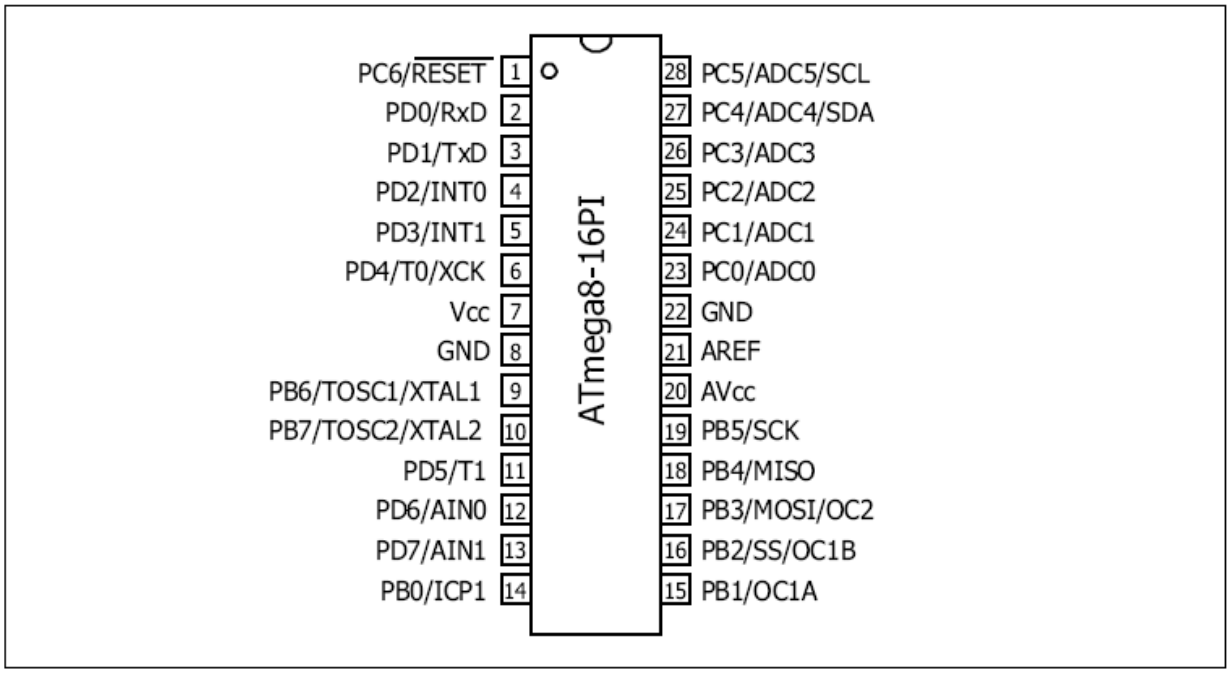


Figura 11-Microcontrolerul ATmega8

- 2 Cronometre/Numaratoare pe 8 biți flexibile cu moduri de comparare, intreruperi interne si externe,
- Cronometru/numărător pe 16 biți cu moduri comparare ,moduri capatare întreruperi interne şi externe;
- Cronometru timp real cu oscilator separat;
- 3 Canale PWM;
- 6 Canale ADC cu acuratețe de 10 biți;
- USART serial-programabil,
- Interfata Seriala, AdC cu 6 canale (8 canale in TQFP si pachete MLF) cu acuratețe de 10 biti,
- Cronometru de supraveghere cu oscilator intern, un port serial SPI si 5 moduri software de economisire de putere.

Modul Idle opreste unitatea centrala in timp ce permite SRAM-ului, cronometrelor/numaratoarelor, porturilor SPI şi sitemului de întrerupere să funcționeze. Modul Power-down salvează conținutul registrelor dar ingheață oscilatorul, făcând incapabile orice alte functii ale cip-ului pâna la urmatoarea întrerupere sau resetare de hard. În modul Power-save numărătorul asincron continuă să functioneze, fapt ce permite utilizatorului sa mentină o bază în timp ce restul dispozitivului este in repaus. Modul ADC Noise Reduction opreşte Unitatea centrală si toate modulele de I/O cu exceptia numărătorului asincron si ADC-ul, pentru a minimaliza zgomotul de comutație din timpul conversiei ADC.În modul Standby functioneaza doar oscilatorul in timp ce restul dispozitivului este in repaus.

Dispozitivul este fabricat folosind tehnologia de memorie volatile cu densitate mare Atmel. Memoria flash poate fi reprogramata folosind o interfata seriala SPI , printr-un program de memorie conventional nonvolatile.Programul boot poate folosii orice interfața pentru a descarca programul de aplicatii din memoria de aplicații flash. Softul din sectiunea de boot flash va continua sa ruleze in timp ce aplicatiile flash sunt updatate furnizând operații reale de citire în timpul scrierii. Combinând o unitate centrala RISC pe 8 biti cu flash self programmable in sistem, pe un cip monolithic , Atmel ATmega8 devine un microcontroler puternic care furnizeaza solutii extreme de flexibile si optime ca preț pentru multe aplicatii.

ATmega8 AVR este dotat cu un set complet de programe si instructiuni ce includ compilatoare C, macro-asamblatoare si simulatoare/corectoare de programe, emulatoare de circuite si kituri de evaluare.

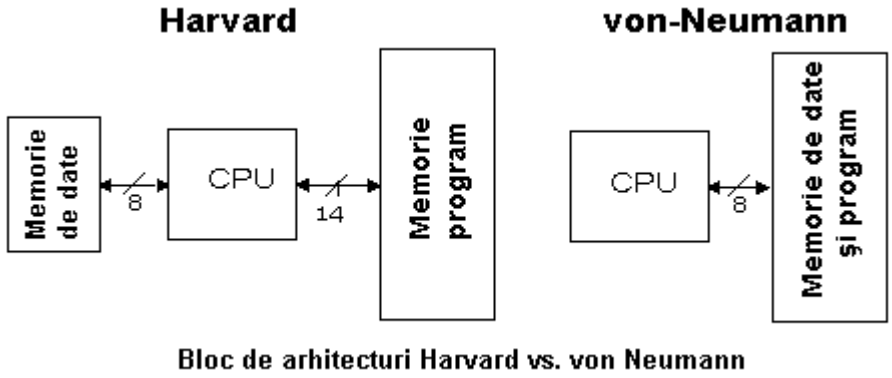


Figura12-Arhitecturile Harvard şi von Newmann

Are o arhitectura RISC. Acest termen este adeseori gasit in literatura despre calculatoare, si are nevoie sa fie explicat aici mai in detaliu. Arhitectura Harvard este un concept mai nou decat von-Neumann. S-a nascut din nevoia de mărire a vitezei microcontrolerului. În arhitectura Harvard, bus-ul de date şi bus-ul de adrese sunt separate. Astfel este posibil un mare debit de date prin unitatea de procesare centrală, si bineînteles, o viteza mai mare de lucru. Separarea programului de memoria de date

face posibil ca mai departe instrucțiunile să nu trebuiască sa fie cuvinte de 8 biti. ATmega8 foloseste 14 biti pentru instructiuni ceea ce permite ca toate instructiunile sa fie instructiuni dintr-un singur cuvant. Este de asemenea tipic pentru arhitectura Harvard sa aiba mai putine instructiuni decat von-Newmann si sa aiba instructiuni executate uzual într-un ciclu.

Microcontrolerele cu arhitectura Harvard sunt de asemenea numite "microcontrolere RISC". RISC inseamna Reduced Instruction Set Computer. Microcontrolerele cu arhitectura von-Newmann sunt numite "microcontrolere CISC". Titlul CISC inseamna Complex Instruction Set Computer.

Pentru ca ATmega8 este un microcontroler RISC, aceasta inseamna ca are un set redus de instructiuni, mai precis 35 de instructiuni (de ex. microcontrolerele INTEL si Motorola au peste 100 de instructiuni). Toate aceste instructiuni sunt executate intr-un ciclu cu exceptia instructiunilor jump si branch. Conform cu ceea ce spune constructorul, ATmega8 ajunge la rezultate de 2:1 in compresia cod si 4:1 in viteza in comparatie cu alte microcontrolere de 8 biti din clasa sa.

Descrierea pinilor.

Portul B este un port de iesire bidirectional pe 8 biți cu rezistoare interne, selectate pentru fiecare bit.

Port C(PC5…PC0) este un bidirectional I/O pe 7 biți cu rezistoare interne selectate pentru fiecare bit. Ieşirea portului C are caracteristici simetrice .

PC6/RESET

Dacă conținutul RSTDISBL este programat, PC6 este folosit ca pin de I/O. Caracteristicile electrice ale portului PC6 difera de cele ale altor pini ai portului C.

Portul D este un port de I/O bidirectional pe 8 biti cu rezistoare interne selectate pentru fiecare bit. Dacă conţinutul RSTDISBL-ului este neprogaramat, PC6 este

folosit ca o resetare a ieşirilor. Un nivel scăzut pe acest pin pentru o lungime minimă de puls va genera o resetare, chiar daca ceasul nu este funcționabil.Pulsuri mai mici nu garantează o resetare

Nume	Număr pin	Tip	Descriere
PB0 ICP1	14	Intrare/Ieşire Intrare	Port Digital PB0 Intrare Captare 1
PB1 OC1A	15	Intrare/Ieşire Ieşire	Port Digital PB1 Ieşire Comparare/PWM 1A
PB2 OC1B SS	16	Intrare/Ieşire Ieşire Intrare	Port Digital PB0 Ieşire Comparare/PWM 1B Intrare secundară pentru SPI şi ISP
PB3 OC2 MOSI	17	Intrare/Ieşire Ieşire Intrare/Ieşire	Port Digital PB3 Ieşire Comparare/PWM 2B Intrare date în mod Slave între busul SPI şi ISP Ieşire date în mod Slave între busul SPI şi ISP
PB4 MISO	18	Intrare/Ieşire Intrare/Ieşire	Port Digital PB4 Intrare date în mod Master între busul SPI şi ISP Ieşire date în mod Slave între busul SPI şi ISP

Nume	Numărul pinului	Tipul	Descriere
PC0 ADC0	23	Intrare/Ieşire Intrare	Port PC0 Intrare analog canal 0
PC1 ADC1	24	Intrare/Ieşire Intrare	Port PC1 Intrare analog canal 1
PC2 ADC0	25	Intrare/Ieşire Intrare	Port PC2 Intrare analog canal 2
PC3 ADC3	26	Intrare/Ieşire Intrare	Port PC3 Intrare analog canal 3
PC4 ADC4 SDA	27	Intrare/Ieşire Intrare Intrare/Ieşire	Port PC4 Intrare analog canal 3 Interfață serială bidirectională
PC5 ADC3	28	Intrare/Ieşire	Port PC5 Intrare analog canal 4

SCL		Intrare Ieşire	Ieşire serială pe 2 fire
PC6 RESET	1	Intrare/Ieşire Intrare	Port PC6 Resetare

.

Nume	Numărul pinului	Tipul	Descriere
PD0 RxD	2	Intrare/Ieşire Intrare	PD0 port digital USARTprimeşte intrare
PD1 TxD	3	Intrare/Ieşire Ieşire	PD1 port digital USARTtransmite ieşire
PD2 INT0	4	Intrare/Ieşire	PD2 port digital Întrerupe extern

		Ieşire	canal 0
PD3 INT1	5	Intrare/Ieşire Ieşire	PD3 port digital Întrerupe extern canal 1
PD4 XCK T0	6	Intrare/Ieşire Intrare/Ieşire Intrare	PD4 port digital Ceas extern USART Intrare externă Timer 0
PD5	11	Intrare/Ieşire Intrare	PD5 port digital Intrare externă Timer 1
PD6 AIN0	12	Intrare/Ieşire Intrare	PD6 port digital Intrare canal 2 comparator analog
PD7 AIN1	13	Intrare/Ieşire Intrare	PD6 port digital Intrare canal 1 comparator analog

Nume	Numărul pinului	Tipul	Descriere

Vcc	7	Intrare	Alimentare tensiune între 4.5 şi 5.5 V
GND	8,22	Intrare	Masa
Avcc	20	Intrare	Alimentare tensiune 5Vpentru CAN
AREF	21	Intrare	Tensiune intrare de referința pentru CAN

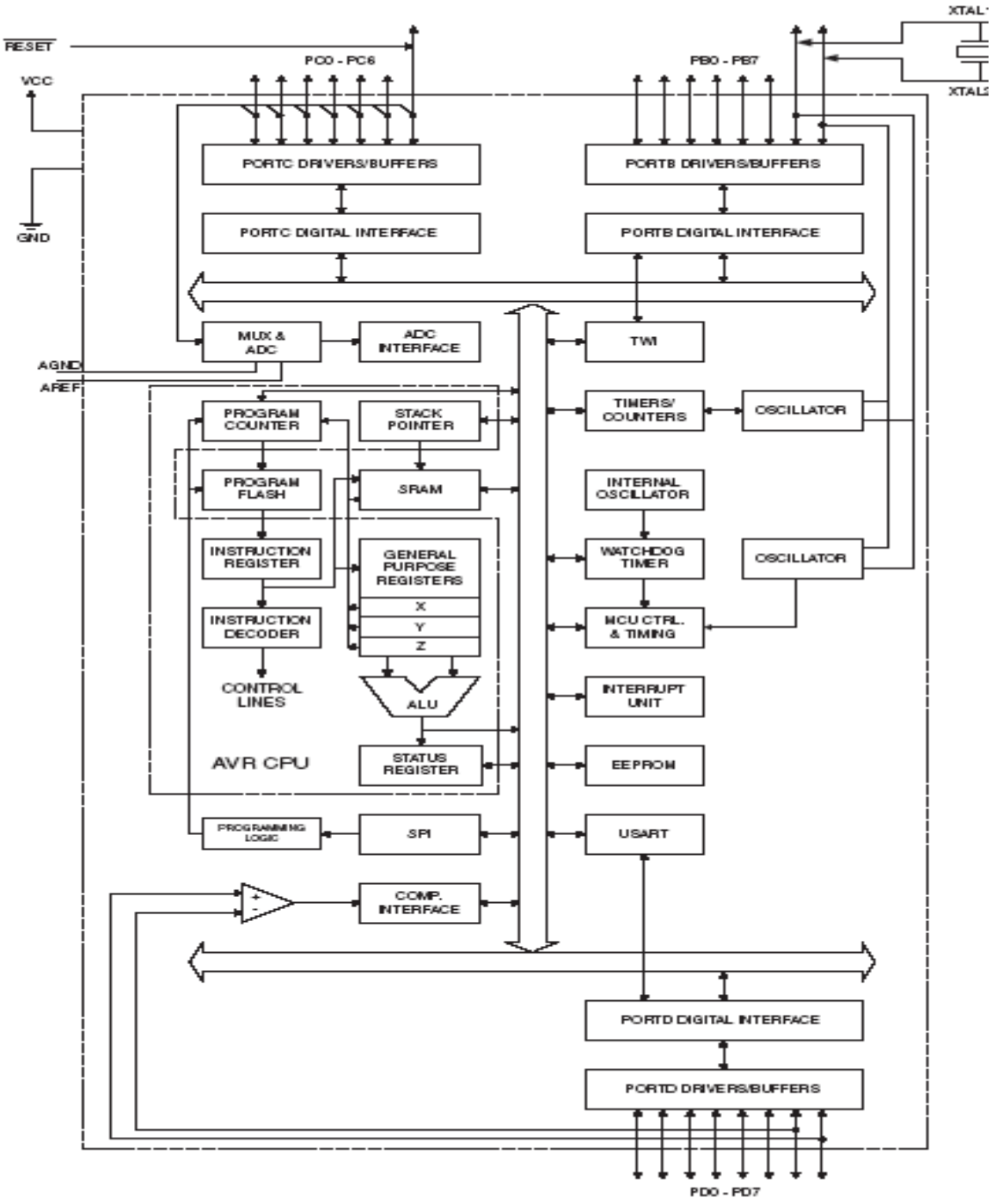


Figura 13-Diagrama bloc a microcontrolerului

Oscilatorul

ATmega8 are încorporat un oscilator intern, la care se conectează un rezonator ceramic modul de conectare este prezentat în figura de mai jos.

Valoarea condensatoarelor ce se conectează la masă se alege în funcție de frecvența

quarțului (uzual 33-100pF). Pentru a fi utilizată la comanda altor blocuri electronice din

sistem ieşirea oscilatorului intern XTAL2 poate fi încărcată cu maxim o sarcină HC. De

asemenea, microcontrolerele pot avea ca semnal de ceas şi un semnal extern aplicat la intrarea

XTAL1 (Fig. 11).

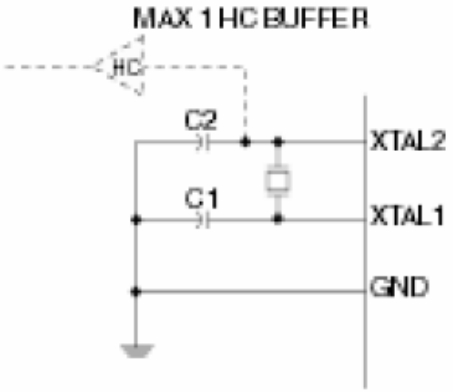


Fig. 14- oscilatorul

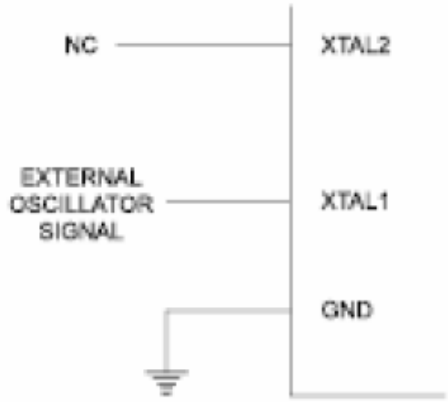


Fig 15 – folosire semnal extern de ceas

Numărătoarele

Microcontrolerul ATmega8 are implementat în structura interna două numărătoare

de 8 şi respectiv şi unul de16 biți. Numărătoarele 0 şi 1 de 8 biți generează întrerupere la depăşire, iar numărătorul 1 de 16 biț i generează întrerupere la depăşire,

pentru comparare şi captură. Ambele numărătoare au câte un circuit de prescalare (divizare şi multiplexare) prin

care se alege sursa semnalului de intrare în numărător. Circuitul de prescalare realizează

divizarea ceasului sistemului cu 8, 64, 256 si1024. pentru fiecare numărător se pot alege

următoarele surse ca semnal de intrare: intrarea conectata la zero, semnalul de ceas, cele patru

ieşiri ale blocului de prescalare sau un semnal extern (front crescător sau descrescător).

Schema circuitului de prescalare a numărătoarelor este prezentata în figura 16.

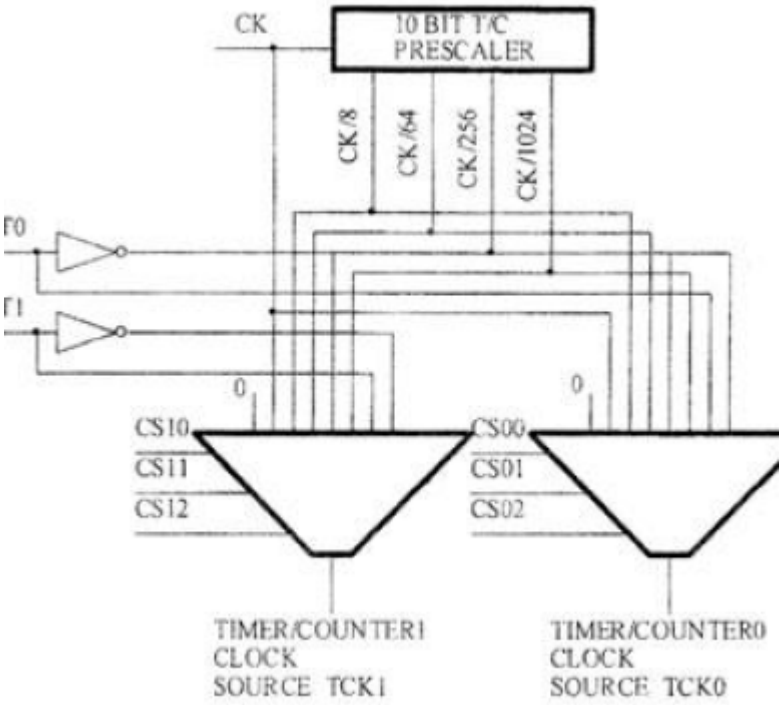


Figura 16-Circuit de divizare pentru numărătoarele 0 şi 1

Numărătorul watchdog

Numărătorul watchdog este pilotat de un oscilator RC separat din structura interna amicrocontrolerului. Frecventa de oscilație este de 1 MHz la tensiunea de alimentare de 5V şi de 350 KHz la o tensiune de alimentare de 3 V. Frecventa semnalului de intrare în numărătorul watchdog se obține din oscilatorul RC prin divizare. Funcționarea numărătorului watchdog este controlata prin registrul de control WDTCR (watchdog timer control register)

Interfața USART

Microcontrolerul ATmega8 are implementată în structura internă o interfață serială asincronă de tip USART. Transmisia pentru această interfață se face cu următoarele caracteristici:

- transmisie pe 8 sau 9 biți

- filtrarea zgomotului, detector de suprarecepț ie

- bit de start fals sau lipsa bit de stop

- generator pentru viteza de transmisie

- 3 surse de întreruperi

Transmisia se realizează prin scrierea în registrul de date UDR a datei ce trebuie transmisă. În funcție de setările interfeței seriale USART datele se vor transmite cu anumite caracteristici. Pentru transmisia de date exista 2 surse de întreruperi şi anume întreruperea pentru registrul de transmisie gol şi întreruperea pentru transmisie de date terminată. Pentru recepție avem o singură sursă de întrerupere şi anume cea de recepție caracter. Recepția se face cu aceleaşi caracteristici ca şi transmisia.

Datele ce se transmit pe interfață au următoarea structură:

- bit de START

- LSB

- …

- MSB

- (bit 9)

- bit de STOP

Convertorul Analog-Digital folosit pe placa de bază este de 10 biți fiind conectat la un Multiplexor Analog pe 8 canale ceea ce permite 8 intrări de tensiune.Tensiunea de referință poate fii decuplată extern la pinul AREF printr-un capacitor pentru eliminare zgomot.

Unitatea de procesare centrală are rolul unui element de conectivitate între celelalte blocuri ale microcontrolerului. Coordonează lucrul altor blocuri şi execută programul utilizatorului.

Clock-ul /ciclul instrucțiune

Clock-ul sau ceasul este starter-ul principal al microcontrolerului, şi este obținut dintr-o componentă de memorie externă numită "oscilator". Dacă ar fi să comparăm un microcontroler cu un ceas de timp, "clock-ul" nostru ar fi un ticăit pe care l-am auzi de la ceasul de timp. În acest caz, oscilatorul ar putea fi comparat cu arcul ce este răsucit astfel ca ceasul de timp să meargă. De asemenea, forța folosită pentru a întoarce ceasul poate fi comparată cu o sursă electrică.

Aplicații

Atmega8-16PI se potriveşte perfect în multe folosințe, de la industriile auto şi aplicațiile de control casnice la instrumentele industriale, senzori la distanță, mânere electrice de uşi şi dispozitivele de securitate. Este de asemenea ideal pentru cardurile smart ca şi pentru aparatele alimentate de baterie din cauza consumului lui mic.

Memoria EEPROM face mai uşoară aplicarea microcontrolerelor la aparate unde se cere memorarea permanentă a diferitor parametri (coduri pentru transmițătoare, viteza motorului, frecvențele receptorului, etc.). Costul scăzut, consumul scăzut, mânuirea uşoară şi flexibilitatea fac Atmega8-16PI aplicabil chiar şi în domenii unde microcontrolerele nu au fost prevăzute înainte (exemple: funcții de timer, înlocuirea interfeței în sistemele mari, aplicațiile coprocesor, etc.).
Programabilitatea sistemului acestui cip (împreună cu folosirea a doar doi pini în transferul de date) face posibilă flexibilitatea produsului, după ce asamblarea şi testarea au fost terminate. Această capabilitate poate fi folosită pentru a crea producție pe linie de asamblare, de a înmagazina date de calibrare disponibile doar după testarea finală, sau poate fi folosit pentru a îmbunătăți programele la produsele finite.

3.2.4. Interfața de comandă a motoarelor de curent continuu.

Partea de acționare a plăcii microcontrolerului constă în circuitul specializat L293D . Acesta este capabil sa controleze 2 motoare de c.c. în ambele sensuri de rotație sau un motor pas cu pas. Capsula L293D conține două punți H capabile să furnizeze în regim constant 600mA per canal, iar în regim puls aproximativ 1 A. In figura de mai jos se arata caracteristicile mecanice ale circuitului L293D,modul de conectare fiind reprezentat în figura

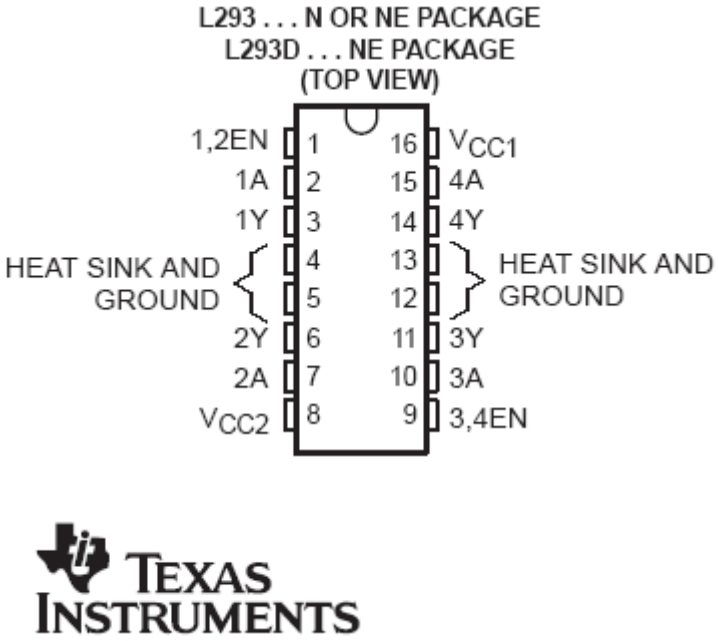


Figura 18 - Capsula DIP16 a L293D şi dispunerea pinilor

Modul de control al circuitului L293D este relativ simplu. El necesita 6 linii de port de la microcontroler care sa fie conectate la intrările input1, input2, input3 şi input4 şi la cele ouă terminale „chip enable” pentru fiecare punte H.Există şi o alta metodă mai economică din punct de vedere al numărului de pini I/O folosiţi de la microcontroler. Această metodă se bazează pe conectarea intrărilor input1, input2, input3 şi input4 pe un bus de 2 linii de comandă, practic lucru ce se materializează prin conectarea input1 şi input4 pe acelaşi pin al microcontrolerului şi input2 cu input3 similar.

INTRĂRI	FUNCȚIE	
Vm=H	C=H,D=L	Rotație CW
	C=L,D=H	Rotație CCW
Vm=L	C=D	Frânare dinamică
	C=X, D=X	Decuplare motor cu oprire

L=O LOGIC, H=1 LOGIC, X NU CONTEAZĂ C,D,Vm=intrările input 1,2 şi enable

Dezavantajele care decurg în urma acestei conexiuni sunt faptul că dubla punte H nu va fi la fel de flexibila în comanda motoarelor de c.c. şi totodată pierzându-şi capacitatea de a acționa motoare pas cu pas. În tabelul este arătată configurația stărilor logice pe intrările punții H şi starea motoarelor conectate la circuit.

Este de remarcat faptul că integratul L293D se alimentează cu două tensiuni (de la două surse). Partea logică a integratului foloseşte o alimentare compatibilă TTL (+5V pe Vss) iar partea de forță – de alimentare de putere – a motoarelor are intrare separată Vs. Scopul acestei separări o constituie minimizarea încălzirii capsulei şi are avantajul ca motoarele pot fi alimentate cu o tensiune de până la 36 de volți. Integratul este protejat termic si se opreşte alimentarea în cazul unei încălziri excesive.

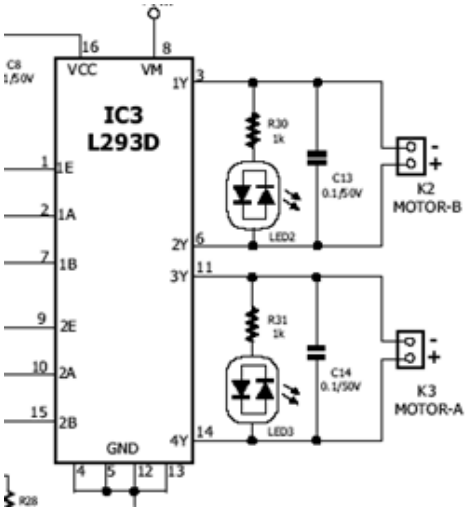


Figura19-Acționarea motoarelor bifazate

3.2.5 Circuitul integrat TL499A

TL499A este un circuit integrat proiectat pentru a oferi o gama larga de reglaj reglementate tensiunii de alimentare. Reglementarea de iesire de tensiune poate fi variat de la 2,9 V la 30 V, printr-o ajustare a doua rezistente externe.In cazul în care ac-TL499A este cuplat la linia de putere, printr-un pas în jos transformator, el functioneaza ca o serie de tensiune cc de reglementare de a menține reglementate de iesire de tensiune. Cu adaos de o baterie de la 1,1 V la 10 V, un inductor, un condensator de filtrare, si doua rezistențe, TL499A funcționeaza ca un pas de trecerea de reglementare în timpul ac-o linie de esec. Reglementate de iesire de tensiune reglabila face TL499A utile pentru o gama larga de aplicatii. Furnizarea de rezerva de putere în timpul ac-o linie de esec face ca TL499A microprocesor extrem de utile în aplicatii de memorie. Este caracterizata de TL499AC pentru exploatarea de la -20° C la 85º C

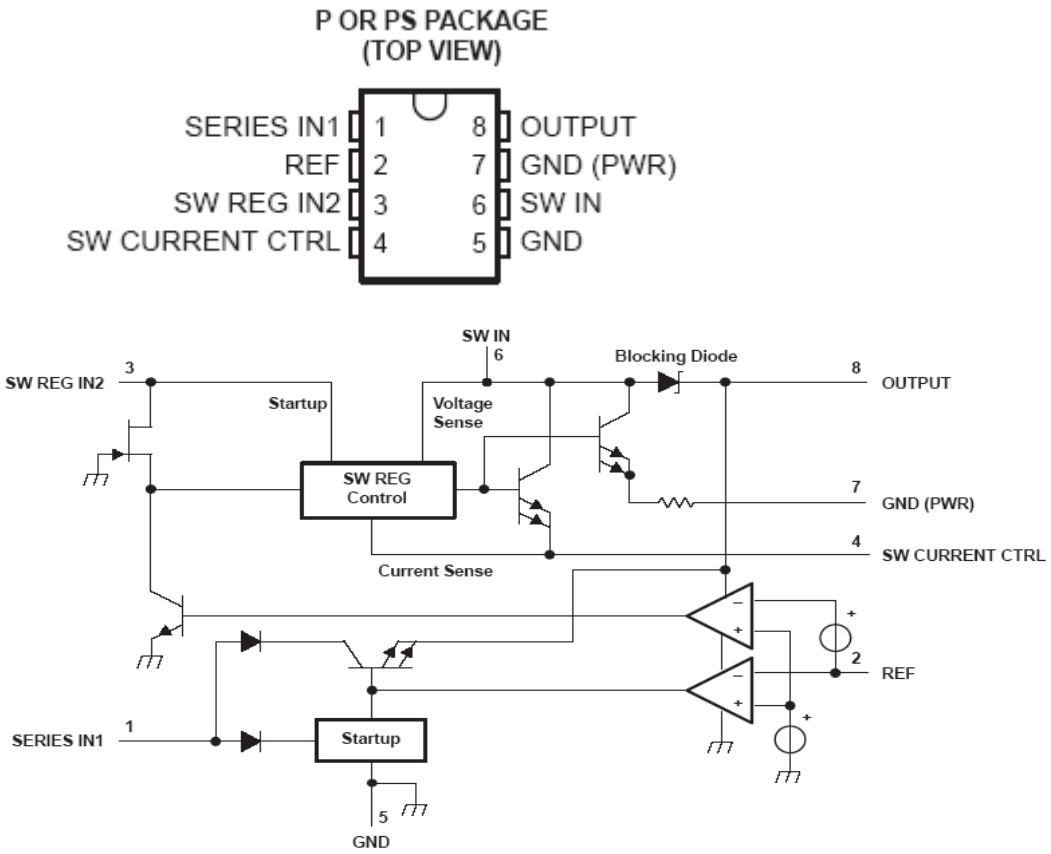


Figura 20- Schema circuitului TL499

3.3 Motoarele de curent continuu

În general, sunt similare în construcție cu generatoarele de curent continuu. Ele pot, de fapt să fie descrise ca generatoare care „funcționează invers”. Când curentul trece prin rotorul unui motor, este generat un câmp magnetic care generează o forță electromagnetică, şi ca rezultat rotorul se roteşte. Acțiunea periilor colectoare şi a

plăcuțelor colectoare este exact aceiaşi ca la generator. Rotația rotorului induce un voltaj în bobinajul rotorului. Acest voltaj indus are sens opus voltajului exterior aplicat rotorului. În timp ce motorul se roteşte mai rapid, voltajul rezultat este aproape egal cu cel indus. Curentul este mic, şi viteza motorului va rămâne constantă atât timp cât asupra motorului nu acționează nici o sarcină, sau motorul nu efectuează alt lucru mecanic decât cel efectuat pentru învârtirea rotorului. Când asupra rotorului se aplică o sarcină, voltajul va fi redus şi un curent mai mare va putea să treacă prin rotor. Astfel, motorul este capabil să primească mai mult curent de la sursa care îl alimentează, şi astfel să efectueze mai mult lucru mecanic.

Deoarece viteza rotației controlează trecerea curentului prin rotor, mecanisme speciale trebuie folosite pentru pornirea motoarelor cu curent continuu. Când rotorul se află în repaus, el, efectiv, nu are nici o rezistență, şi dacă voltajul normal este aplicat, va trece un curent mare, ceea ce ar putea avaria periile colectoare sau motorul. Mijloacele obişnuite pentru prevenirea acestor accidente este folosirea în serie a unei rezistențe, la început, împreună cu rotorul, pentru a limita curentul până când motorul începe să dezvolte un curent suficient. Pe parcurs ce motorul prinde viteză, rezistența este redusă treptat, fie manual ori automat.

Viteza cu care un motor cu curent continuu funcționează, depinde de puterea câmpului magnetic care acționează asupra rotorului, cât şi de curentul rotorului. Cu cât este mai puternic câmpul magnetic, cu atât este mai mică rata rotației necesare să creeze un curent secundar necesar pentru a contracara curentul aplicat. Din acest motiv viteza motoarelor cu curent continuu poate fi controlată prin variația câmpului curentului. Motoarele folosite sunt de tip curent continuu de tip reductor cu raport de transmisie de 48:1 alimentate la o tensiune de 3V, 6V şi o intensitate de 600 mA.Dezvoltă un cuplu de 1,27mN*m la pornire şi 0,39 mN*m în regim normal, dezvoltă 2400 rot/min .Motorul are următoarele componente:

Rotorul este format din tablă de siliciu cu bobinaj sârmă de cupru cu diametrul Φ 0.1 mm

Statorul este format din tablă de siliciu şi este bobinat cu sârmă de cupru cu diametrul Φ 0.1 mm.Diametrul interior al statorului este de 15.2 mm iar cel exterior de 20mm

2 lagăre realizate din bronz grafitat.Acestea au rol de sustinere, centrare şi rostogolire a rotorului.

Pinion din plastic fixat pe axa rotorului prin presare.Presarea este datorată elasticitații plasticului.O a doua montare a pinionului nu mai are rezistența corespunzătoare şi se utilizează altă metodă prin folosirea unor adezivi.

Figura 21-Motoare de curent continuu

3.4 Senzorul

Acesta a apărut odată cu dezvoltarea microelectronicii, împreună cu alte noțiuni de mare impact, cum ar fi cele de „microprocesor”, „microcontroller”, „transputer”, „actuator” etc.,

Astfel, o mare parte din elementele tehnice senzitive sunt încadrate în categoria de traductor. Un traductor este un dispozitiv care converteşte efecte fizice în semnale electrice, ce pot fi prelucrate de instrumente de măsurat sau calculatoare.

Prin senzor se va înțelege dispozitivul tehnic destinat înzestării maşinilor cu simțuri. Are rolul determinării unei sau unor proprietăți şi în funcție de nivelul de

integrare, poate avea funcții mai simple sau mai complexe după cum este arătat în figura de mai jos

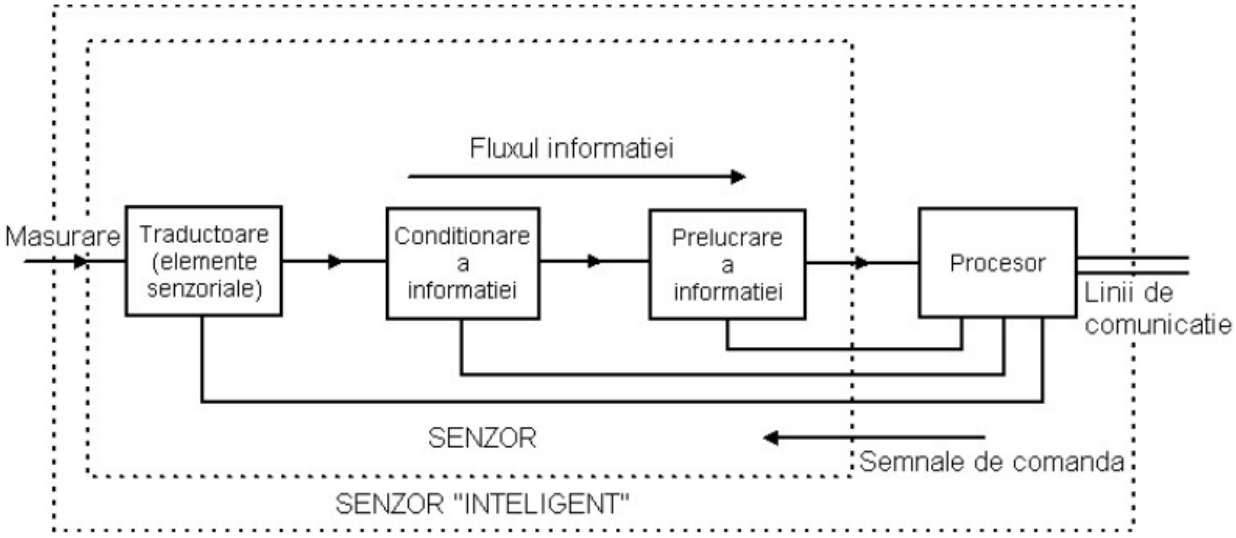


Figura 22-Principiul de funcționare al unui senzor

3.4.1 Structuri ale sistemelor senzoriale

Senzorul cuprinde traductorul/traductoarele pentru transformarea mărimii de intrare într-un semnal electric util, dar şi circuite pentru adaptarea şi conversia semnalelor şi eventual, pentru prelucrarea şi evaluarea informațiilor. Senzorul care include şi unitățile micromecanice şi microelectronice de prelucrare, realizate prin integrare pe scară largă (LSI) sau foarte largă (VLSI), se întâlneşte în literatura de specialitate şi sub denumirile de "sistem senzorial" sau "senzor inteligent" (smart-sensor). Producerea senzorilor inteligenți este facilitată de dezvoltarea tehnicii microsistemelor, care permite integrarea în volume extrem de mici atât a traductoarelor de diferite tipuri, cât şi a micromecanicii şi microelectronicii de prelucrare.

Gruparea senzorilor după acțiunea robotului cu mediul înconjurător :

Senzorii interni (intero-receptori), care servesc la obținerea unor informații legate de funcționarea robotului, cum ar fi poziția relativă a elementelor cuplelor

cinematice, vitezele şi acceleraţiile liniare şi unghiulare, deformaţiile elementelor lanţului cinematic ş.a.

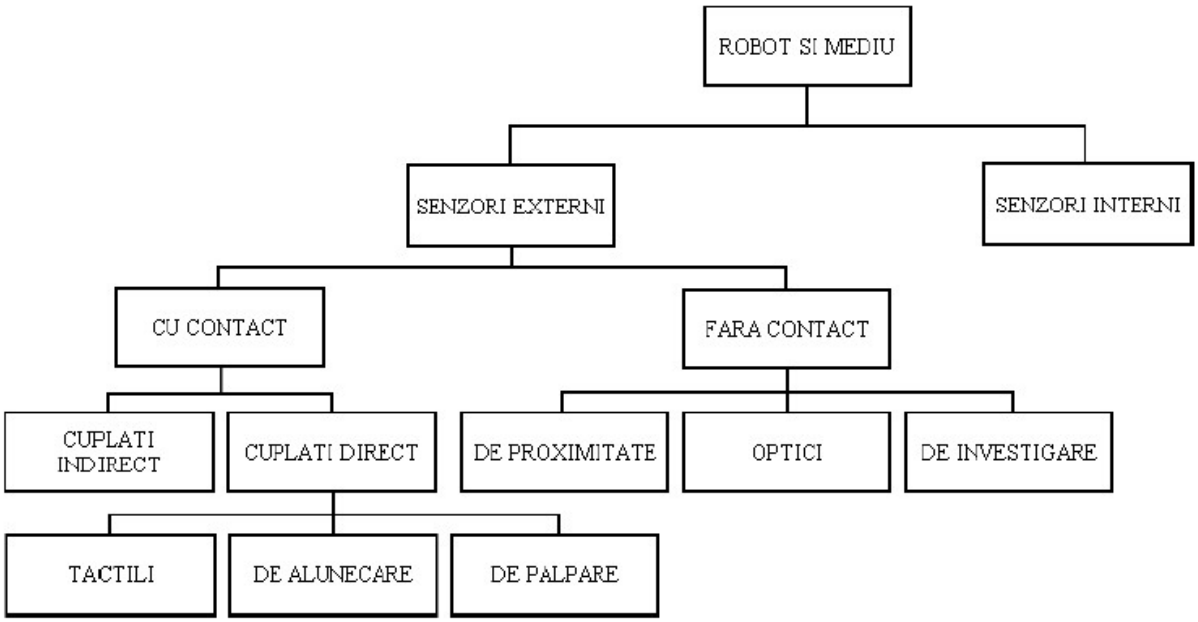


Figura 23-Tipuri de senzori

.

Senzori externi (denumiţi de unii autori şi extero-receptori), utilizaţi pentru culegerea unor informaţii asupra mediului înconjurător şi asupra interacţiunii robot/mediu; servesc la identificarea prezenţe şi stabilirea tipului, poziţiei, orientării, culorii sau a altor proprietăţi ale obiectelor din mediu, la identificarea unor obstacole, la determinarea forţelor de interacţiune robot/mediu.

Un criteriu care poate permite clasificarea senzorilor externi este cel referitor la contactul cu obiectele din mediu . Un senzor care măsoară poziţiile/deplasările în cuplele cinematice este un senzor intern, un senzor de investigare, care baleiază mediul înconjurător pe principiul radarului, este un senzor extern fără contact, un senzor tactil este un senzor extern cu contact direct, iar un senzor de forţă/moment este un senzor extern cu contact indirect, întrucât forţele de interacţiune cu mediul nu

sunt exercitate direct asupra senzorului, ci sunt resimțite de acesta prin propagarea lor de-a lungul unor elemente intermediare.

Senzorii fără contact pot servi la recunoaşterea obiectelor, a poziție şi orientării lor şi la controlul calității. Senzorii de proximitate sunt senzori de zonă foarte apropiată; ei furnizează informații despre existența obiectelor şi sunt montați pe efectorul final sau în apropierea acestuia. Senzorii optici sunt senzori de zonă apropiată, iar cei de investigare de zonă îndepărtată.

3.4.2 Senzorii folosiți în echiparea minirobotului

3.4.2.1 Senzorul infraroşu Sharp GP2D120

Un senzor special folosit pentru evitarea obstacolelor este senzorul infraroşu Sharp GP2D120 care prezintă urmatoarele caracteristici tehnice:

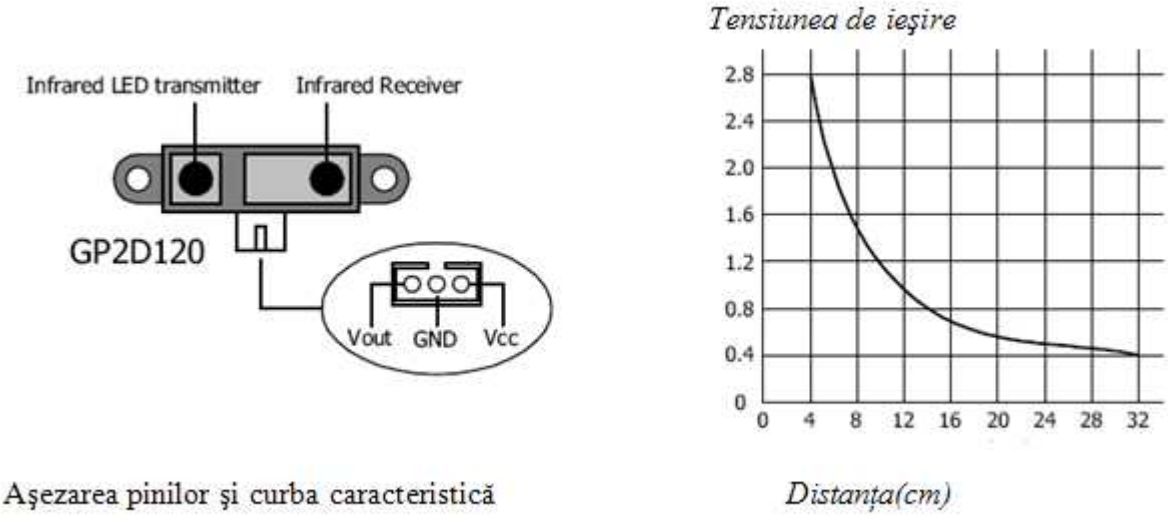
- Foloseşte principiul de reflexie al luminii infraroşii pentru măsurarea distanței,
- Raza de acțiune este intre 4 şi 30 de cm.
- Alimentarea se face la o tensiune de 4,5 pâna la 5 V şi o intensitate de 33 mA.
- Tensiunea de ieşire este cuprinsă în intervalul 0.4-2.4 V la o alimentare de 5 V.

Interfata sa prezinta 3 fire: alimentarea(Vcc), masa(GND) şi tensiunea de ieşire(Vout) şi necesită un conector JST de 3 pini.

Modul de funcționare

Principiul care stă la baza acestei metode de masurare folosite de senzorul Sharp se bazează pe formarea unui triunghi între Emițătorul de rază infraroşie, punctul de reflexie şi detectorul de infraroşu. Astfel, emitatorul emite un puls de lumina infrarosie. Lumina străbate campul vizual şi fie loveşte un obstacol sau continuă

deplasarea. În cazul inexistenței unui obstacol, lumina nu va fi reflectată şi nu se va detecta nici un obiect. Dacă lumina se reflectă de pe un obiect, se va întoarce la detector şi va creea un triunghi între emițător, punctul de reflexie şi eetector, ca în figura urmatoare:



Aşezarea pinilor şi curba caracteristică

Figura24-Date tehnice privind funcționarea senzorului Sharp

Prin conexiunea senzorului infraroşu cu convertizorul Analog/Digital rezultă datele analogice care sunt convertite în distanțe corecte cu ajutorul softului de programare.Astfel pentru o valoare de 307 corespunde o distanța de 8 cm.

R = (2933 / (V + 20)) – 1

R=distanța in cm.

V=datele anlogice din conversia analog/digitală.Intervalul este intre 0 şi 1,023

$$L = \frac{F \times A}{X}$$

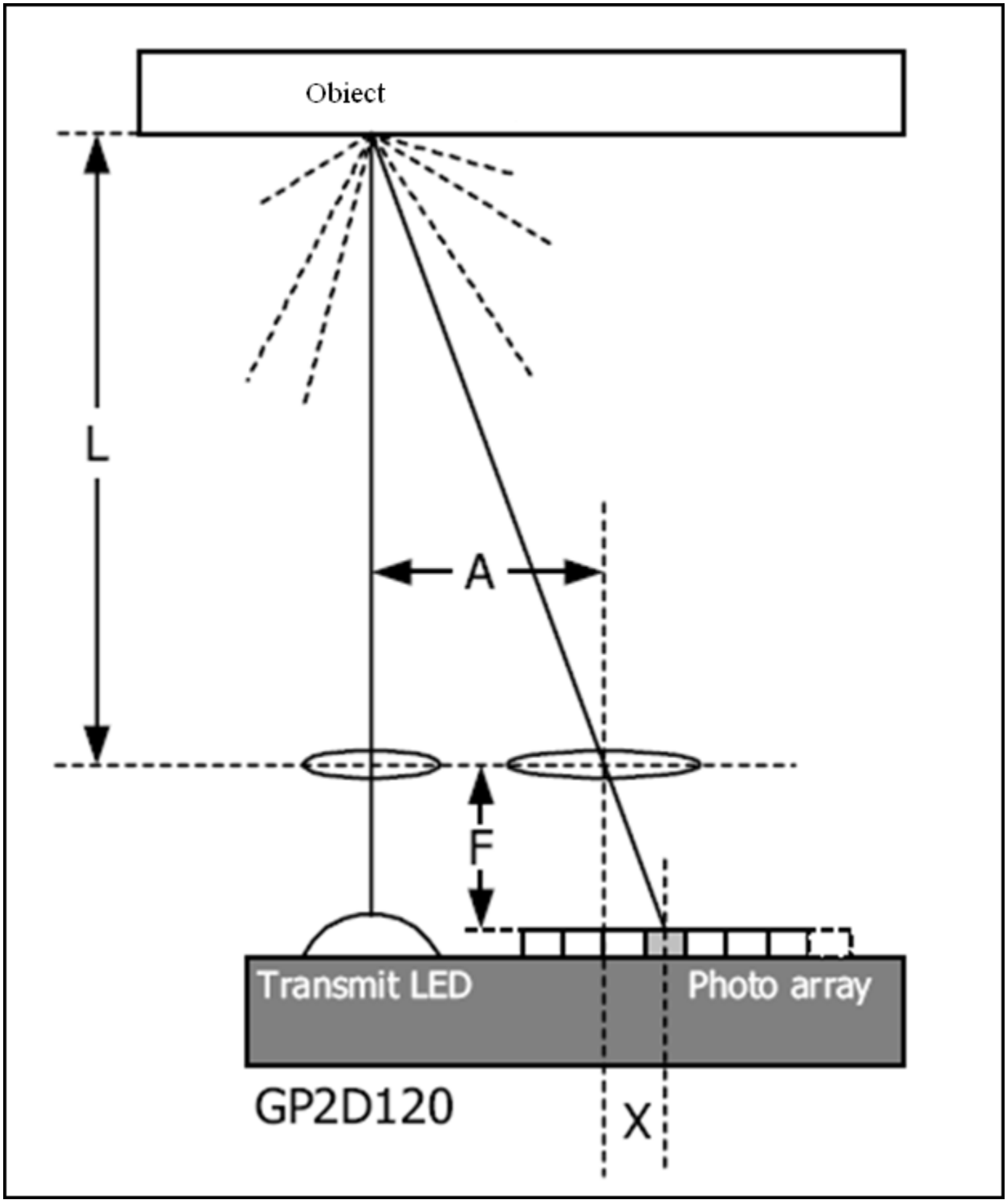
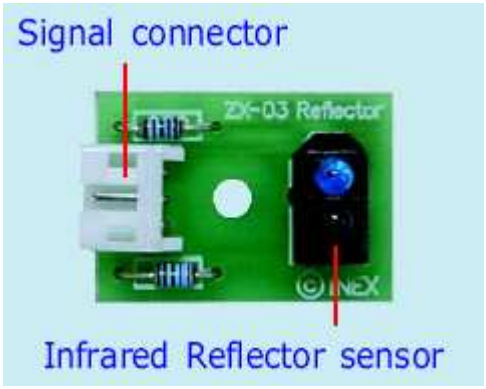


Figura25-Modul de funcţionare al senzorului Sharp

Tensiune de iesire	Date analogice	Distanta in cm
0.4	82	33
0.5	102	26
0.6	123	22
0.7	143	19
0.8	164	16
0.9	184	14
1.0	205	13
1.1	225	12
1.2	246	11
1.3	266	10
1.4	287	9
1.5	307	8
1.6	328	8
1.7	348	7
1.8	369	7
1.9	389	6
2.0	410	6
2.1	430	6
2.2	451	5
2.3	471	5
2.4	492	5
2.5	512	5
2.6	532	4

Tabelul cu valori ale senzorului infrarosu

3.4.2.2 Reflectorul infraroşu



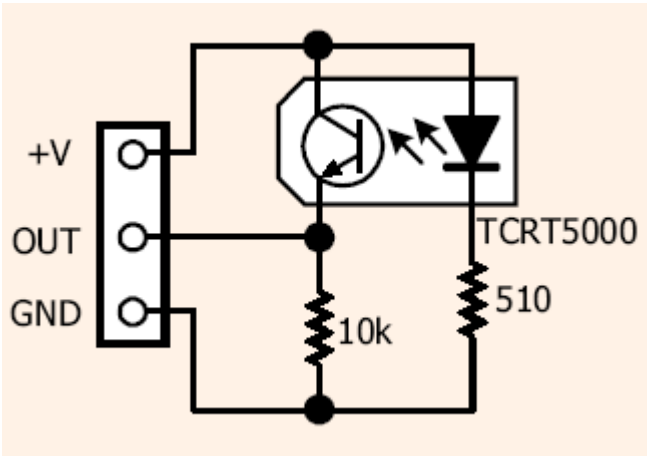


Figura26-Reflectorul infraroşu vedere şi schemă circuit

Inima acestui circuit este senzorul care detectează reflecţiille de la lumina infraroşie.Este format dintr-un LED Infraroşu ce emite lumină infraroşie pe suprafață.Foto-tranzistorii vor primi lumina infraroşie reflectată.Dacă nu se reflectă lumină infraroşie, terminalul OUT are voltaj scăzut la măsurare.În cazul primirii de lumină infraroşie trece curent prin fototranzistori în funcţie de intensitatea luminii care variază în funcţie de distanţa de reflexie.Senzorul TCRT 5000 poate fii folosit pentru o distanţă de 0.1-1.5 cm.La o tensiune de ieşire de 0.5-5V datele de ieşire variază între 30 şi 1023.Este folosit pentru a detecta culoarea suprafeței(albă sau neagră).Astfel pentru suprafața albă valoarea este între 600 şi 950 iar pentru suprafața neagră valoarea este între 100 şi 300.

Valoarea de referință pentru a detecta linia este (600+100)/2= 350

3.4.2.3 Receiver infraroşu ZX-IRM

În transmiterea semnalelor prin lumină infraroşie pentru distanțe de 5 10 m funcționează pe principiul telecomenzii de televizor.Frecvența de emitere este de 38 kHz, receiverul demodulează semnalul de frecvența şi îl trimite ca date seriale la microcontroler

Dacă senzorul nu detectează frecvența de 38 kHz cu lumină infraroşie ieşirea este logic 1,in caz contrar ieşirea e logic 0.

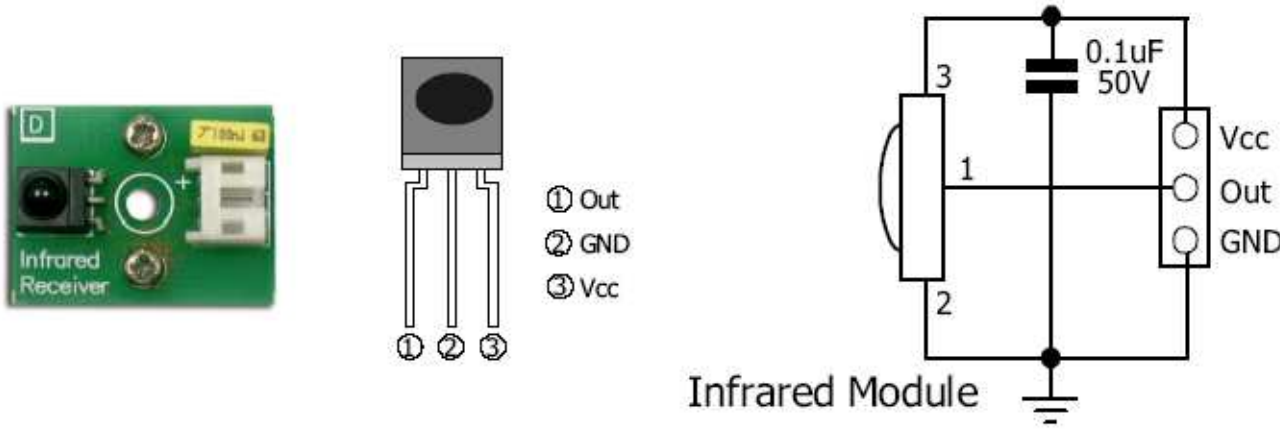


Figura27-Receiver infraroşu vedere şi schema circuit

3.4.2.4 Telecomandă cu emisie infraroşu:

Această telecomandă funcționează cu comunicarea serială.Datele transmise manual de la butoane vor fii modulate la o frecvență de 38kHz.Pentru aceasta trebuie instalat pe robot receiverul infraroşu.

Caracteristici ale telecomenzii:

- Distanța de operare este între 4 şi 8 metri în spațiu deschis

• Are 4 butoane activate de 4 întrerupătoare

• Consum redus de putere.Trece în modul power-save odată ce datele au fost transmise

• Transmite date serial folosind standardul RS-232 cu o rată de transfer de 1200 bps şi formatul datelor 8N1(8 biți de date nici un bit de paritate şi un bit de stop.)

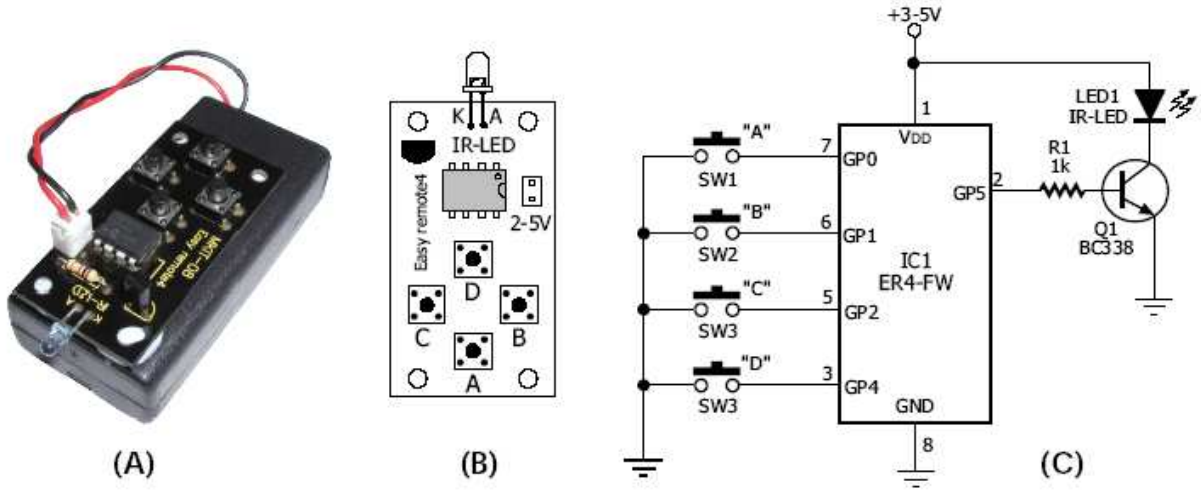


Figura 28- A-Poza. B-Vedere a planului telecomenzii C-Schema circuitului.

3.3.2.5 Afişor de cristale lichide(32 caractere 2 linii 16)

Este un senzor LCD care comunică cu microcontrolerul printr-o interfață serială.

• Acceptă date transmise serial la la o frecvență de 2400 sau 9600 bps(biți pe secundă) cu format data 8N1(8 biți de date nici un bit de paritate şi un bit de oprire).

• Acceptă nivelul TTL sau RS-232 prin selectarea a 2 jumperi.Suportă controlerul standard LCD HITACHI HD44780 sau SEIKO EPSON SED1278 compatibil.

- Este operațional la o tensiune de alimentare de 5÷ 12 volți.

- Conectorul de interfață are :+5 V tensiune de alimentare(+), intrare serială a datelor(S) şi masa(G).

Majoritatea caracterelor LCD nu pot fii schimbate deoarece sunt înmagazinate în memoria ROM. Primele 8 simboluri care corespund ASCII 0..7 sunt înmagazinate în memoria RAM(CGRAM).Ele sunt afişate în mărimea 5 cu 8 puncte.

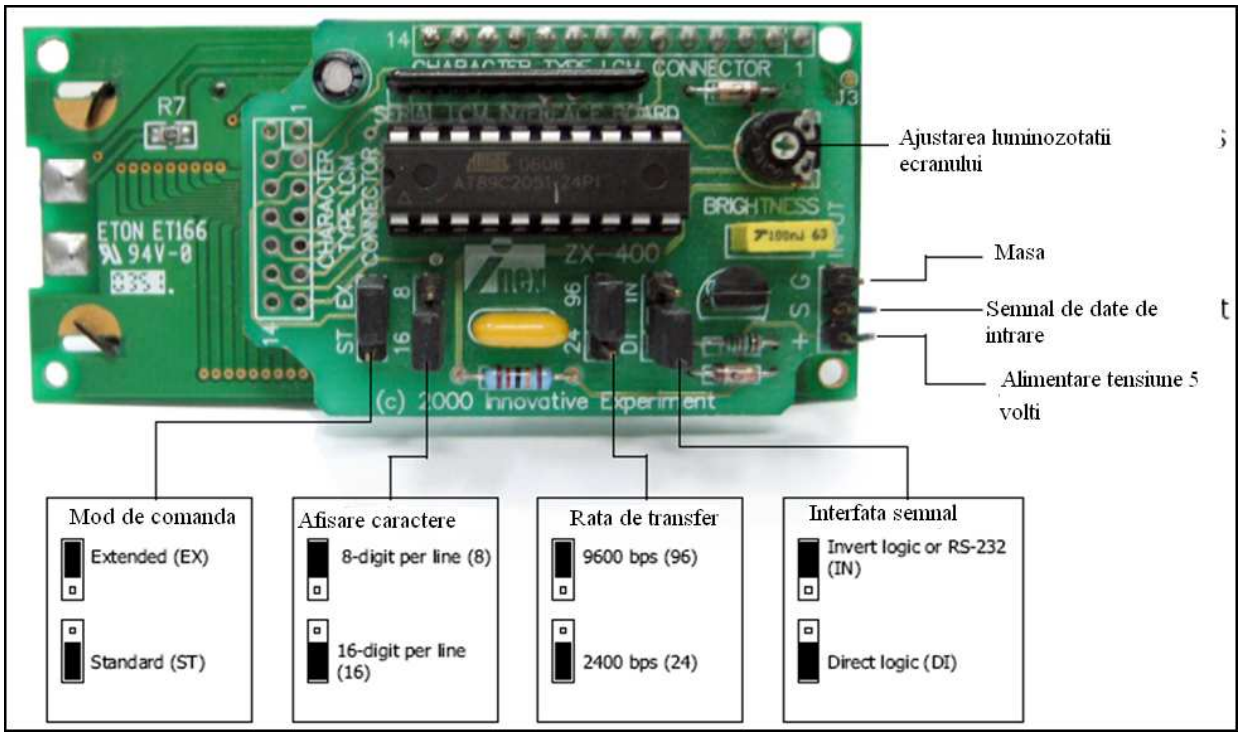


Figura 29-Detalii despre selectarea jumperilor

1 Modul de comandă al jumperilor.SLCD are 2 moduri de comandă.Unul este de tip Standard.Acesta este compatibil cu tipul Scott Edwards's iar celălalt este de tip Scott Edwards's.

2 Numărul de caractere pe linie.Se poate seta 8 caractere sau 16 caractere pe linie.

3 Rata de transfer a datelor.Se poate seta 9600 biți pe secundă sau 2400 biți pe secundă cu formatul datelor de tip 8N1(8 biți de date nici un bit de paritate şi un bit de stop).

4 Semnalul de interfață. 2 selecții de tip Invert logic TTL/CMOS sau RS-232 şi de tip Direct logic(DI).

3.3.2.6 Senzorul de temperatură.

Detectează temperatura ambientului,lucrează cu un thermistor ntc(coeficient negativ de temperatură) şi acest traductor analog are 2 ieşiri: reacțiune pozitivă şi negativă a tensiunii aplicate în funcție de temperatura ambientului.Temperatura sa normală de lucru e de 25^0C.

Figura 30-Senzorul de temperatură.

3.3.2.7.Contactorii

Contactorii folosiți la acest robot au următoarea schemă electrică:

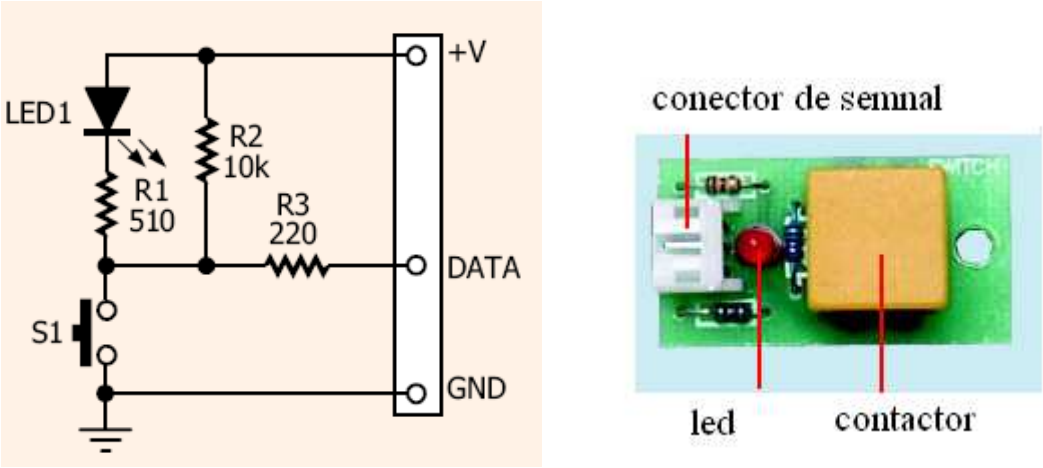


Figura 31-Senzorul de contact vedere şi schemă circuit

Când contactorul nu este apăsat rezultă ”1” logic în iar cand este apăsat arată ”0” logic şi se aprinde ledul. Aceşti senzori sunt folosiți pentru detectarea unor obstacole.

CAPITOLUL IV

Proiectarea interfețelor de comunicare

Sistemele de calcul pot să comunice în două moduri: paralel şi serial. În modul paralel, de obicei se utilizează 8 sau mai multe linii (cabluri). Exemple de utilizare a comunicației paralele sunt imprimanta sau hard discul. Comunicația paralelă se utilizează doar pentru distanțe foarte scurte (câțiva metri). Motivul este de fapt că pentru cablurile lungi, semnalele sunt atenuate şi pot apare distorsiuni. În plus nu este deloc de neglijat costul cablurilor şi problemele ce pot apare la conexiuni. Avantajul comunicației paralele este viteza mare de transmisie.

Comunicația serială este utilizată pentru sisteme aflate la mare distanță intre ele.

Comunicația serială utilizează un număr redus de cabluri. În comunicația serială, datele transmise trebuie serializate la transmisie şi deserializate la recepție. Pentru aceasta, la transmisie se utilizează un registru paralel-serie iar la recepție un registru serie-paralel.

Registrul paralel-serie este utilizat pentru transformarea unei date de 8 biți intr-un şir de biți.

Registrul serie-paralel este utilizat pentru transformarea unui şir de 8 biți intr-un octet.

Pentru distanțe lungi, comunicația paralelă poate utiliza linia telefonică. În acest caz

semnalele logice 0 şi 1 trebuie transformate în semnale sinusoidale. Această conversie este realizată de un dispozitiv numit modem (MODulator/DEModulator). Pentru distanțe scurte această conversie nu este necesară. De exemplu, tastatura şi mouse-ul comunică serial

informația către unitatea centrala a PC.

Comunicația serială utilizează două metode:

- sincronă: se transferă blocuri de octeți (caractere)

- asincronă : se transferă câte un octet

Se pot realiza programe pentru realizarea comunicației seriale însă acestea pot fi lungi. Este indicat ca programatorul să se concentreze asupra problemelor specifice aplicației şi nu asupra unei probleme extrem de generale precum comunicația serială. Aici intervin şi aspectele economice legate de productivitatea muncii. Din acest motiv producătorii de circuite au realizat circuite ce rezolvă problemele legate de comunicația serială. Aceste circuite se numesc UART (Universal Asyncronus Receiver Transmitter).

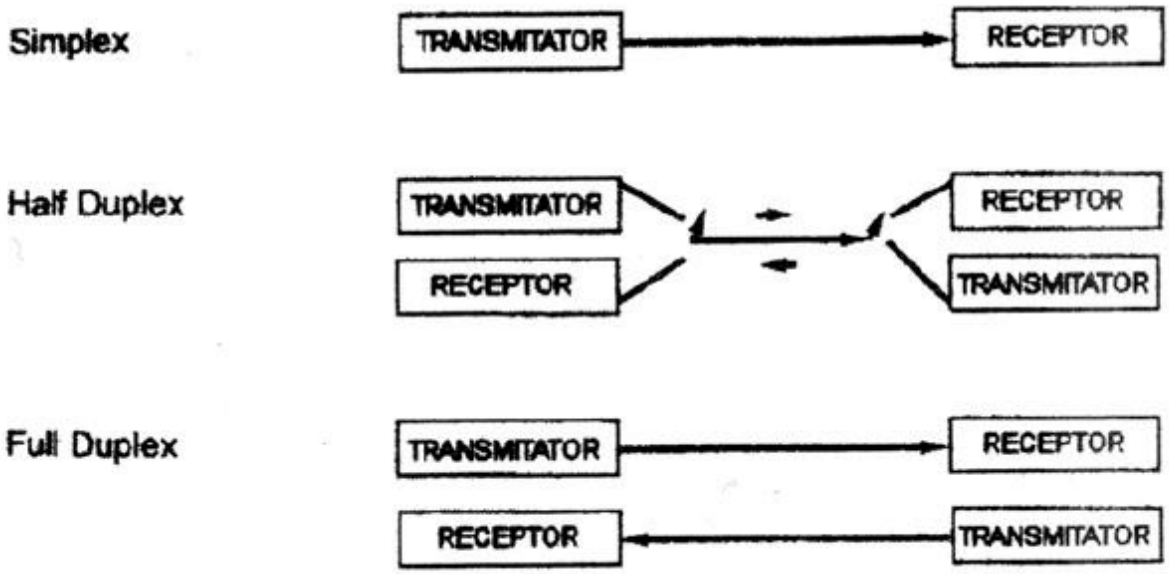


Figura 31 - Metode de comunicație

Microcontrolerul ATMega8 conține un asemenea modul (UART) înglobat. Din acest motiv în cele ce urmează ne vom referi doar la comunicația asincronă. Dacă datele pot fi transmise şi recepționate simultan, se spune ca avem comunicație „full duplex”

Dacă datele nu pot fi trimise şi recepționate simultan, se spune ca avem comunicație „half duplex” . Aici termenul „simultan” se refera la faptul ca ambele sisteme pot transmite date în

acelaşi moment de timp. O altă situație este cazul în care unul din sisteme este doar un transmițător celalalt este doar un receptor (comunicație „simplex”). Pentru comunicația full

duplex sunt necesare două fire (plus firul de masă).

a) Comunicația asincronă

Cele două sisteme, pentru a putea comunica intre ele, trebuie sa respecte o serie de reguli. Trebuie stabilite convenții referitoare la:

- modul de împachetare al datelor;
- numărul de biți ai unui caracter;
- cum se identifica începutul unei date;
- cum se identifica sfârşitul unei date;
- care este durata unui bit.

b) Formatul datelor

Comunicația asincronă este frecvent utilizata pentru transmisia orientată pe caracter; comunicația sincronă este utilizata pentru transmisia orientată pe blocuri de date. În comunicația asincronă fiecare caracter este încadrat intre biți de start şi biți de stop. De fapt totdeauna se foloseşte un bit de start (0 logic) şi de obicei un bit de stop (1 logic).

În figura de mai jos este reprezentat modul de transmisie al caracterului ASCII „A” al cărui cod binar este 0100 0001. Se observa ca transmisia biților se face în ordinea: START(0) , D0(1), D1(0)… D7(0), STOP(1).

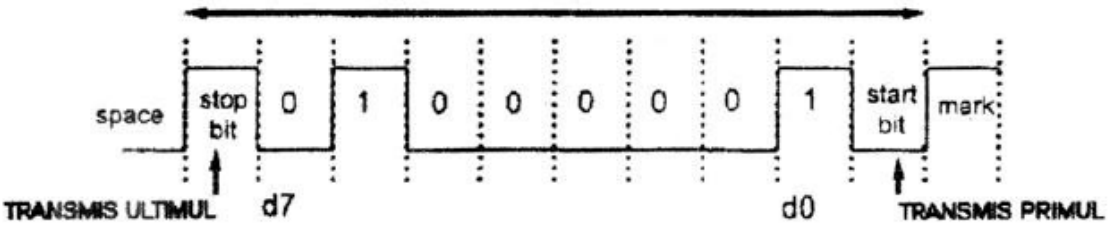


Figura 32 - Transmisia caracterului ASCII „A”

Se transmite mai întâi bitul cel mai puțin semnificativ al octetului (LSB – D0). Se observa că de fapt pentru transmisia unui caracter se utilizează 10 biți. Sunt şi alte posibilități de transmisie. De exemplu se poate prevede un bit suplimentar care sa reprezinte paritatea octetului. Scopul acestui bit suplimentar este verificarea păstrării integrității datelor. De aceea, programatorul trebuie sa cunoască datele de catalog ale portului serial şi modul lui de programare.

c) Rata de transfer

Rata de transfer reprezintă numărul maxim de biți ce pot fi transmişi intr-o secundă.

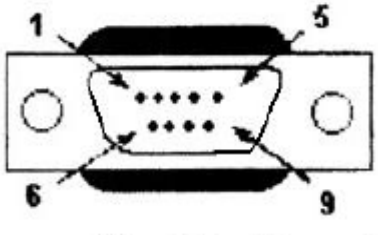
Este dată în bps (biți per second). Alt termen des utilizat pentru bps este baud rate. Cei doi termeni nu sunt identici. Termenul baud rate este preluat din terminologia modemurilor unde este definit ca numărul de schimbări pe secundă ale unui semnal. Sunt situații când o singură modificare a semnalului conduce la transferul mai multor biți de date. Totuşi şi dacă nu se utilizează modemul se pot utiliza ambii termeni. Rata de transfer depinde de portul de comunicație al sistemului. De exemplu, pentru PC-urile mai vechi rata de transfer de date este 100… 9600 bps. Noile PC-uri suporta o rata maximă de transfer de 115.2 Kbps.

d) Standardul RS232

Pentru a permite compatibilitatea echipamentelor realizate de diferiți producători, s-a adoptat standardul numit RS232 realizat de EIA (Electronics Industries

Association) în 1960, modificat apoi mai tâ rziu în mai multe rânduri. Standardul actual RS232C adoptat în 1969 este utilizat de PC-uri şi de numeroase alte echipamente. Deoarece standardul a fost adoptat înainte de apariţia familiei logice TTL, standardul nu este compatibil cu nivelele TTL. Nivelul 1 logic este reprezentat de o tensiune electrică cuprinsa intre -3 şi -25V iar nivelul logic 0 este reprezentat de o tensiune electrica cuprinsa intre +3 şi +25V, zona situata intre -3V şi 3V fiind nedefinită.

e) Pinii interfeţei seriale RS232 Tabelul 9



Interfata RS232 conector DB-9	
Pin	Descriere
1	Data carrier detect (DCD)
2	Received data (RxD)
3	Transmitted data (TxD)
4	Data terminal ready (DTR)
5	Signal ground (GND)
6	Data set ready (DSR)
7	Request to send (RTS)
8	Clear to send (CTS)
9	Ring indicator

Figura33-conectorul DB-9

f) Semnalele de control (handshaking)

Pentru a se realiza rapid şi sigur comunicaţia intre două echipamente, trebuie ca transferul de date să ţină cont de unele situaţii specifice. De exemplu se poate întâmpla ca unul din echipamente să nu mai poată primi date datorită faptului că nu mai are unde să le depună. Într-un asemenea caz el trebuie să informeze transmiţătorul că trebuie să aştepte. Se definesc următoarele semnale de control mai des utilizate:

- DTR (data terminal ready). După alimentarea terminalului şi după ce trece prin

rutinele de inițializare, el trimite semnalul DTR ce indică faptul ca este gata pentru comunicație. Semnalul este de intrare pentru modem (DCE).

- DSR (data set ready). Este un semnal emis de DCE (modem) ce indică că este gata să primească date de la DTE.

- RTS (request to send). Este un semnal prin care DTE informează DCE ca poate să transmită o dată.

- CTS (clear to send). Este un semnal de răspuns pentru semnalul RTS prin care DCE

informează DTE că este gata să primească date. Acest semnal este utilizat de DTE

pentru a începe transmisia datelor.

4.1 PX-400 Cutie Programator In-Sistem cu interfață serială

Este folosit pentru programarea codului in memoria flash a microcontrolorului AVR.

- Conectarea cu calculatorul se face prin portul serial RS-232.Dacă calculatorul are doar port USB,poate fii folosit un convertor USB la Serial.UCON-232S este recomandat in această situa□ie.

- Programarea microcontrolorului se face prin cablu ISP.Suportă citirea scrierea ştergerea şi protejarea datelor.

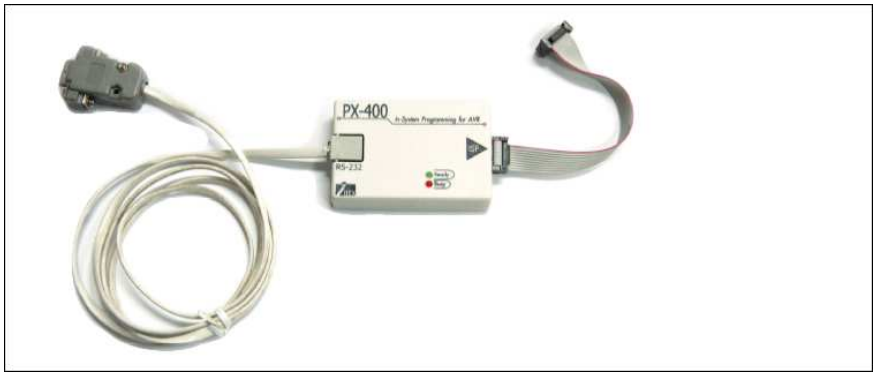


Figura 34-Cablul PX-400

- Necesită o tensiune de alimentare de 5V de pe placa microcontrolorului.

- Este operabil cu softul AVR Prog. Acest soft este inclus in AVR Studio fiind compatibil deasemenea cu softul Avr-Osp II.

4.2 Interfața SPI

Interfața serie SPI (serial peripheral interface) este o interfață sincronă, de mare viteză, pentru transferul datelor între un microcontroler AVR şi un dispozitiv periferic sau între două microcontrolere AVR.

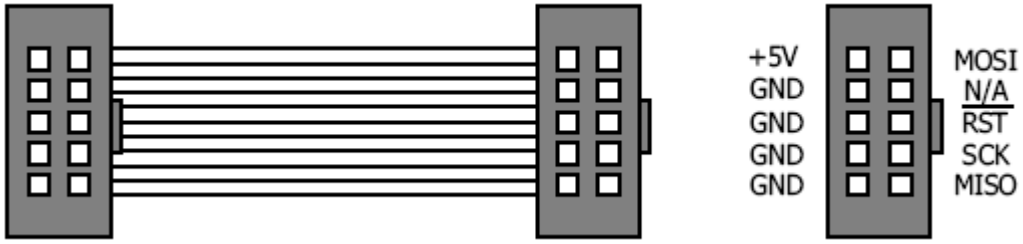
Transferul pe interfața SPI se face cu următoarele caracteristici: transfer full duplex (pe trei fire), operare master slave, transfer cu primul bit LSB sau MSB, patru viteze de transfer, semnalizare sfârşit de transmisie şi activare din modul de consum redus. Scrierea datelor în registrul de date al dispozitivului master va genera activarea generatorului de ceas şi data va fi transferata bit cu bit pe linia MOSI, prin intermediul registrului de deplasare.

4.3 Cabluri de transfer a datelor

Rolul cablurilor de semnal este de a conecta placa de bază cu diferiți senzori şi cu calculatorul.Cablul ISP este folosit pentru programarea microcontrolorului, cabluri PCB3AA-8 pentru interconectarea cu senzori şi un cablu port serial pentru conexiunea dintre calculator si placa de baza la conectorul ISP.

4.3.1 Cablul ISP

Este un cablu tip panglică cu 10 fire.Este folosit pentru conexiunea dintre cutia de programare ISP şi placa microcontrolorului.Modul de distribuție al firelor este arătat in schema de mai jos



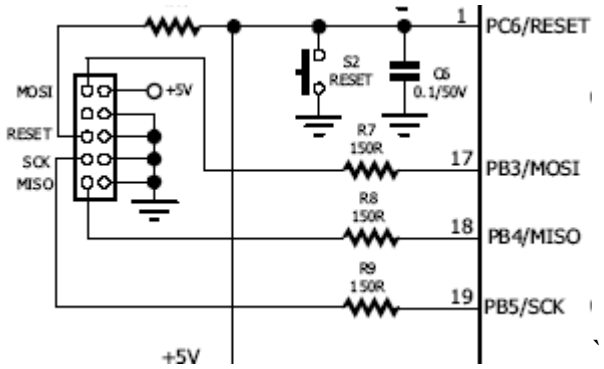


Figura 35-Modul de conectare al cablului ISP

4.3.2 Cablul JST3AA-8

Este un cablul standard de la INEX cu lungimea de 20 cm folosit pentru conexiunea dintre senzori şi placa microcontrolerului.Aşezarea firelor este arătată în figura de mai jos

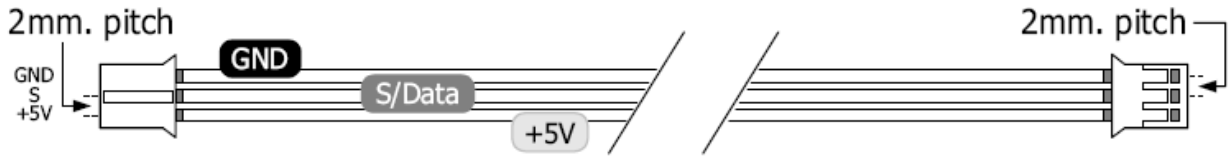


Figura 36-Modul de conectare al cablului ISP

4.3.3 Cablul port serial CX-4

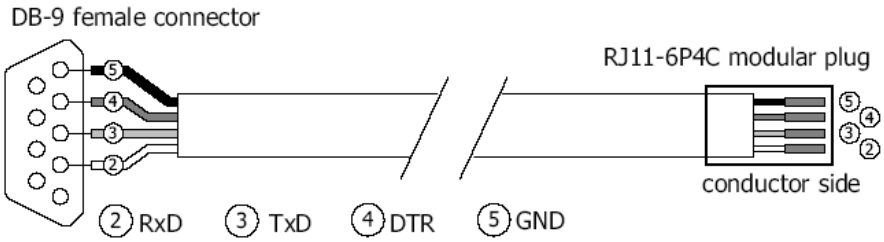


Figura 37-Modul de conectare al cablului port serial CX-4

Nume semnal	Semnificatia/Functia
RXD	Receive Data - receptie date

TXD	Transmit Data – transmisie date
GND	Masa digitală
DTR	Data Terminal Ready – terminal pregătit pentru transmisie

Este folosit pentru conexiunea dintre portul serial al calculatorului RS-232 şi dispozitivele externe cum ar fii placa microcontrolorului.Un capăt de conector este de tipul DB-9 feminin iar celalalt de tip RJ-11 6P4C(6 pini şi 4 contacte).Are o lungime de 1.5 metri iar schema lui este arătată în figura de miai sus.

CAPITOLUL V

PROGRAMAREA ROBOTULUI

Microcontrolerul care echipează robotul, ATMEL ATmega8-16PI face parte din generațiile recente ale producătorului american şi suportă programarea pe o platformă standardizată pentru microcontrolerele AVR RISC pe 8 biți, folosind limbajul de programare C. Ca soft enumerăm AVR Studio pentru programarea microcontrolerului şi un compilator CAVR 1 AVR Studio:program de simulare, soft dezvoltat de Corporația Atmel.Acest soft ruleaza sub Microsoft Windows95 şi Microsoft Windows NT.Windows XP SP2 este recomandat acum.Fişierul compilat este de tip HEX şi este coborît în memoria plãcii de bazã cu ajutorul cutiei de programare PX-400.

2 CAVR este un compilator pentru C.

Bibliotecile sunt fişiere suport care permit programatorului o mai uşoară integrare in limbajul de programare C

Softul programator:Acest soft este folosit pentru downloadarea fişierului compilat HEX.Acesta este AVRProg şi este o componentă în AVR Studio

5.1 Instalarea AVR Studio

Primul program care se instalează este AVR Studio aplicație livrată de către producător.Pentru a contiua instalarea trebuie acceptați termenii de licență impuşi de producator.

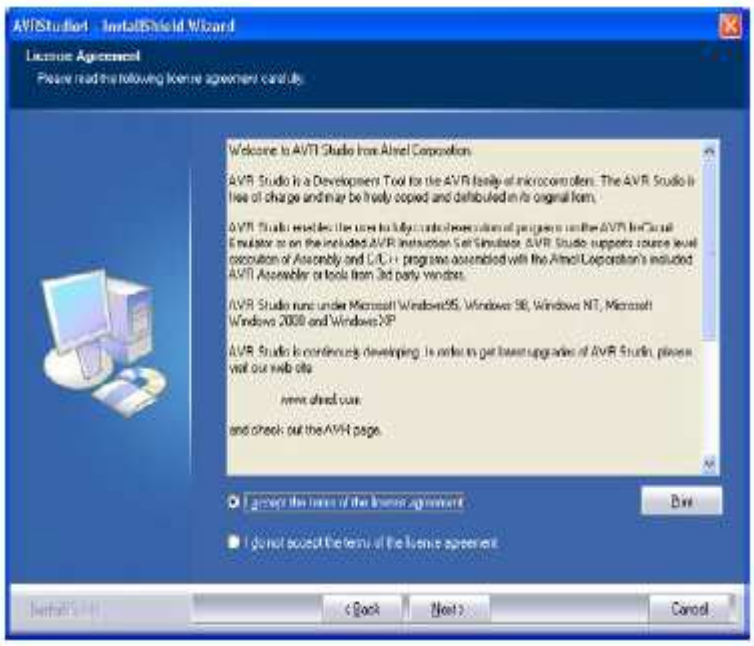


Figura38-Instalare AVR Studio

După instalarea şi deschiderea programului va apare fereastra principală AVR Studio care arată precum în figura de mai jos.În stânga sus se găseşte fişierele şi librăriile care sunt apelate în programul construit în C. În fereastra principală este scris programul C, iar dedesubt se află fereastra de execuție care arată starea proceselor care includ rezultatul compilat precum şi erorile de compilație a fişierului HEX precum şi alte mesaje de avertizare.Deasupra se află bara de meniuri. Paşii de dezvoltare a unui program C în AVR Studio sunt

- Scrierea programului cu editorul text AVR Studio V4.0
- Compilarea codului C în codul de asamblare pentru microcontroller

- Convertirea codului de asamblare în fişier format HEX
- Downloadarea codului în memoria program a microcontrolerului
- Rularea programului de catre microcontroler.

De asemenea este necesar ca după instalarea softului fişerele librărie să fie copiate separat într-un dosar pentru a fii apelate mai apoi de program.

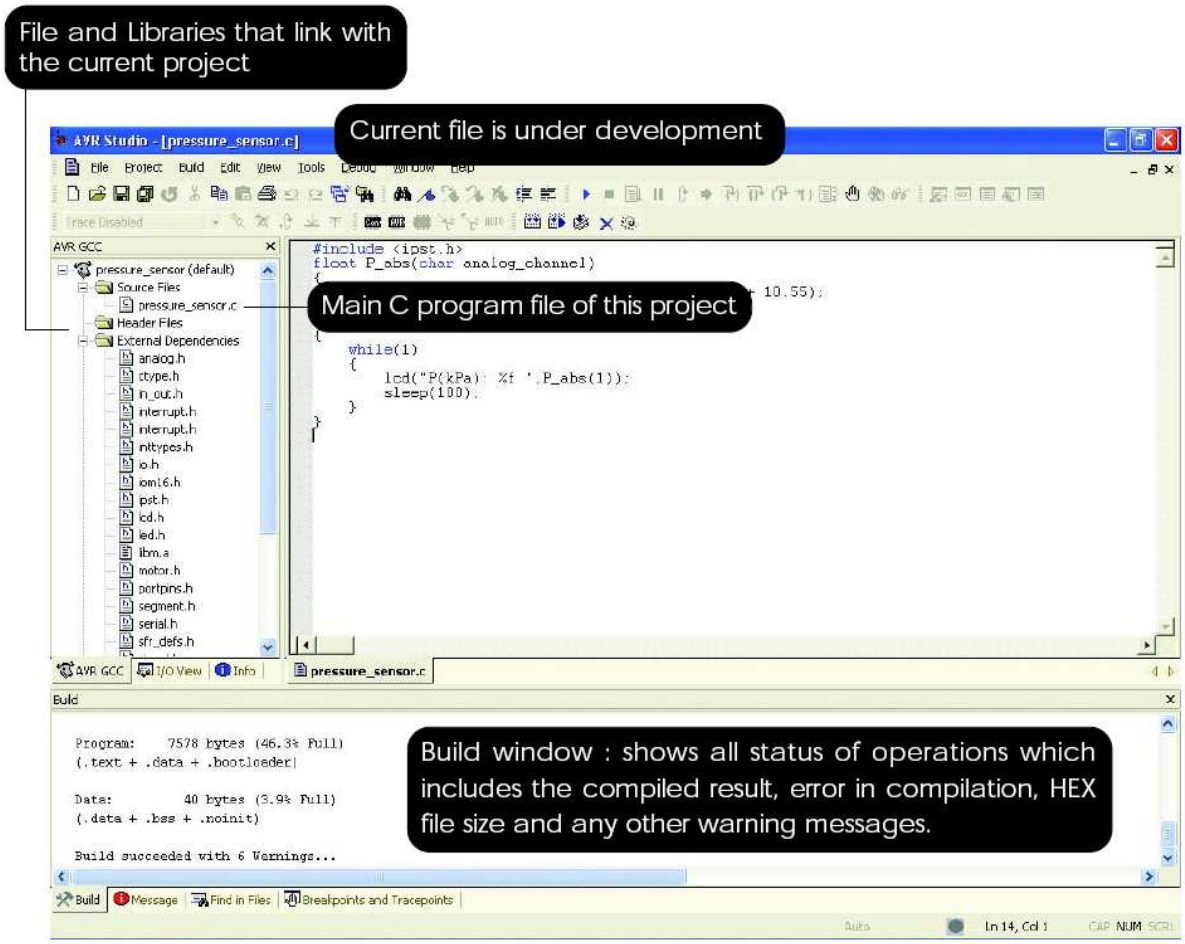


Figura 39-Bara de meniuri a programului AVR Studio

5.2 Crearea programului C în AVR Studio

Din Project se selectează Configuration Options unde la iconul General se determină următoarele date astfel: dispozitiv atmega8 şi frecvența de 16000000 Hz

Figura 40-Modificare în opțiunile proiectului.

Pentru apelarea librăriilor se selectează iconul Include Directories şi se accesează folderul în care sunt localizate.

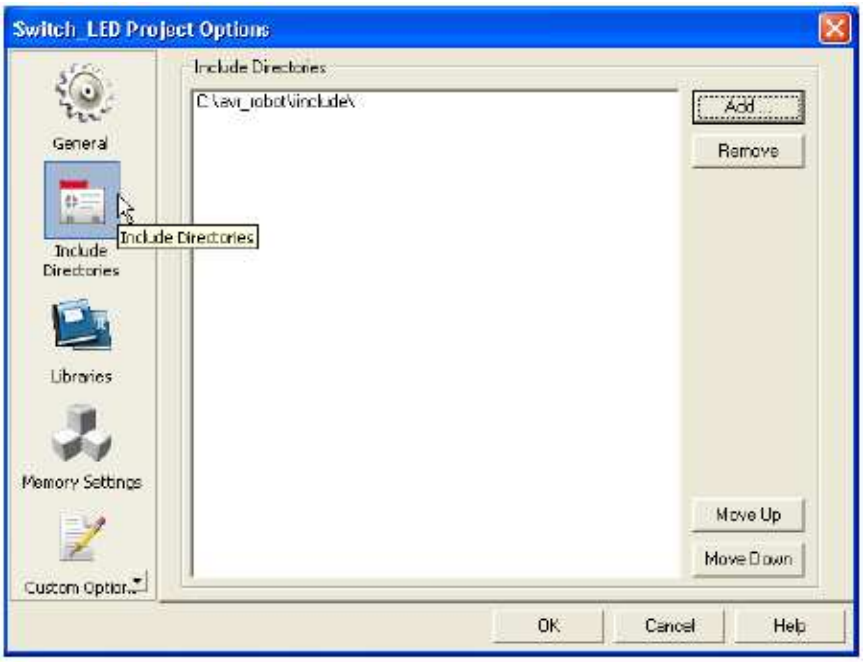


Figura 41-Alegerea fişierelor header

În căsuța Available Link Objects se selectează itemul lim.a care apare la fereastra Link with These Objects.Se apasă butonul OK pentru terminare

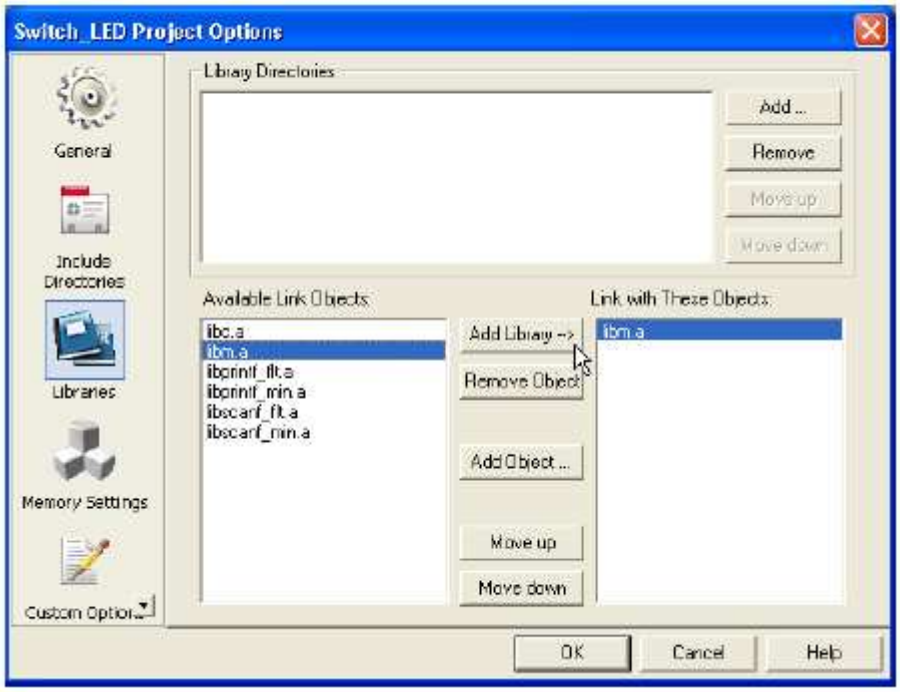


Figura 42-Adăugarea de librării

Se scrie programul în fereastra principală şi se compilează la menu Build opţiunea Build sau se apasă tasta F7.Dacă programul conţine erori ne va apărea fereastra BUILD OUTPUT care conţine numărul de erori şi avertismente.Se repară erorile şi mai apoi se compilează prin comanda TOOL →AVR Prog…

Este necesar ca pentru rularea programului robotul sa fie legat la pc prin cablul serial PX- 400 şi comutatorul power să fie activ.

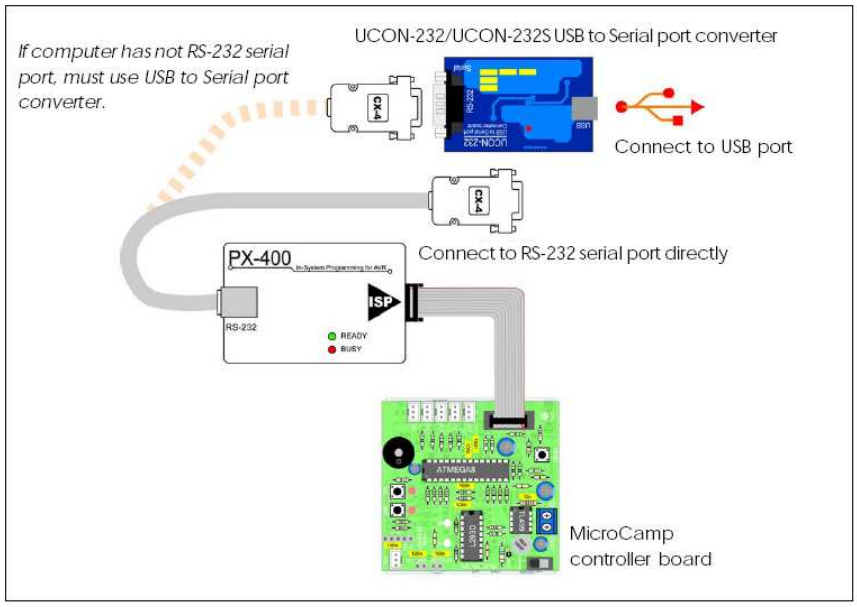


Figura 43-Modul de conectare la PC

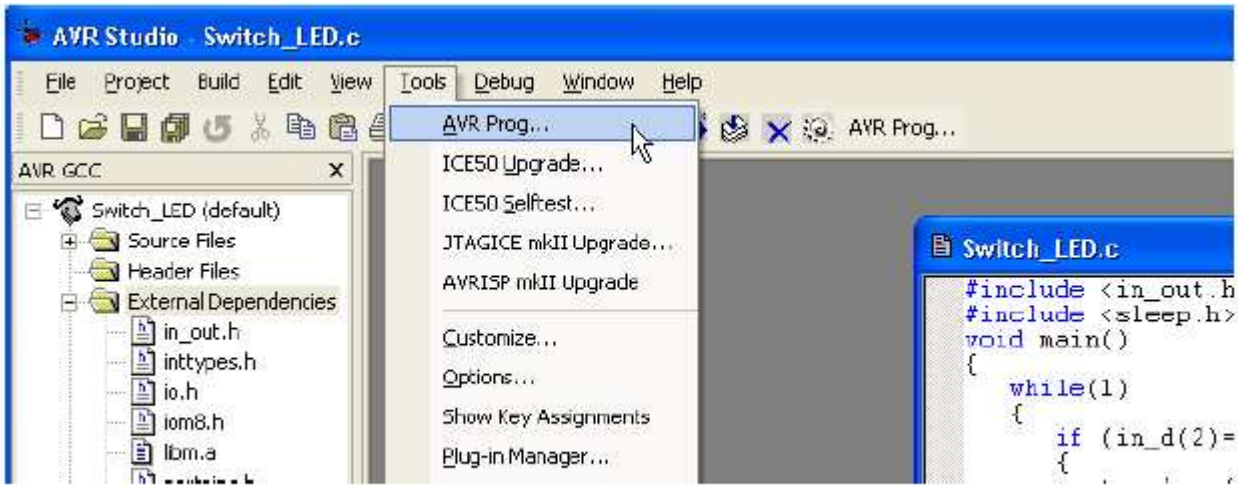


Figura 44-Alegerea debug-ului

Fişierul care iniţial era format .gcc se compilează în format hex.Acest fişier conţine codul convertit din C în cel al robotului hex.Este introdus ulterior în memoria microcontroler-ului prin apăsarea butonului Program

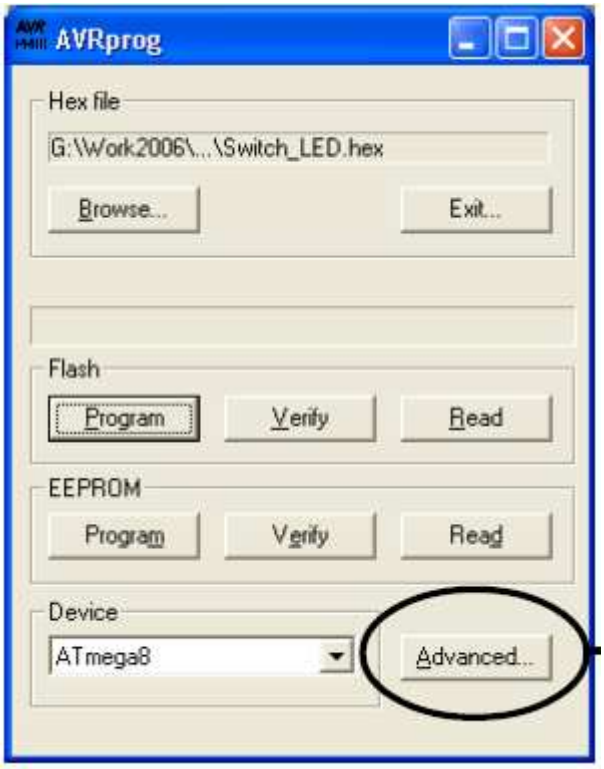


Figura 45-Programarea microcontrolerului.

Programarea microcontrolerului folosind limbajul de programare C

Limbajul C este folosit în programarea roboților alături de alte limbaje precum Pascal,Basic,LISP etc.

Comenzile în C sunt executate de către microcontroler pas cu pas de la începutul programului până la sfârşit iar la sfârşitul fiecărei instrucțiunii se poziționează“;”.

Cuvinte rezervate pentru limbaj: Tabel 11

Auto	Default	Float	Long	Sizeof	Case	Typedef	
Break	Do	For	Register	Static	Double	While	d
Goto	Return	Struct	Void	Char	Else	Continue	
Short	Switch	Volatile	Const	Enum	Int	Extern	

Tipuri de date folosite în limbajul C pentru programarea robotului care se încadrează în următoarele valori:
Tabel 13

Tabel 12

Tip	Valori
Char	-128…+127
unsigned	0…255

char	
Int	-32768…+32767
unsigned int	0…+4294967295
float	Orice număr real
long	2^{-31}….2^{31}-1

Operator	Explicație
==	Comparație pentru “egalitate”
!=	Comparație pentru “diferit”
<	Comparație pentru “mai puțin”
>	Comparație pentru “mai mult”
<=	Comparație pentru “mai puțin sau egal”
>=	Comparație pentru “mai mult sau egal”
!	NU
&&	ŞI
||	SAU

Sistemul numeric în C

1 Numere zecimale

2 Numere binare de tip 0Bbbbbb. B ia valori de 0 sau 1

De exemplu numărul binar 0b10010010=146

Modul de calcul:$(1 \times 2^7)+ :(0 \times 2^6)+ (0 \times 2^5)+ (1 \times 2^4)+ (0 \times 2^3)+ (0 \times 2^2)+ (1 \times 2^1)+(0 \times 2^0)=146_{10;}$

3 Numere hexazecimale de forma $0^{\times}FF=255$

Modul de calcul:$(15 \times 16^1)+ :(15 \times 16^0)=255_{10}$; $0^{\times}FF=0b1111111$ ca număr binar.

Set comenzi robot

Ver 1.0

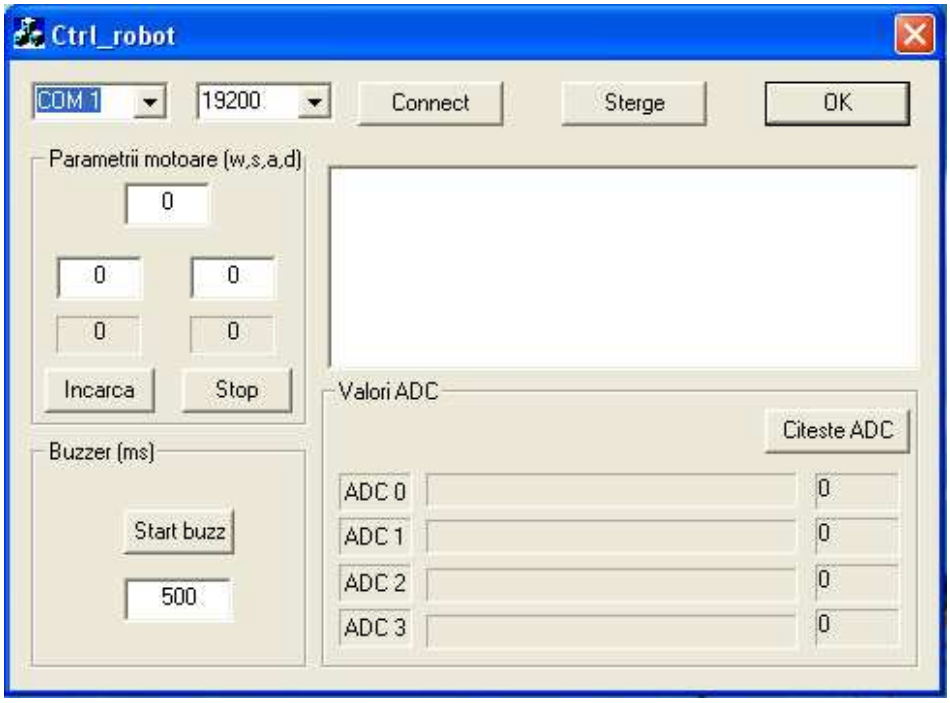


Figura 40-Fereastra Ctrl-robot

Elemente constituente

- Motor stânga, dreapta;
- Convertor analog digital (ADC);
- Buzzer;
- Comunicație serială;

Comenzile implementate trebuie să: comande motoarele(ambele intr-o comandă), trimite comanda buzzer(timpul programabil cat timp v-a fi activ), citeasca valorile ADC pentru 4 canale.

Comenzile vor avea o structura urmatoare:
<cmd><camp0><camp1> …. <campn><sum_ct_>

Semnificatia octeților:

- cmd – identificator comanda (intervalul 0:127 = 0x00:0x7F);
- camp0...campn – date mesaj, particularizate pentru fiecare comanda în parte;
- sum_ct – suma control, xor la toti octeții recepționati în comanda curentă.

Decalarea între mesaje se face în funcție de timpul dintre caractere. Astfel dacă timp de 5 ms nu am primit nici un caracter nou consider transmisia terminată.

Raspunsul trimis de placa v-a fi trimis la intervalul de 7 ms de la receptia ultimului caracter, (2ms) de la declararea receptiei.

Răspunsul la comandă poate fi: comanda acceptată (sau date cerute), nici un raspuns (pentru comanda cu suma de control eronată, sau lungime mesaj invalidă), comanda inexistenta (daca comanda nu este implementeta), parametrii invalizi (datele

atasate comenzii sunt în afara domeniului de acoperire). Erorile se semnalizează prin codul comenzii căreia i se face face sau (or) cu o masca 0x80. Astfel eroarea comenzii 0x01, va avea valoarea 0x01 | 0x80 = 0x81.

Forma raspunsului pentru comenzile respinse:
<cmd | 0x80><reason>

Unde reason:

1	-	Comanda neimplementata
2	-	Parametrii in afara limitelor

Comenzi terminal

Seteaza directie/viteza motoare (0x01)

Comanda:

0x01	sens_mt	pwr_mt_1	pwr_mt_2
1 byte	1 byte	1 byte	1 byte

sens_mt	-	Sensul de rotire al motorului 1 si 2 .
pwr_mt_1	-	indica puterea motorului 1 (factor de umplere PWM) valoarea cea mai semnificativa fiind trimisa prima (High).
pwr_mt_2	-	2 octeti ,indica puterea motorului 2 (factor de umplere PWM) valoarea cea mai semnificativa fiind trimisa prima (High).

Campul sens_mt:

mt_1		mt_2		x	x	x	x
Bi t 7	Bi t 6	Bi t 5	Bi t 4	Bi t 3	Bi t 2	Bi t 1	Bit 0

Semnificatie mt_1/ mt_2:

bit	bit 1	bit 2	
	0	0	Motor oprit
	0	1	Sens rotire fata
	1	0	Sens rotire inapoi
	1	1	Rezervat – nefolosit

Raspuns:

Raspuns corect : 0x02

Comada buzzer (0x02)

Comanda:

0x02	buzz_tm_H	buzz_tm_L
1 byte	1 byte	1 byte

Comanda porneste buzzerul si trimite nr de mesaj cât timp buzzerul v-a funcționa incepand de la momentul recepționarii comenzii.

buzz_tm_H:buzz_tm_L reprezinta timpul de functionare buzzer exprimat in ms. Durata ia valori intre 1 si 2500ms

Raspuns:

Citire canale analogice (0x03)

Comanda:

0x03
1 byte

Comanda este formată doar din codul comenzii , fără parametrii

Raspuns:

Răspunsul cuprinde patru câmpuri de date cu valorile citite de la cele 4 canale ale ADC-ului. Câmpurile adc_0, adc_1, adc_2, adc_3 iau valori în intervalul 0:255, corespunzător tensiunilor masurate 0:Vref.

0x03	adc_0	adc_1	adc_2	adc_3
1 byte	1 byte	1 byte	1 byte	1 byte

Identificare dispozitiv (0x7F)

Comanda:

0x7F
1 byte

Răspuns:

0x03	ID_dev	Ver_X	Ver_Y
1 byte	1 byte	1 byte	1 byte

ID_dev – identificator dispozitiv 0xAA.

Versinea curentă program controller Ver X.Y .

Programul comandă robotul de la tastaură prin cablul serial cu un conector RS-232 la calculator şi un conector pentru interfața serială la celălalt capăt.iar prin apăsare switch 1se schimba comanda…robotul citind datele digitale de la senzorul infraroşu Sharp.Programul este construit în CAVR.

```
#include "mega8.h"
#include "serial.h"
#include "motor.h"
#include "LCD_control.h"
#include "timer.h"
#include "analogic.h"
#include "buzzer.h"
#include "in_out.h"
#include "compiller_specific.h"

#define GLOBAL_MAIN
 #include  "main.h"


stop_watch_2Btime_main;

void main(void)
```

```
{
 init_serial();
 init_timer();
 init_motor();
 init_analogic();
 init_buzz();
 init_in_out();

  #asm("sei");

stop_watch2B_set(&time_main, 1000);
run_mode = RUN_WITH_PC;

 start_buzz(100);                // sa bazie si el 100 ms la start

 while(1)
 {    analogic_task();
  motor_task();
  buzz_task();
          in_out_task();

  if(run_mode == RUN_WITH_PC)
  {
          if( input_up & 0x01 )
          {       input_up &= ~0x01;
                  run_mode = RUN_ALONE;
        LED4_ON();
          } else
                  serial_task();
```

```
  } else
  if(run_mode == RUN_ALONE)
  {
          if( input_up & 0x01 )
          {       input_up &= ~0x01;
                  run_mode = RUN_WITH_PC;
                  LED4_OFF();
                  clear_LCD();
          } else
          {
                  test_LCD();
          }
  }

  if(stop_watch2B_tst(&time_main) == TRUE)    // periodic la 1 sec
  { stop_watch2B_set(&time_main, 1000);


  }

 } // end while(1)

}

#include "mega8.h"
#include "compiller_specific.h"

#define GLOBAL_ANALOGIC
 #include "analogic.h"
```

```
void init_analogic(void)
{
// -------- hardware layer -----------
  ADMUX = 0x40;              // referinta externa cu condensator la Aref,ADCL-full ADCH-2biti, mux ADC0
// ADCSRA = 0x9D;   // activez ADC, +(1<<ADIE) activez intreruperea,frecv ADC = Xtal/32
// MCUCR = 0x10;     // DAC noise reduction

  ADCSRA = 0x97;             // activez ADCS,frecv ADC = Xtal/32
  MCUCR = 0x00;              // idle

  ADCSRA |= 0x40;            // pornesc o conversie

// ------ application layer ----------
  adc_ch = 0x00;
  adc_val[0] = 0;
  adc_val[1] = 0;
  adc_val[2] = 0;
  adc_val[3] = 0;
}


/* ********************************************************** *
  citesc pe rand canalele ADC de la 0 la (CH_NR_MAX-1)
  valorile obtinute sunt mediate pt fiecare canal si facute disponibile
  astfel programul citeste asincron val ADC-ului
 * ********************************************************** */
```

```
void analogic_task(void)
{ unsigned int adc_read_val;            //
  unsigned char sum_idx;                //

  if(!(ADCSRA & 0x40))                                          // sunt in
timpul conversiei??
  {     // am un nou rezultat de la ADC
            w_low( adc_filter[adc_ch][adc_filter_idx] ) = ADCL;
        w_high( adc_filter[adc_ch][adc_filter_idx] ) = ADCH;

        sum_idx = adc_filter_idx;       // indexul folosit pentru a face suma
citirilor pt canalul curent
        adc_read_val = 0;               // aici fac suma pentru canalul curent

        do
        {
         // insumez toate ultimele 16 citiri (fac mediere)
         sum_idx++;
         sum_idx &= CH_FILTER_NR_MSK;                                   //
index modulo 16
         adc_read_val += adc_filter[adc_ch][sum_idx];
        }while(sum_idx != adc_filter_idx);

        adc_read_val /= CH_FILTER_NR*4;             // fac media cu 16,
                                        // si mai impart la 4 pentru a aduce
rezultatele pe 8 biti

        adc_val[adc_ch] = w_low(adc_read_val);
```

```
            adc_ch++;
            if(adc_ch == CH_NR_MAX)
             {      // final ciclul, reiau ciclul de masurat canalele ADC
                    adc_ch = 0;
                    adc_filter_idx++;
                    adc_filter_idx &= CH_FILTER_NR_MSK;                    //
index modulo 16
             }

            ADMUX &= ~0x07;
            ADMUX |= adc_ch;          // schimb canalul ADC
            ADCSRA |= 0x40;                   // pornesc o noua conversie
        }
    }

    #include "timer.h"
    #include "compiller_specific.h"

    #define INTERN_BUZZ
    #define GLOBAL_BUZZ
     #include "buzzer.h"


    void init_buzz(void)
    {
      PORTD.4 = 0;                    // trec in zero portul
      DDRD.4 = 1;                     //        trec portul pe iesire
    }
```

```
void start_buzz(unsigned int time)
{
   stop_watch2B_set(&buzz_task_desc.time, time);        // timpul de start
   buzz_task_desc.state = BZ_ON;
}

/* ********************************************************** *
  testez valoarea receptionata pt buzzer
  ret: TRUE val corecta, FALSE val eronata
 * ********************************************************** */
unsigned char set_buzzer_str(unsigned char *buff)
{
  unsigned int buzz_time;

  w_high(buzz_time) = *(buff++);
  w_low(buzz_time) =  *buff;          // copiez datele din doi octeti intr-un intreg

  if(buzz_time <=2500)
  {//  start_buzz(buzz_time);
     stop_watch2B_set(&buzz_task_desc.time, buzz_time);       // timpul de start
     buzz_task_desc.state = BZ_ON;
          return TRUE;          // buzzer pornit cu succes
  }
          return FALSE;
}


void buzz_task(void)
```

```
{
  if( buzz_task_desc.state == BZ_ON)
  {       if(stop_watch2B_tst(&buzz_task_desc.time) == TRUE)
          {       buzz_task_desc.state = BZ_OFF;
          }
   } else
  if( buzz_task_desc.state == BZ_OFF)
  {       // ma asigut ca buzz nu este alimentat
          PORTD.4 = 0;                    // chiar daca are condensator, in cazul
curent
  }
}
#include "mega8.h"
#include "compiller_specific.h"

#define GLOBAL_IN_OUT
 #include  "in_out.h"

void init_in_out()
{
  DDRD &= ~0x0C;          // trec portul D corespunzator tastelor pe intrare
  DDRC |= 0x20;           // trec ledul pe iesire

  PORTD &= ~0x0C;         // pull-up-urile sunt oprite
  PORTC &= ~0x20;         // ma asigur ca ledul este stins, in prima faza
}

void in_out_task()
{   unsigned char input_temp;
```

```
            input_old = input;
            input = ~((PIND & 0x06)>>2);
            // =>   tasta 2 apasata bitul b0 == 1
            //      tasta 3 apasata bitul b1 == 1

            input_temp = input ^ input_old;         // observ variatiile intre starea curenta
si cea trecuta

            if(input_temp)
            { // daca au avut loc variatii
              input_up |= (input_temp & input);        // starea curenta a bitilor este 1, si
au avut loc variatii
              input_dn |= (input_temp & (~input));     // starea curenta a bitilor este 0, si
au avut loc variatii
            }
      }


      #include "serial.h"
      #include "timer.h"
      #include "analogic.h"
      #include "compiller_specific.h"

      #define GLOBAL_CLD_CTRL
       #include "LCD_control.h"

      unsigned char flash nibble_2_hex[] = "0123456789ABCDEF";
```

```
static stop_watch_2B time_send_msg;                // static aloc variabila pe toata durata rularii programului

void clear_LCD()
{
        TX0_buff_count_end = 0;

        TX0Buffer[TX0_buff_count_end++] = 254;
      TX0Buffer[TX0_buff_count_end++] = 0x01; // clear LCD

        bTX0_send = bTRUE;
        bTX0_start = bTRUE;
       UCSRB |= 1<<5;                // TX int (la TSR liber)

       stop_watch2B_set(&time_send_msg, 300);   // peste 300 ms o sa mai trimit mesaj pe seriala
}

void test_LCD()
{
  unsigned long analogic_val;

  if(stop_watch2B_tst(&time_send_msg) == TRUE)                    // periodic la 1 sec
  { stop_watch2B_set(&time_send_msg, 300);

                TX0_buff_count_end = 0;

                TX0Buffer[TX0_buff_count_end++] = 254;
```

```
                TX0Buffer[TX0_buff_count_end++] = 0x80;
                TX0Buffer[TX0_buff_count_end++] = 'T';
                TX0Buffer[TX0_buff_count_end++] = 'e';
                TX0Buffer[TX0_buff_count_end++] = 's';
                TX0Buffer[TX0_buff_count_end++] = 't';
                TX0Buffer[TX0_buff_count_end++] = ' ';

                TX0Buffer[TX0_buff_count_end++] = 'L';
                TX0Buffer[TX0_buff_count_end++] = 'C';
                TX0Buffer[TX0_buff_count_end++] = 'D';

                        TX0Buffer[TX0_buff_count_end++] = 254;
                TX0Buffer[TX0_buff_count_end++] = 0x80 | 0x39;

                TX0Buffer[TX0_buff_count_end++] = 'A';
               TX0Buffer[TX0_buff_count_end++] = 'D';
               TX0Buffer[TX0_buff_count_end++] = 'C';
               TX0Buffer[TX0_buff_count_end++] = '0';
               TX0Buffer[TX0_buff_count_end++] = ' ';
               TX0Buffer[TX0_buff_count_end++] = '0';
               TX0Buffer[TX0_buff_count_end++] = 'x';
               TX0Buffer[TX0_buff_count_end++] = nibble_2_hex[ (adc_val[0]>>4 )
& 0x0F];       // nibblul cel mai semnificativ
               TX0Buffer[TX0_buff_count_end++] = nibble_2_hex[ adc_val[0] & 0x0F
];       // nibblul cel de jos

                        analogic_val = (unsigned long)((unsigned long)adc_val[0]*100);
        // vreau sa prind zecimale
                        analogic_val *= 5;
```

```
                    analogic_val /= 256;

            TX0Buffer[TX0_buff_count_end++] = ' ';
            TX0Buffer[TX0_buff_count_end++] = ' ';
            TX0Buffer[TX0_buff_count_end++] = (unsigned
char)(analogic_val/100) + '0';      // 1V
            TX0Buffer[TX0_buff_count_end++] = '.';
            analogic_val %= 100;                  // calculez restul la impartirea cu 100
            TX0Buffer[TX0_buff_count_end++] = (unsigned char)(analogic_val/10)
+ '0';  // 0.1V
            analogic_val %= 10;
            TX0Buffer[TX0_buff_count_end++] = (unsigned
char)(analogic_val)+'0';
            TX0Buffer[TX0_buff_count_end++] = 'V';

                    bTX0_send = bTRUE;
                    bTX0_start = bTRUE;
             UCSRB |= 1<<5;                  // TX int (la TSR liber)
      }
    }
    #include "mega8.h"
    #include "platform_specific.h"

    #define INTERN_MOTOR
    #define GLOBAL_MOTOR
     #include "motor.h"



    #define MOTOR_INITIAL 100
```

```
/* ********************************************************** *
  initializare periferice driver motor
 * ********************************************************** */
void init_motor(void)
{
    DDRB |=  0x07;              // iesirile PWM si setare sens PB0 trecute pe
out
  DDRD |=  0xE0;                // setare sens PD5:PD7 trecute pe out

// WGM13:WGM10 = 0x05;                      - Fast PWM, 8-bit
// COM1A1:COM1A0, COM1B1:COM1B0 = 0x00          - dezactivare
canal pwm
//                                          = 0x02          - canal
pwm activ
// CS12:CS10 = 0x05                             - prescaler 1024
// Fpwm = Fosc/(2*N*TOP) = 19.14Hz      => Tpwm = 0.0522s

  TCCR1A = 0xA0 | 0x01;       //  COM1A1,COM1B1, WGM10
  TCCR1B = 0x08 | 0x05;       //   WGM12, CS11, CS10          // prescaller
1024

  OCR1AL = MOTOR_INITIAL;
  OCR1BL = MOTOR_INITIAL;           // sa nu fie zero
  engine_control.pwm_val_left = MOTOR_INITIAL;
  engine_control.pwm_val_right = MOTOR_INITIAL;

  TIFR |= 0x04;                     // sterg fanion intrerupere TOV1
  TIMSK |= 0x04;                    // activare int timer1 TOIE1
```

```
}


/* ************************************************************ *
  initializare periferice driver motor
 * ************************************************************ */
void motor_task(void)
{
  if(engine_control.mode_left == MT_STOP)
  {
        TCCR1A &= ~0x80;
  }else
  if(engine_control.mode_left == MT_FRW)
  {
        TCCR1A |= 0x80;
        MT_CH1_FRW();
  }else
  if(engine_control.mode_left == MT_REW)
  {
        TCCR1A |= 0x80;
        MT_CH1_REW();
  }

  //----------------- al doilea motor --------------
  if(engine_control.mode_right == MT_STOP)
  {
        TCCR1A &= ~0x20;
  }else
  if(engine_control.mode_right == MT_FRW)
```

```
  {
        TCCR1A |= 0x20;
        MT_CH2_FRW();
  }else
  if(engine_control.mode_right == MT_REW)
  {
        TCCR1A |= 0x20;
        MT_CH2_REW();
  }
}


interrupt [TIM1_OVF] void pwm1_handle(void)  // int 52.2 ms
{
  OCR1AL = engine_control.pwm_val_left;
  OCR1BL = engine_control.pwm_val_right;
}



/* ********************************************************** *
  incarc parametrii primiti pe seriala
 * ********************************************************** */
void set_motor_str( unsigned char *buff)
{
   unsigned char temp_var;

   temp_var = *(buff++);
```

```
TIMSK &= ~0x04;                    // dezactivare int timer1 TOIE1, atata timp cat modific parametrii de functionare


engine_control.pwm_val_left       = *(buff++);
if(engine_control.pwm_val_left == 0)
{       // PWM receptionat este zero, opresc motorul
         engine_control.mode_left = MT_STOP;
         engine_control.pwm_val_left = 1;
} else
if( (temp_var & (0x03)<<6) == (0x01<<6) )
{       // rotire fata
   engine_control.mode_left = MT_FRW;
} else
if( (temp_var & (0x03)<<6) == (0x02<<6) )
{       // rotire spate
   engine_control.mode_left = MT_REW;
} else
{       // motor oprit
   engine_control.mode_left = MT_STOP;
}


engine_control.pwm_val_right      = *buff;
         if(engine_control.pwm_val_right == 0)
{       // PWM receptionat este zero, opresc motorul
         engine_control.mode_right = MT_STOP;
         engine_control.pwm_val_right = 1;
} else
```

```
  if( (temp_var & (0x03)<<4) == (0x01<<4) )
  {       // rotire fata
     engine_control.mode_right = MT_FRW;
  } else
  if( (temp_var & (0x03)<<4) == (0x02<<4) )
  {       // rotire spate
     engine_control.mode_right = MT_REW;
  } else
  {       // motor oprit
     engine_control.mode_right = MT_STOP;
  }


  TIMSK |= 0x04;                          // activare int timer1 TOIE1

}
#include "mega8.h"
#include "compiller_specific.h"

#define GLOBAL_SERIAL
 #include  "serial.h"

#include "main.h"       // pt ID placa, versine program
#include "analogic.h"   // pt a citi canalele ADC
#include "buzzer.h"
#include "motor.h"

#define XTAL 160000
#define BAUD 96
```

```
/* ********************************************************** *
  initializare comunicatie seriala
 * ********************************************************** */
void init_serial()
{
   UBRRH = (unsigned char) ((XTAL/(16*BAUD)-1)>>8);
   UBRRL = (unsigned char) (XTAL/(16*BAUD)-1);
   UCSRA = 0x00;                    // nu prea are sens initializarea asta
   UCSRC = 0b10000110;              // asyncron, parity disable, 1 bit stop,
8 biti
   UCSRB = 0b10011000;              // RX_int-en, RX-en, TX-en

   RX0_buff_count = 0;
   TX0_buff_count = 0;

   RX_TX_time = SER_TIME_SEND_MSG;                    // evit
preluarea unui msg fals la initializare

   bTX0_send = bFALSE;
   bRX0_done = bFALSE;
  bTX0_start = bFALSE;
}


// ----------------------------------------------------------
//            intreruperi comunicatie seriala
// ----------------------------------------------------------
interrupt [USART_RXC] void RX_handle(void)          // RX
```

```
{    RX0Buffer[RX0_buff_count] = UDR;
   if(RX0_buff_count < (BUFF_MAX-1))
   { RX0_buff_count++;
     RX_TX_time = 0;
   }
}

interrupt [USART_DRE] void TX_handle_1(void)         // TX
  USART_TXC
{   UDR = TX0Buffer[TX0_buff_count];
   TX0_buff_count++;

   if(TX0_buff_count == TX0_buff_count_end)
   { UCSRB &= ~(1<<5);     // disable int, nu mai incarc cu date
    bTX0_send = bFALSE;      // pot sa compun un alt mesaj
    TX0_buff_count = 0;
   }
}



/* ************************************************************ *
  task comunicatie seriala
 * ************************************************************ */
void serial_task()
{
  if(bRX0_done == bTRUE)
  {       bRX0_done = bFALSE;
          switch ( RX0Buffer[0] )
          {       case SET_MOTORS:
```

```
                    if(RX0_buff_count_end == SET_MOTORS_SZ )
                    {  if(TRUE == sum_check_tst(RX0Buffer,
RX0_buff_count_end))
                                {
                                        if( RX0Buffer[1] & 0x0F)
                                        {   // am primit niste biti rataciti prin octetul de sens
                                                TX0_buff_count_end = 0;
                                                TX0Buffer[TX0_buff_count_end++] =
SET_MOTORS | 0x80;              // eroare
                                                TX0Buffer[TX0_buff_count_end++] =
0x02;
                                        } else
                                        {   // mesaj corecet setez motorul si trimit raspunsul
                                                set_motor_str( &RX0Buffer[1]);

                                                TX0Buffer[0] = SET_MOTORS;
                                        TX0_buff_count_end = 1;
                                        }

                                        sum_check_add(TX0Buffer,
TX0_buff_count_end);
                                        TX0_buff_count_end ++;
                                        bTX0_send = bTRUE;
                                }
                    }
                    break;

                    case SET_BUZZ:
            if(RX0_buff_count_end == SET_BUZZ_SZ )
```

```
                    {  if(TRUE == sum_check_tst(RX0Buffer,
RX0_buff_count_end))
                        {
                            if( TRUE == set_buzzer_str(&RX0Buffer[1]) )
                            {   // buzzer pornit, trimit comanda reusita
                                TX0Buffer[0] = SET_BUZZ;
                                TX0_buff_count_end = 1;
                            } else
                            {       // timpul setat in afara domeniului asteptat
                                TX0_buff_count_end = 0;
                                TX0Buffer[TX0_buff_count_end++] =
SET_BUZZ | 0x80;          // eroare
                                TX0Buffer[TX0_buff_count_end++] =
0x02;
                            }
                                sum_check_add(TX0Buffer,
TX0_buff_count_end);
                                TX0_buff_count_end ++;
                                bTX0_send = bTRUE;
                        }
                    }
                    break;

                    case GET_ADC:
            if(RX0_buff_count_end == GET_ADC_SZ )
                    {  if(TRUE == sum_check_tst(RX0Buffer,
RX0_buff_count_end))
                        {       TX0_buff_count_end = 0;
                                TX0Buffer[TX0_buff_count_end++] = GET_ADC;
```

```
                                TX0Buffer[TX0_buff_count_end++] = adc_val[0];
                                TX0Buffer[TX0_buff_count_end++] = adc_val[1];
                                TX0Buffer[TX0_buff_count_end++] = adc_val[2];
                                TX0Buffer[TX0_buff_count_end++] = adc_val[3];
                                sum_check_add(TX0Buffer,
TX0_buff_count_end);
                                TX0_buff_count_end ++;
                                bTX0_send = bTRUE;
                        }
                }
                break;

                case GET_ID:
        if(RX0_buff_count_end == GET_ID_SZ )
                {  if(TRUE == sum_check_tst(RX0Buffer,
RX0_buff_count_end))
                        {       TX0_buff_count_end = 0;
                                TX0Buffer[TX0_buff_count_end++] = GET_ID;
                                TX0Buffer[TX0_buff_count_end++] = DEV_ID;
                                TX0Buffer[TX0_buff_count_end++] = VER_X;
                                TX0Buffer[TX0_buff_count_end++] = VER_Y;
                                sum_check_add(TX0Buffer,
TX0_buff_count_end);
                                TX0_buff_count_end ++;
                                bTX0_send = bTRUE;
                        }
                }
                break;
```

```
                default:
                if(TRUE == sum_check_tst(RX0Buffer, RX0_buff_count_end))
                {       // comanda necunoscuta dar cu suma control corecta
                        TX0_buff_count_end = 0;
                        TX0Buffer[TX0_buff_count_end++] = RX0Buffer[0] |
0x80;
                    TX0Buffer[TX0_buff_count_end++] = 0x01;
        // comanda necunoscuta
                        sum_check_add(TX0Buffer, TX0_buff_count_end);
                        TX0_buff_count_end ++;
                        bTX0_send = bTRUE;
                }
            }  // end switch
        }       // end


        if( bTX0_send == bTRUE && bTX0_start == bTRUE)
        {       // am mesaj de trimis, si sunt in fereastra de timp in care pot trimite
           bTX0_start = bFALSE;

                UCSRB |= 1<<5;                  // TX int (la TSR liber)
        }
      }


     /* ************************************************************ *
       testez datele dintr-un buffer sa vad daca au suma control corecta
       ret: TRUE suma corecta, FALSE suma eronata
      * ************************************************************ */
```

```
unsigned char sum_check_tst(unsigned char *buff, unsigned char length)
{ unsigned char sum;
  sum = 0;

  do
  {
   sum ^= *(buff++);
   length--;
  }
  while(length!=0);

  if(sum == 0)
  return TRUE;
  return FALSE;
}


/* ************************************************************ *
  adaug suma de control unui buffer
  lungimea  pachetului de date trebuie sa fie de minimum 1
 * ************************************************************ */
void sum_check_add(unsigned char *buff, unsigned char length)
{
  unsigned char sum;
  sum = 0;

  do
  {
   sum ^= *(buff++);
```

```
   length--;
  }
  while(length!=0);

  *buff = sum;            // adug suma de control
}



#include "mega8.h"
#include "compiller_specific.h"

#define GLOBAL_TIMER
 #include "timer.h"

#define  INTERN_BUZZ
#include "buzzer.h"

#include "serial.h"

static unsigned int timer_val;
static unsigned int timer_val_bk;

/* ********************************************************** *
  initializare timer2 folosit pentru masurarea timpului
      intarzieri etc.
 * ********************************************************** */
void init_timer(void)
{
  // XTAL = 16Mhz
```

```
  TCCR2 = 0x0C;       // CTC, no ports, prescaler 64 - WGM21, CS22
  OCR2 = 250;         // XATL/64/250 = 1000 -> 1ms
  TIMSK = 0x80;       // intrerupere cand avem match regs
}


// -----------------------------------------------------------
//             intrerupere timer 2
// -----------------------------------------------------------
interrupt [TIM2_COMP] void timer_2 (void)       // 1 ms
{
  timer_val ++;

  if(buzz_task_desc.state == BZ_ON)
  {       PORTD.4 = ~PORTD.4;
  }

// ---------- responsabil cu mng serialei ---------------------
  if(RX_TX_time < SER_TIME_SEND_MSG)
  {
          RX_TX_time++;
          if(RX_TX_time == SER_TIME_END_MSG)
          {       // am mesaj receptionat nou
                  RX0_buff_count_end = RX0_buff_count;
                  RX0_buff_count = 0;
                  bRX0_done   = bTRUE;
          } else
          if(RX_TX_time == SER_TIME_SEND_MSG)
          {       // pot sa trimit raspunsul
```

```
                bTX0_start = bTRUE;
            }
  }
// ---------------- end seriala ------------------------------
}


/* ************************************************************ *
  timpul de asteptat se seteaza la initializare timer
 * ************************************************************ */
/* ************************************************************ *
  incarc momentul de inceput cronometrare si timpul dorit pt cronometrare
 * ************************************************************ */
void stop_watch2B_set(stop_watch_2B *time_struct, unsigned int time)
{
  time_struct->time_length = time;
  TMR1_INT_DIS();           // disable int
  time_struct->time_start = timer_val;
  TMR1_INT_EN();
}

/* ************************************************************ *
  verific daca a trecut timpul asteptat
  intorc  TRUE  - perioada cronometrata a trecut
                FALSE - mai astept perioada nu a trecut
 * ************************************************************ */
unsigned char stop_watch2B_tst(stop_watch_2B *time_struct)
{
  TMR1_INT_DIS();
```

```
    timer_val_bk = timer_val;
    TMR1_INT_EN();
    if(time_struct->time_length < (unsigned int)(timer_val_bk - time_struct-
>time_start))
     return TRUE;
     return FALSE;
}
```

Programul compilat în AVR Studio pentru detectarea şi ocolirea obstacolelor folosind senzorul infraroşu este arătat mai jos.Robotul se deplasează iar dacă senzorul infraroşu detectează un obstacol la o distanţa de 8 cm se întoarce la dreapta şi continuă drumul. Programul sursă este urmatorul:

```
include <stdlib.h>

// *Pentru convertire tipuri de date    *//

#include <motor.h>
  // *Controlul motoarelor*//

#include <sleep.h>
  //* Pentru întârziere*//

#include <sound.h>

//*Pentru semnal sonor*//

#include <analog.h>                                                    //

//* Convertor analog-digital*//

void main()
  //*Program principal*//

{
```

```
  unsigned int sensor=0;
  unsigned char i=0;
  sleep(200);    sound(4000,50);
//* inceput cu un beep*//
  while(1)
  {
        sensor=0;
        for (i=0;i<5;i++)
        { sensor=(sensor+analog(0)); }
        // *Citeşte datele de la GP2D12 de 5 ori*//
        sensor=(sensor/5);
/*Se face media aritmetică a distanțelor*//
        if (sensor>260)
        // * Condiție ca distanța să nu fie mai mare de 10 cm ?
        {
              backward(50);sleep(800);
        // *Schimbarea direcției  *//
              s_left(50);sleep(600);
        }
        else
```

```
            {
                forward(50);
                                //* Merge înainte*//
            }
        }
    }
```

BIBLIOGRAFIE

1. Gheorghe I Gheorghe, Valentin Pau, Doru Dumitru Palade, Mecatronica, Editura Cefin, Bucureşti, 2002.

2.Mircea Nițulescu,[1998], Roboți mobili, Editura SITECH Craiova.

3. Philippe Coiffet, La robotique, Principes et aplicațions.Editura Hermes Paris,1986.

3. V. Valcovici, Ştefan Bălan, R Voinea, Mecanica teoretică, Editura Tehnică Bucureşti,1986.

4. Gâlmeanu, C., Contribuții la sinteza roboților mobili cu 3 grade de mobilitate, Teză de doctorat, UPB, 2000.

5 .Mihăiță Ardeleanu, Roboți industriali mobili, Curs.

6.Valerica Cimpoca,Aparatura optoelectronică, Curs.

7.Catalog pentru componente optoelectronice,Băneasa SA,Bucureşt 2000.

8.Inițiere în limbajul C, Damian Costea.

9.Vasile M. Catunenu, Tehnologie electronică, Editura Didactică şi pedagogică, Bucureşti 1981

10.Lucian Ciobanu,Elemente de proiectare a sistemelor flexibile de fabricație şi a roboților industriali, Editura BIT,Iaşi 1997.

11 .www.microchip.com

12.www.atmega8.com

13.www. inex.co.th

14.www.inexglobal.com

15.www.maximum robotics

15.Numeroase alte site-uri aduse în fața ochilor mei de www.google.com

16 Note de curs şi laboratoare.