Cuprins

1. Introducere

Dorința omului de a învăța şi de a evolua cât mai rapid spre o ființă supremă Homo-Superior îl face pe acesta să descopere lucruri noi şi din ce în ce mai spectaculoase.În ultimi ani oamenii au făcut paşi importanți în dezvoltarea tehnologiei datorită apariției circuitelor integrate.Circuite ce azi se poate spune că stau la baza tehnologiei.

Roboții ocupă un spațiu larg în această lume, un procent mare dintre activitățile pe care le desfăşurau şi le desfăşoară oamenii sunt suplinite şi completate de aceştia.Astfel regăsim roboți în domeniul medical, în industrie, în domeniul militar.etc. Datorită vitezei de lucru, a preciziei, a siguranței pe care o oferă şi a randamentului ridicat oamenii aleg utilizarea cât mai deasă a roboților.

În general roboții sunt utilizați prin interfațarea lor cu un calculator sau alte echipamente de interfațare.

Lucrarea de față are ca scop parcurgerea cât mai multor noțiuni dintr-o gamă largă de obiecte cum ar fi electronica, robotica cât şi automatica.Pentru a realiza aceasta, am ales studierea unui robot, care se deplasează pe un traseu, dirijat de la distanță de către o persoană, ca apoi, comutând pe sistemul autonom, robotul să fie capabil să se întoarcă şi să parcurgă acelaşi traseu singur, ba chiar să fie în stare să traverseze alt traseu de dificultate diferită față de primul.

Un avantaj enorm față de alte sisteme asemănătoare este acela al interfeței neuronale: structură ce se doreşte a copia cât mai mult posibil modul de gândire al oamenilor şi a-l transpune în practică. Folosindu-se ca valori de antrenare semnalele senzorilor şi comenzile date de utilizator, aceasta memorează si apoi calculează traseul optim de parcurs, lucru ce duce la îmbunătățirea autonomiei robotului. Practic acest robot poate să parcurgă un traseu de 2-3 ori mai rapid față de alți roboți ce au caracteristici asemănătoare şi sunt utilizați în domenii similare.

Robotul reprezintă un ansamblu de blocuri mecanice-electrice, senzori, motoare şi module electrice de comandă pentru a asigura controlul acestuia. Se cunosc diferite tipuri de roboți, dar acesta face parte din categoria unităților mobile de explorare.

O astfel de unitate este ideală pentru explorarea mediilor nefavorabile omului.Robotul poate fi folosit uşor pentru găsirea unor oameni aflați prinşi sub dărâmaturi, în domeniul militar ajutând genişti, sau în diferite misiunii de salvare.

Pentru a creea acest robot sau integrat mai multe componente cum ar fi: microcontrollerul MCF5213, motoare de curent continuu, senzorii IR, transciever HC-06,Bloc

de alimentare, Bloc de comanda motoare, soft pentru microcontroller cât şi pentru comandă de la distanță, despre toate acestea se va vorbi în capitotlele următoare.

2. Fundamentare teoretică

2.1 Motoare de curent continuu

Motorul electroc este mecanism ce transforma energia electric în energie mecanică.Se cunosc două categorii mari de motoare, motoare de curent continuu şi motoare de curent alternative

Motoarele de curent continuu sunt destul de răspândite atât în aplicațiile de uz casnic: mixere, aspiratoare, roboți de bucătărie, maşini de spălat, diverse jucării cât şi în mediul industrial brațe robotice, linii de transmisie, automobile.

Utilizarea lor frecventă se datorează atât prețului care este redus cât şi datorită faptului că se pot comanda foarte uşor.În figura 1de mai jos se prezintă nişte motoare de curent continuu.

Figura.1 Motoare de curent continuu

Există pentru categorii de motoare de curent continuu clasificate în funcție de modul în care este conectată înfăşurarea de excitație :

- Motor cu excitație serie – înfăşurarea rotorică şi înfăşurarea statorică sunt conectate în serie
- Motor cu excitație paralelă – înfăşurarea statorică şi cea rotorică sunt conectate în paralel la aceeaşi sursă de tensiune
- Motor cu excitație mixtă – unde înfăşurarea statorică este compusă din două înfăşurări una conectată în serie şi cealaltă conectată în paralel

- Motor cu excitație independentă – unde atât înfăşurarea statorică cât şi cea rotorică sunt conectate la surse separate de tensiune

Se prezintă în figura 2 scheme echivalente ale celor pentru categorii de motoare

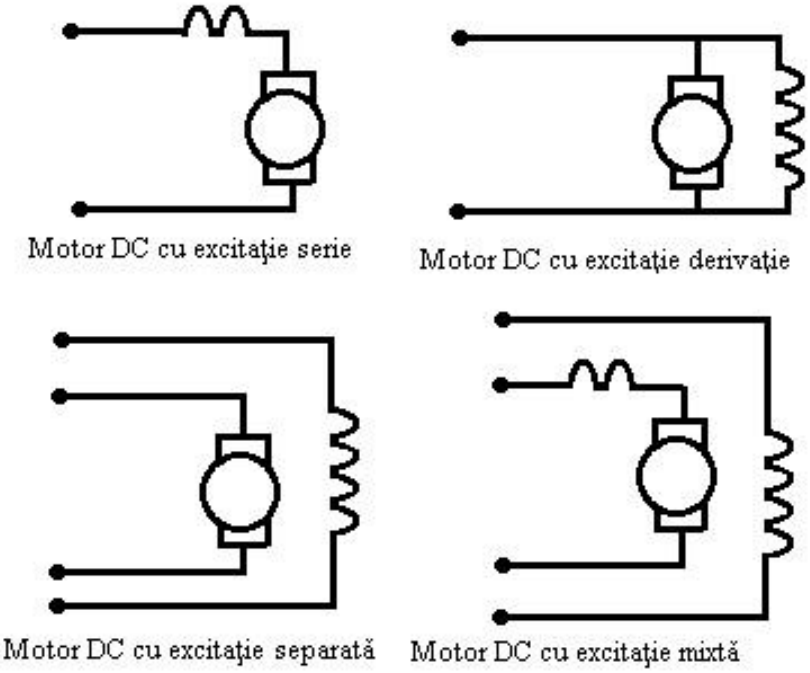


Figura.2 Modele de motoare de curent continuu

Motoarele de curent continuu sau motoarele DCcu magneți permanenți şi perii colectoare sunt confecționate din foarte puține materiale şi anume: un stator, un rotor ,comutator, periile colectoare.

Turația motorului creşte proporțional cu tensiunea aplicată pe înfăşurarea de pe rotor şi este invers proporțională cu câmpul magnetic de excitație.Funcție de variația tensiunii pe bobina de pe rotor variză şi turația motorului.

Cuplul dezvoltat de motor este proporțional cu curentul electric ce trece prin rotor şi cu câmpul magnetic de excitație.Reglând turația în funcție de camp se diminuează şi cumplul dezvoltat de motor.

Statorul creează un camp magnetic staționar care învelşte rotorul.Acest camp este generat fie cu ajutorul unor bobine (înfăşurări electromagnetice) ori cu ajutorul unor magneți permanenți.

Rotorul este compus dintr-un ax pe care se găsesc una sau mai multe înfăşurări. O dată ce acestea sunt alimentate ele produc un camp magnetic.(câmpul magnetic învârtitor).Apare

atracție magnetică între polii magneticii de pe stator şi cei de pe rotor aşa se generează rotația axului.

Prin intermediul periilor si al comutatorului sunt alimentate înfăşurările.Periile realizează o comutație mecanică ceea ce reprezintă un dezavantaj major datorită uzurii acestora în timp.

În figura 3 se prezintă principiul de funcționare al unui motor de curent continu.

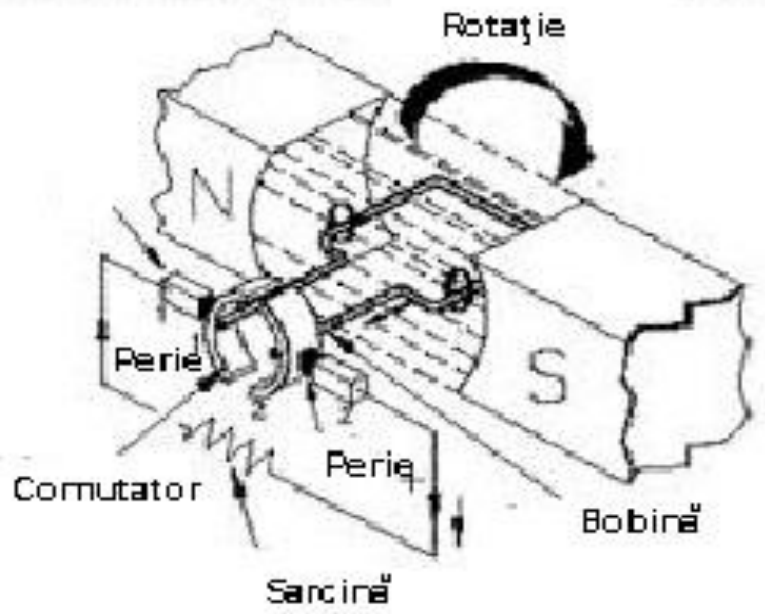


Figura.3 Principiul de functionare al unui motor DC

Apare sarcina asupra periilor care comandă bobina, aceasta crează impreună cu magneții permanenți de pe stator câmpul magnetic învârtiror, care roteşte axul.Comutatorul inversează sensul curentului la jumătatea fiecărei turi pentru a menține mişcarea de rotație uniformă în acelaşi sens.

Prin felul în care motorul DC este construit, el este foarte uşor de utilizat în diverse aplicații în care este necesar controlul vitezei respectiv variația acesteia.Prin urmare se ştie că viteza de rotație este direct proporțională cu tensiunea de alimentare.Pentru a varia tensiunea de alimentare foarte uşor se utilizează tehnica de modulație PWM (Pule Width Modulation).

2.2 Comanda motorului de curent continuu prin PWM (Pulse Width Modulation)

Metoda prin care am hotărât comanda motorului este prin generare de impulsuri PWM cu ajutorul unui microcontroller.Modulația este obținută prin utilizarea comutatorelor electronice de putere.

Valoarea medie a tensiunii si curentului este obținută prin inchidederea şi deschiderea comutatoarelor de putere intre acestea si alimentare la o frecventă foarte mare.Cu cât timpul în care comutatorul stă pe on este mai mare decât timpul în care comutatorul stă pe off, cu atât va creste puterea debitată pe sarcină.

Este necesară o frevență de comutație mult mai mare decât o frecvență care să afecteze sarcina (înfăşurările).Pentru a realiza cu succes reglajul unei lumini folosind modulația PWM, este necesară o frecvență de câțiva kiloherți, pentru controlul unui motor DC sunt necesari zeci de kiloherți, iar pentru a amplificatoarele audio sute de kiloherți.

Noțiunea de „ ciclu" (duty cycle) reprezintă raportul dintre frontul crescător al semnalului si frontul descrescător al acestuia.O valoare mică a ciclului corespunde unei puteri mici, acesta este măsurat în procente, 100% reprezintă că semnalul se află permanent în starea high.

Avanatajul major al al utilizării PWM-ului este că puterea risipită prin comutator când acesta este în starea low este aproximativ egală cu zero, iar cand acesta trece în starea high tensiunea este atât de mică încât se poate neglija.Puterea pe comutator reprezintă produsul dintre tensiunea si curentul care trec prin acesta şi ea este aproximativ egala cu zero.

Modulația pulsurilor în durată utilizează un semnal dreptunghiular cu înalțimea funție de rezultatul variației valorii medii a undei.Se consideră o formă de undă f(t) cu o valoare de aplitudine minimă Ymin, o valoare maxima Ymax şi un ciclu ativ D, valoare medie a unei unde se calculează cu ajutorul formulei:

$$\bar{y} = \frac{1}{T}\int_0^T f(t)\,dt.$$

Forma de undă a semnalului este prezentată în figura 4 de mai jos.

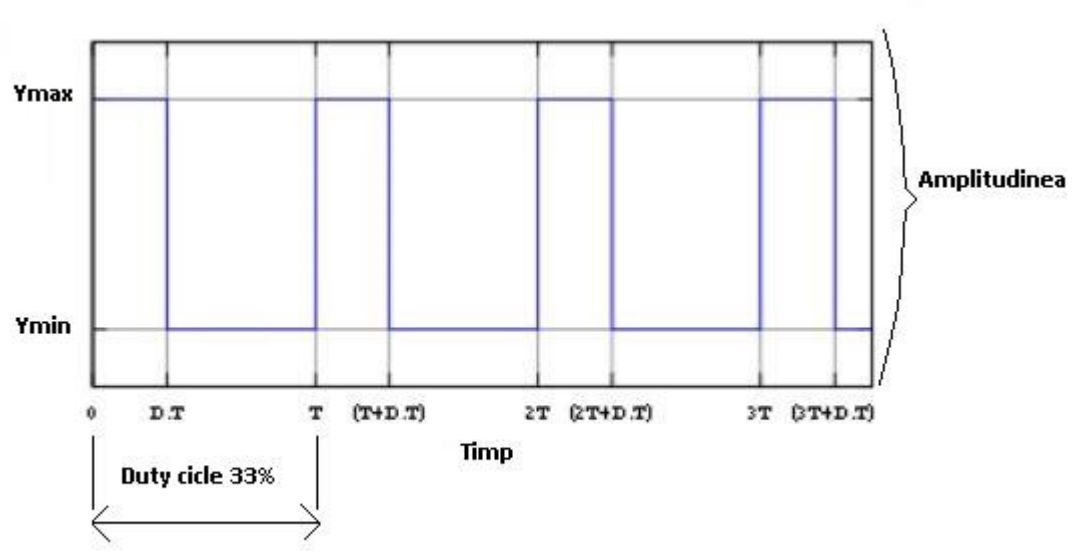


Figura.4 Forma de undă a semnalului PWM

Ştiindu-se că f(t) reprezintă un semnal dreptunghiular periodic, valoarea amplitudinii lui este y_{max} pe intervalul $0 < t < D \cdot T$ şi respectiv y_{min} pe intervalul $D \cdot T < t < T$. Luând în vedere aceste aspecte, formula de mai sus va deveni:

$$\begin{aligned}\bar{y} &= \frac{1}{T}\left(\int_0^{DT} y_{max}\,dt + \int_{DT}^{T} y_{min}\,dt\right) \\ &= \frac{D \cdot T \cdot y_{max} + T\,(1-D)\,y_{min}}{T} \\ &= D \cdot y_{max} + (1-D)\,y_{min}\end{aligned}$$

În majoritatea cazurilor expresia de mai sus se poate simplifica destul de mult deoarece, prepoderent Ymin = 0 şi deci $\bar{y} = D \cdot y_{max}$. Se observa din aceasta ca valoarea medie a semnalului ($\bar{y}$) este direct proporţională cu cilul activ (dutycycle) **D.**

Un avantaj enorm pe care il mai au motoarele DC este ca nu este necesar decât un puls PWM comparativ cu Motoarele Brushless, unde sunt necesare trei semnale, mai mult de atât Motoarele Brushless necesită circuite invertoare destul de complexe, greu de realizat şi pretenţioase la comandă.Pentru acestea este necesară sincronizarea celor trei semnale la momentul potrivit altfel motorul merge sacadat.

La motorul DC în primul rând se comandă cu un singur semnal PWM de putere realizat de un microcontroller şi aplificat cu ajutorul tranzistoarelor, ceea ce înseamnă economie de pini, în al doilea rând circuitul cu care se realizează comanda lui este extrem de simplu de realizat şi eficient.

Se prezintă în continuare diferite modalităţi de comandă a motoarelor de curent continu, o coamndă cu tranzistor Low Side Fig.5, o alta comandă cu tranzistor High Side Fig.6 şi comandă în punte H Fig.7.

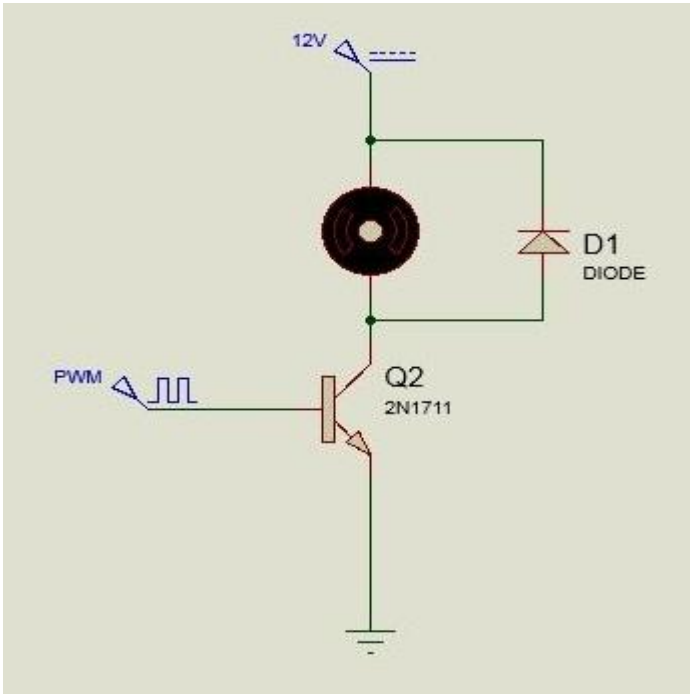


Figura.5 Schemă de comandă a motorului DC cu tranzistor Low Side

Acest tip de circuit reuşeşte să comande motorul doar într-un singur sens. Este folosit în aplicaţii de siguranţă.Dacă apare din greşeală un scurtcircuit motorul nu porneşte, ceea ce reprezintă un avantaj enorm.

Circuitul este alimentat la o sursă de curent continuu de 12V, sursă care este conectată direct la motor.La activarea unui semnal PWM pe baza tranzistorului Q2, se deschide jonctiunea colector – emitor, ceea ce lasă loc curentului din sursă sa treacă prin motor alimentându-l pe acesta.Motorul începe să se rotească într-un anumit sens.

La oprirea semnalului de activare se închide jonctiunea colector – emitor , motorul se opreşte, dar la oprirea motorului înfăşurările acestuia îmagazinează destulă energie pentru a se comporta ca o sursă fapt ce ar duce la arderea tranzistorului în dorința curentului de a se descărca.Datorită diodei care este conectată în paralel cu motorul, curentul din înfăşurările motorului au cale liberă de descărcare înapoi în sursă.

În continuare se prezintă o schemă de comandă a motorului DC cu tranzistor High Side.

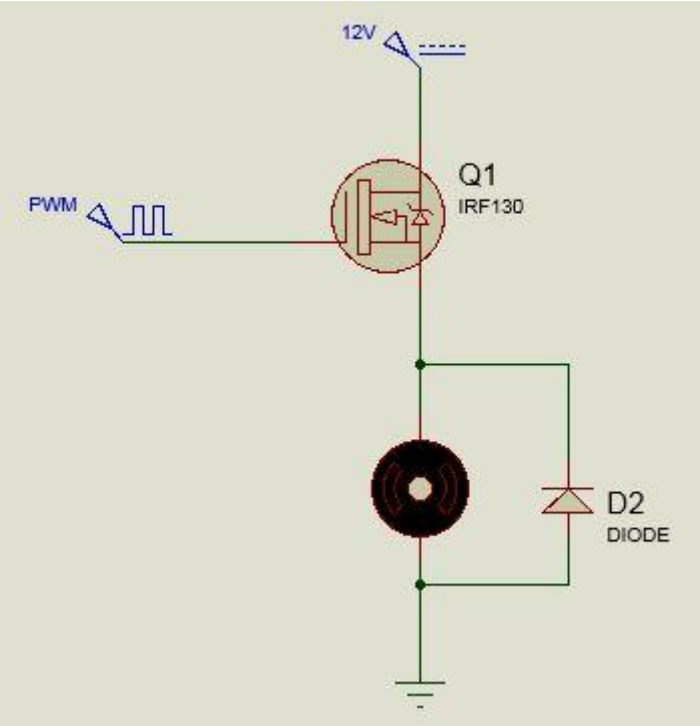


Figura.6 Schemă de comandă a motorului DC cu tranzistor High Side

La fel ca în celalt caz tranzistorul este comandat într-un singur sens.Ce e diferit în această schemă, tranzistorul bipolar Q2 a fost înlocuit cu un tranzistor mos, datorită necesității de descărcare a înfăşurării motorului DC de data aceasta era suprasolicitată jonțiunea emitoare şi chiar în acest caz curentul nu putea parcurge tranzisorul datorită faptului ca era blocat, astfel se ardea.Soluția simplă este tranzistorul mos care are legată o diodă zenner între drenă şi sursă care permite cu lejeritate descărcare înfăşurărilor motorului.

Aceste două scheme reprezintă unele dintre cele mai simple şi ieftine metode ce comandă a unui motor.

Există situații când apare necesitatea comandării unui motor DC în ambele sensuri de rotație.În acest caz ne vine în ajutor puntea H.

O schemă generală a unei punți H este prezentată în figura 7.

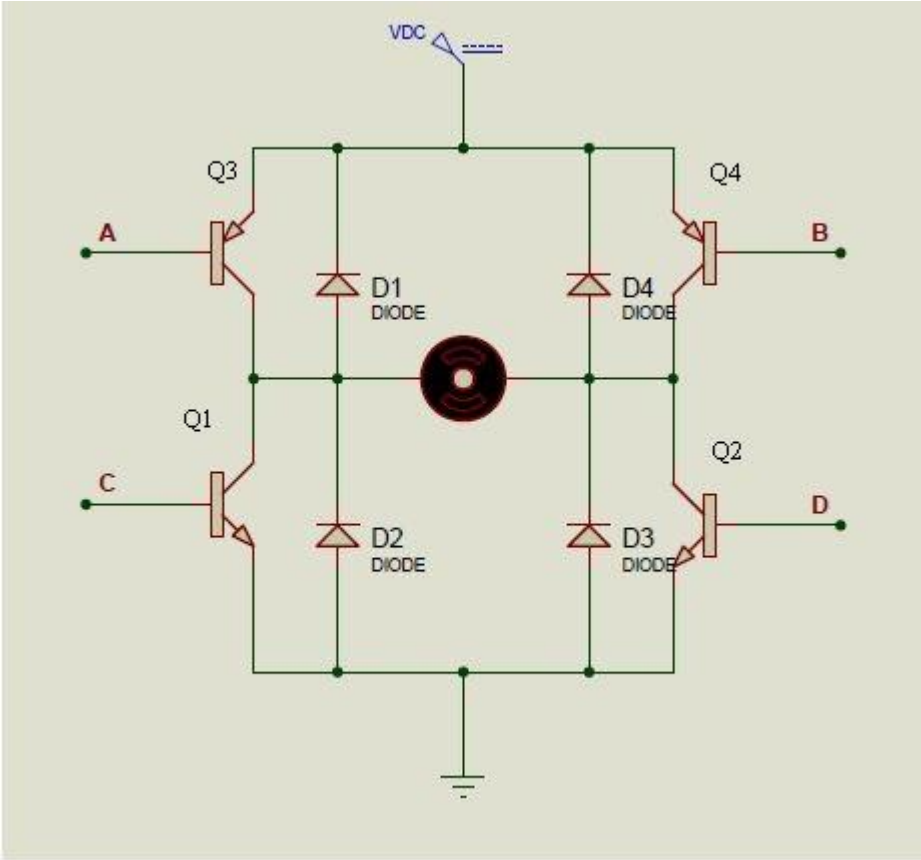


Figura.7 Schema generală a unei punți H

După cum se observă imaginea de mai sus prezintă o punte H, ea este formată din două punți de tranzistoare cu sarcina pe cetru, ceea ce formează litera H, de aici îi vine numele.

Deosebit față de celelate două circuite prezentate mai sus este faptul ca puntea H în funcție de cum sunt polarizate cele patru tranzistoare, permite trecerea curentului prin motor în ambele sensuri, prin urmare motorul se va învârti în ambele sensuri.

Astfel funționarea punții este prezentată în continuare.

Sursa VDC alimentează întregul circuit, se observă ca două tranzistoare din aceeasi punte nu pot lucre simultan deoarece se creează scurt între sursă şi masă ceea ce nu este de

dorit.Prin urmare tranzistoarele vor fi comandate două cate două din punți diferite pentru a realize trecerea curentului prin sarcină care în cazul nostru este motorul DC.

Se observă două cazuri pentru realizarea comenzii şi două cazuri de frânare.

Primul caz în care tranzistorul Q3 şi tranzistorul Q2 sunt activate, în această situație motorul se învârte în sens destrogir.

Cazul doi în care sunt activate tranzistoarele Q1 şi Q4, în această situație motorul se învârte în sens levogir.

Şi ultimele două situații în care tranzistoarele sun activate pe orizontală două cate două caz în care motorul este blocat.

Un lucru important care trebuie observat este că, chiar dacă puntea este compusă din patru tranzistoare, comanda întregii punți se poate realize cu două sau chiar cu un singur semnal PWM lucru extrem de avantajos şi efficient.

Apar şi în cadrul punții H, patru diode de descărcare, conectate antiparalel, care au acelaşi rol ca la circuitele de mai sus, şi anume descărcarea înfăşurărilor motorului, pentru a înlatura posibilitatea distrugerii tranzistoarelor.

În tabelul următor se prezintă logica de comandă a punții H.

Sensul de rotație	A	B	C	D
Levogir	0	1	1	0
Destrogir	1	0	0	1
Oprire	1	1	0	0
Oprire	0	0	1	1

Tabelul Nr.1 Logica de comandă a punții H

În construcția unei punți H pot fi utilizate atât tranzistoare bipolar cât şi cele MOS.

Există circuite integrate care înglobează în interiorul lor una sau mai multe punți H, dar acestea nu integrează şi diodele de descărcare din necesitatea de a evita supraîncălzirea circuitului.Un exemplu de astfel de punte H integrată este circuitul L298N.

2.3 Circuitul de comanda al motoarelor L298N

Pentru a realiza comanda motoarelor din aplicația prezentată s-a folosit circuitul driver L298N.Circuitul are integrat intr-o capsulă Multiwatt 15 două punți H.

Acest circuit se poate folosi atât la comanda motoarelor pas cu pas cât şi a motoarelor de curent continuu.Datorită arhitecturii sale fine, circuitul oferă posibilitatea relizării unui montaj pentru comandă în putere la îndemâna oricui.

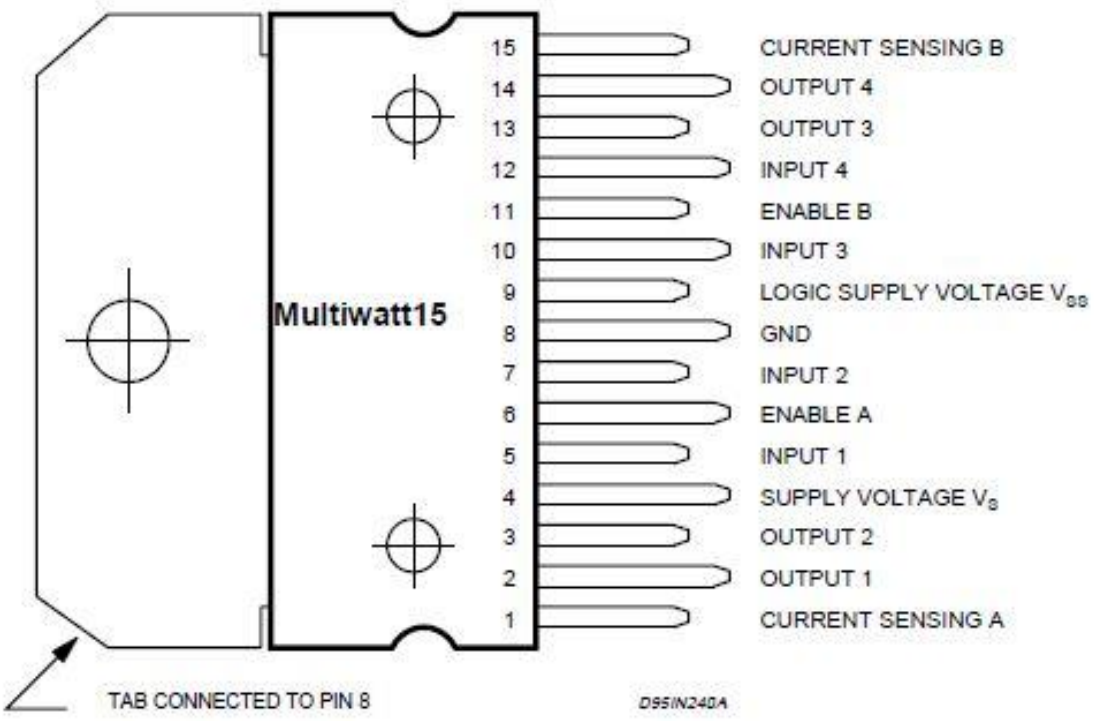


Figura.8 Schema cu pinii circuitului L298N

Din figura de mai sus se observă că circuitul are 15 pini, 4 iesiri de comandă pentru motoare respective patru intrări de comandă pentru punte.

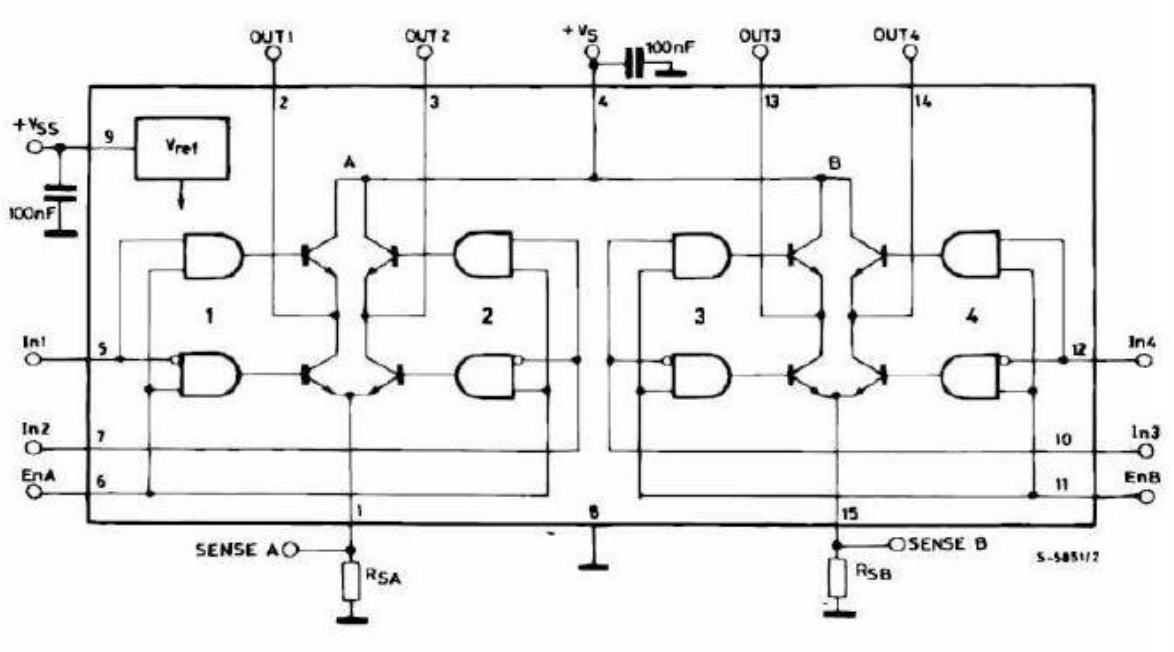


Figura.9 Schema bloc a circuitului L298N

Din schema bloc se observă că circuitul are nevoie doar de 3 pini de comandă, deoarece tranzistoarele in linie de pe ramura unei punți folosesc semnal comun de comandă.Semnalul de comadă ajunge la 4 tranzistoare prin intermediul a 4 porți AND.

Cele patru tranzistoare de jos primesc semnalul complementat față de superioarele lor care primesc semnalul curat, datorită faptului că porțile lor de comandă au câte o intrare activată pe semnal "Low" şi aceasta inversează semnalul. Semnalul de comandă Enable ajunge la cele patru tranzistoare de jos prin intermediul porților AND cu intrări commune.Datorită acestei implementări tranzistoarele de pe acelaşi braț nu vor intra niciodată simultan în conducție, prin urmare se evită scurtul sursei la masă.

Circuitul oferă posibilitatea de a verifica curenții care trec prin cele două punți prin intermediul a două rezistențe de 1 Ohm şi 3W, ce sunt conectate la pinul 1 respectiv pinul 15 al circuitului.Tot furnizorul ne mai specifcă faptul că pentru a obține o informație cât mai reală ca curentului ce traversează cei doi rezistori trasele la care sunt conectați pinii să fie cât mai subțiri, de aşa natură pentru a evita rezistențele parasite.

Se prezintă în tabelul 2 de mai jos parametri principali de care trebuie ținut cont în utilizarea circuitului.

Simbol	Parametru	Valoare tipică	Valoare Minimă	Valoare maximă	Unitate de măsură
Vss	Nivel Logic	5	4.5	7	V
Vs	Alimentare motoare		Vih + 2.5	46	V
Io	Curent de ieşire			2	A
Fc	Frecvența comutație	25		40	KHz

Tabelul Nr.2 Date de catalog circuit L298N

2.4 Comunicarea seriala RS232

Transmisia de date a evoluat de la calculatoare care comunicau prin diferite echipamente periferice la rețele complexe de calculatoare. Chiar la înlăturearea firelor şi paşirea spre mediul de comunicare wireless.

Prin comunicarea serial se poate înțelege transferul secvențial al unor informații intre două puncte de comunicare, aşa cum este numit în specialitate legătură punct cu punct.

Deși comunicarea paralelă este mult mai rapidă şi mai eficientă, decât cea serială majoritatea încă mai optează pentru comunicarea serială RS232 datorită costurilor reduse de întreținere, ca cablurilor si a conectorilor.Există limitări datorită distanței care nu pot fi depăşite de comunicarea paralelă.Comunicarea serială se realizează bit cu bit.

Comunicațiile sunt caracterizate de trei elemente pricipale:

- **Semnale –** controlul fluxului de date, rezolvarea erorilor de transmisie
- **Date –** cantitate, codificare, decodificare
- **Temporizări –** sincronizarea intre emitor si receptor, frecvența

După modul în care se realizează comunicarea avem comunicare serială asicronă şi comunicare serială sincronă.Comunicarea serială sincronă utilizează un semnal de ceas cu care identifică momentul în care o dată este validă.Comunicarea serioală asincronă se

foloseşte de modul în care este structurată informaţia transmisă şi realizează sincronizarea între receptor şi emitor cu ajutorul acesteia.

Pentru o înțelegere cât mai bună între echipamentele produse de diverse firme s-au introdus nişte standarde pentru comunicarea serială.Aşa apare şi devine cel mai cunoscut standard comunicarea serială RS 232.

RS 232 este un standard de comunicare bidirecţională , asincronă, folosit pentru a trasmite codul de caractere ASCII.Bitul de informaţie se codifică prin nivele mai mari de 3 V pentru „0” logic şi prin nivele de tensiune mai mici de -3 V pentru „1” logic. Valorie de tensiune pot ajunge până la +-25V.

În figura 10 de mai jos este prezentat structura pachetului de informaţie si cum se transmite prin sandardul RS 232.

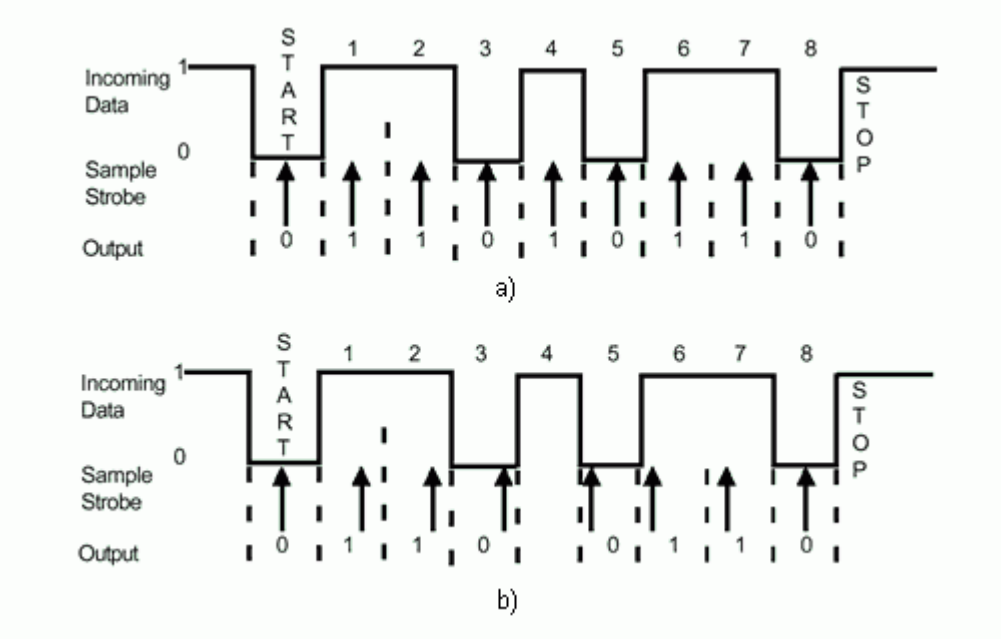


Figura.10 Transmiterea pachetului de informatii prin standardul RS232

Dacă atât emiţătorul cât şi receptorul au semnal de tact comun atunci se spune că avem lucru în mod sincron.În cazul în care tactele sunt separate avem comunicare asincronă.

În **comunicarea asincronă**, nu se emite tact în acelaşi timp cu datele, ci se introduce cate un ***bit de start***, la începutul fiecărui octet ce se transmite.Pnetru fiecare character se realizează o transmisie independentă, cu ajutorul biţilor de Start,Stop şi Paritate.Viteza cu care se lucrează este stabilită la începutul transmisiei.Pentru a fi în fază, receptorul detectează bitul de Start.Pauza între caractere este necesară pentru funcţionarea corectă, iar ea se relizează cu bitul de Stop.

La **comunicarea sincronă**, mesajul se transmite forate rapid, biți sunt transmiși unul dupăaltul și nu este necesară apariția biților de Start și de Stop.Pentru realizarea sincronizării mesajul transmis este precedat de caractere sepeciale de sincronizre, care vor fi detectate de receptor. Caracterele de sincronizare sunt transmise continuu chiar dacă nu se sunt transmise date.

Codificarea datelor și controlarea erorilor

Erorile de transmisie apar atunci când circuitele funcționează în medii zgomotoase ca de exemplu: diferite câmpuri electrice ale circuitelor de putere, activarea unor motoare mari (industriale), etc. Aceste semnale sunt transmise și recepționate fără control de firele de comunicații care au o comportare asemănătoare unei antene.Deoarece în calculatoarele tensiunile cu care se lucrează sunt de valori foarte mici, aceste zgomote manifestă un efect pregnant, astfel se caută să se realizeze filtrari masive, care să creeze o formă de imunitate asupra circuitului.

Metodele de corecție și detecție a erorilor sunt întalnite în mod frecvent în scrierile pe DVD-uri.Aceste metode implic introducerea de informații care sunt neesențiale, pe lângă datele utile.Se cunosc următoarele metode:

- **Sume de control** pe blocuri – se aplică simplu dar nu sunt prea eficiente
- **Biți de paritate –** se aplică simplu, dar nu oferă siguranță mare
- **Împărțire polinomială** – greu de calculat, oferă securitate

Un exemplu ar fi acela că receptorul trimite înapoi o copie a datei primite.În acest mod se înjumătățește lățimea benzii folosite.O alternativă ar mai fi ca emițătorul să transmită dată urmată de o copie a ei.

Toate metodele de tratare a erorile utilizazează informații redundante.În cele mai multe cazuri informațiile sunt codificate înainte de a fi transmise.

Sumele de control pe blocuri reprezintă o metodă de detecție a erorilor. Prima oară datele se împart pe blocuri, care apoi sunt însumate obținându-se o sumă care apoi este trunchiată, inversată și adăugată la sfârșit.La recpeție, blocurile primite incluzătoare ale sumei de la sfârșit, se adună pe măsură ce soses și dacă suma obținută nu este egală cu zero atunci

înseamnă că datele sunt eronate și este necesară o retransmitere a datelor.Nu se poate reliza corecția erorii. Pentru a identifica și corecta erorile multiple s-a dezvoltat mecanismul BCH.

Paritatea este cea mai discutata metoda de detecție a erorilor pentru protecția transmisiilor seriale de caractere ASCII. La oricare din metode, emițătorul prelucrează o parte din date și generează un fel de semnătura pe care apoi o transmite împreună cu date utile. Când mesajul ajunge la receptor, acesta prelucrează datele primite si generează o semnătură pe care o compară cu cea primită. Dacă cele doua semnături nu coincid, atunci s-a produs o eroare. Metoda bitului de paritate se poate aplica pentru date binare de orice lungime. Pentru fiecare cuvânt este adăugat un bit de paritate (semnătura). Paritatea poate fi pară (cuvântul conține un număr par de 1) sau impară (cuvântul conține un număr impar de 1). Calcularea parității se poate face cu operatorul XOR (SAU Exclusiv) între biții cuvântului. Prin această metodă este posibila doar **detecția erorii singulare**, când sunt afectați un număr impar de biți. O eroare dublă (afectează un număr par de biți) nu poate fi detectată prin acest mecanism. Prin urmare, această metodă nu oferă prea multă securitate. Un singur bit de paritate nu oferă informații despre poziția erorii.

Standardul RS 232 se poate implementa prin două protocoale, unul hardware și unul software.

Protocolul software folosește coduri XON/XOFF, intregrate în blocul de date, pentru a porni și a opri fluxul de date atunci când este necesar.

Protocolul hardware presupune folosire de semnale fizice DTR/DSR pentru controlul transmisiei. Cu ajutorul acestor semnale receptorul poate să oprească fluxul de date transmis.

DTR/DSR au același rol ca XON/XOFF, cu deosebirea ca sunt semnale fizice și nu sunt integrate prin software de către utilizator.

Se prezintă mai jos în figura 11 modul în care se conectează atât transmițătorul cât și receptorul în cazul celor două protocoale pentru realizarea comunicației bidirecționare.

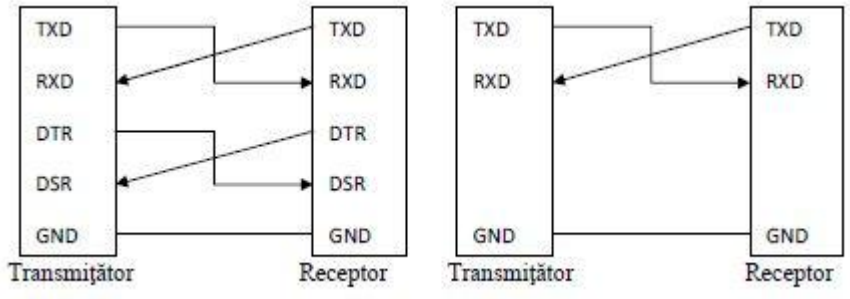


Figura.11 Modul de conectare al transmițătorului la receptor

2.5 Microcontroller-ul MCF5213

Microcontroller-ul MCF5213 face parte din categoria de microprocesoare RISC (Reduced Instruction Set Computing).

Acest microcontroller funcționează pe 32 de biți şi opereză la o frecvență maximă de 80 Mhz, oferă astfel performanțe ridicate şi consum mic de energie funcționând la 3.3 V.

Memoria disponibilă include pana la 256 Kocteti de memorie flash și 32 Kocteti de memorie SRAM (Static Random Access Memory). Modulele prezente pe chip includ:

- Versiunea 2-a a nucleului ColdFire care executa 76 Milioane de Instrucțiuni Pe Secunda (MIPS) la o frecventa de 80 MHz rulând din memoria Flash cu unitatea MAC (Multiply Accumulate) și divizor hardware
- Modul queued serial peripheral interface (QSPI)
- Convertor Analog Digital cu 8 canale de 12 biți
- Controler pentru acces direct la memorie (DMA) cu 4 canale
- 4 temporizatoare cu posibilitatea comparării valorilor pe 32 biți care pot fi folosite pentru captura intrării/ieșire (DTIM)
- Modul FlexCAN controller area network (CAN)
- Trei porturi UART (Universal asynchronous/synchronous receiver/ transmitters)
- Modul interfață magistrală circuit Inter-integrat (I2C)
- Temporizator pentru uz general (GPT General Purpose Timer) cu patru canale utilizabil pentru comparări capturaintrării/ ieşire, modularea pulsurilor în durată (PWM) , și acumularea pulsurilor
- Temporizator pentru modularea lățimii pulsurilor cu 8 canale/ 4 canale, pe 8 biți/ 16 biți
- Două temporizatoare pe 16 biți pentru întreruperi periodice (PIT)
- Modul temporizator de tip watchdog programabil software
- Controler de întreruperi capabil să se descurce cu 52 de surse
- Port pentru acces în modul test/ depanare (JTAG, BDM)

În următoarea figură se prezintă diagrama bloc a dispozitibului.

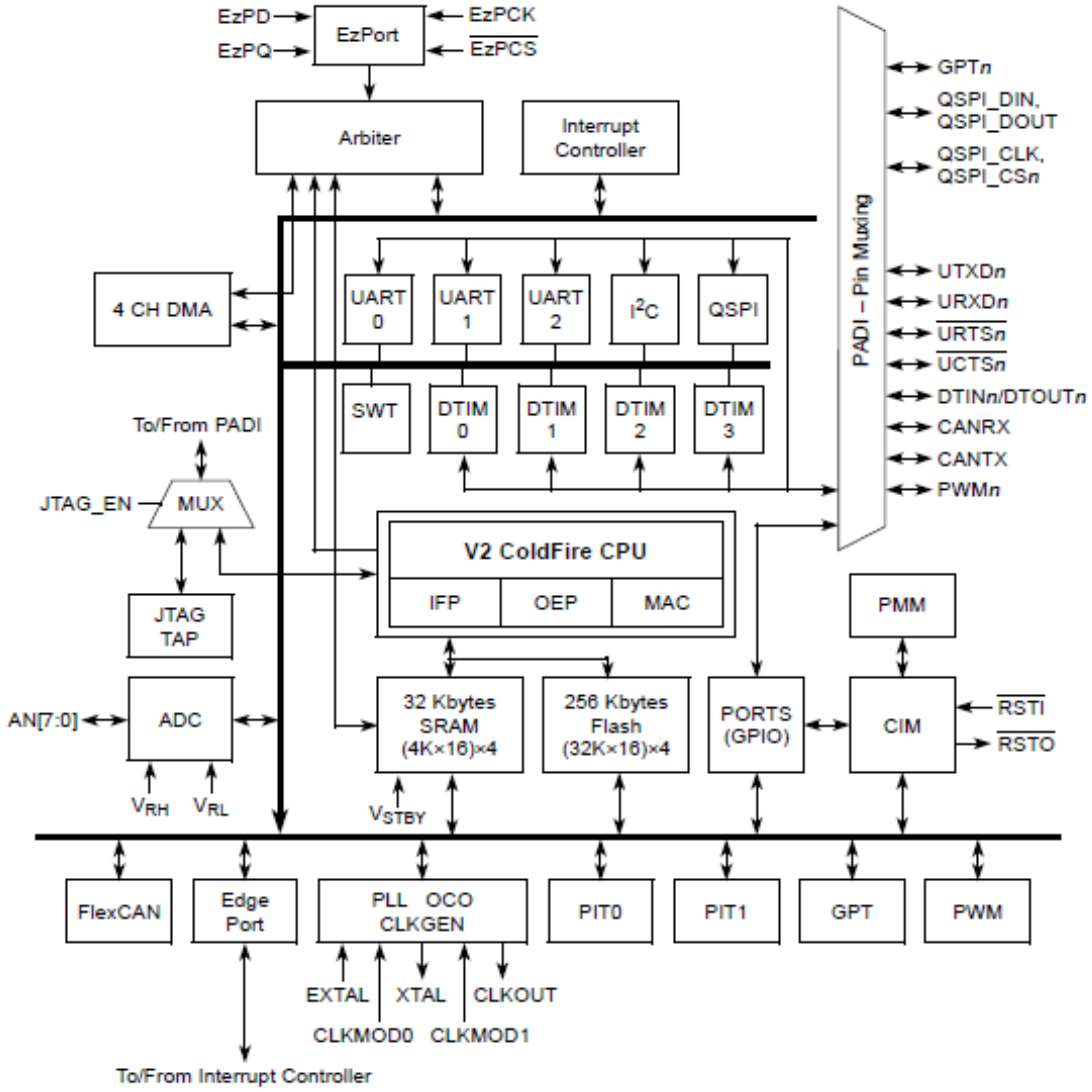


Figura .12 Diagrama bloc a dispozitivului FreeScale

Deși microcontrolerul MFC5213 dispune de multe opțiuni și avantaje, vom discuta doar despre acelea care ne interesează în mod direct pentru proiectul de față.Acestea sunt porturile UART, temporizatorul pentru utilizări generale (GPT) și modulele temporizator pentru modularea lățimii pulsurilor.

Porturile UART

Dispozitivul are trei porturi UART full-duplex care funcționează independent. Cele trei porturi pot fi cronometrate cu semnalul de ceas al magistralei, eliminând astfel necesitatea utilizării unui semnal de ceas extern. Pentru pachete de dimensiune mai mică, cel de-al treilea port UART este multiplexat cu funcții de intrare/ieșire.

Temporizatorul pentru uz general (GPT)

Temporizatorul pentru uz general constă dintr-un modul temporizator programabil de 16 biți cu patru canale dirijat de un scalor programabil de 7 etape .Fiecare din cele patru canale poate fi folosit pentru captarea datelor de intrare sau compararea ieșirii. În plus, canalul trei poate fi configurat să funcționeze că un acumulator de puls. O funcție pentru depășireatemporizatorului permite software-ului extinderea capacității de cronometrare a sistemului dincolo de intervalul de 16 biți ai contorului.

Funcțiile pentru captura intrării și compararea ieșirii permit măsurarea unor forme de undă de intrare și simultan generarea unor forme de undă la ieșire. Funcția de captură a intrării poate face captura în momentul unei tranziții. Funcția de comparare a ieșirii poate genera forme de undă la ieșire și întârzierile software pentru temporizator. Acumulatorul pe 16 bițipoate funcționa caun contor de evenimente sau că un acumulator dependent de timp.

Descrierea funcțională a regiștrilor pentru PWM

Regiştrii folosiți pentru generarea de semnale PWM sunt descrişi în tabelul de mai jos:

Table 24-1. PWM Memory Map

IPSBAR Offset[1,2]	Register	Width (bits)	Access	Reset Value	Section/Page
Supervisor Read/Write Only Access					
0x1B_0000	PWM Enable Register (PWME)	8	R/W	0x00	24.2.1/24-3
0x1B_0001	PWM Polarity Register (PWMPOL)	8	R/W	0x00	24.2.2/24-4
0x1B_0002	PWM Clock Select Register (PWMCLK)	8	R/W	0x00	24.2.3/24-4
0x1B_0003	PWM Prescale Clock Select Register (PWMPRCLK)	8	R/W	0x00	24.2.4/24-5
0x1B_0004	PWM Center Align Enable Register (PWMCAE)	8	R/W	0x00	24.2.5/24-6
0x1B_0005	PWM Control Register (PWMCTL)	8	R/W	0x00	24.2.6/24-7
0x1B_0008	PWM Scale A Register (PWMSCLA)	8	R/W	0x00	24.2.7/24-8
0x1B_0009	PWM Scale B Register (PWMSCLB)	8	R/W	0x00	24.2.8/24-9
0x1B_000C + *n* *n* = 0–7	PWM Channel *n* Counter Register (PWMCNT*n*)	8	R/W	0x00	24.2.9/24-9
0x1B_0014 + *n* *n* = 0–7	PWM Channel *n* Period Register (PWMPER*n*)	8	R/W	0xFF	24.2.10/24-10
0x1B_001C + *n* *n* = 0–7	PWM Channel *n* Duty Register (PWMDTY*n*)	8	R/W	0xFF	24.2.11/24-11
0x1B_0024	PWM Shutdown Register (PWMSDN)	8	R/W	0x00	24.2.12/24-12

1 Addresses not assigned to a register and undefined register bits are reserved for expansion. Write accesses to these reserved address spaces and reserved register bits have no effect.

2 A 32-bit access to any of these registers results in a bus transfer error.

Figura .13 Memoria PWM

PWM Clock Select

Microcontrollerul conține patru semnale de ceas (TIMERE) și anume ceasul A ,B ,A Scalat și B Scalat toate se leagă de ceasul pricipal al sistemului.

Semanlele de ceas A și B se pot configura să ruleze la 1, ½, ..., 1/128 din perioda ceasului principal al microcontrollerului, semnalele de ceas SA și folosesc ceasul A și B ca intrări pe care le împart mai departe prin intermediu unui contor reîcărcabil.Frecvențele ceasurilor SA și SB sunt frecevențele ceasurilor A, B îpărșite la 2,4,6,8...,512.

Fiecare canal PWM are capacitatea de a folosi unul dintre cele două grupuri de ceasuri prescalat (A sau B) sau scalat (SA sau SB).Diagrama de mai jos reprezintă cele patru ceasuri și modul lor de scalare .

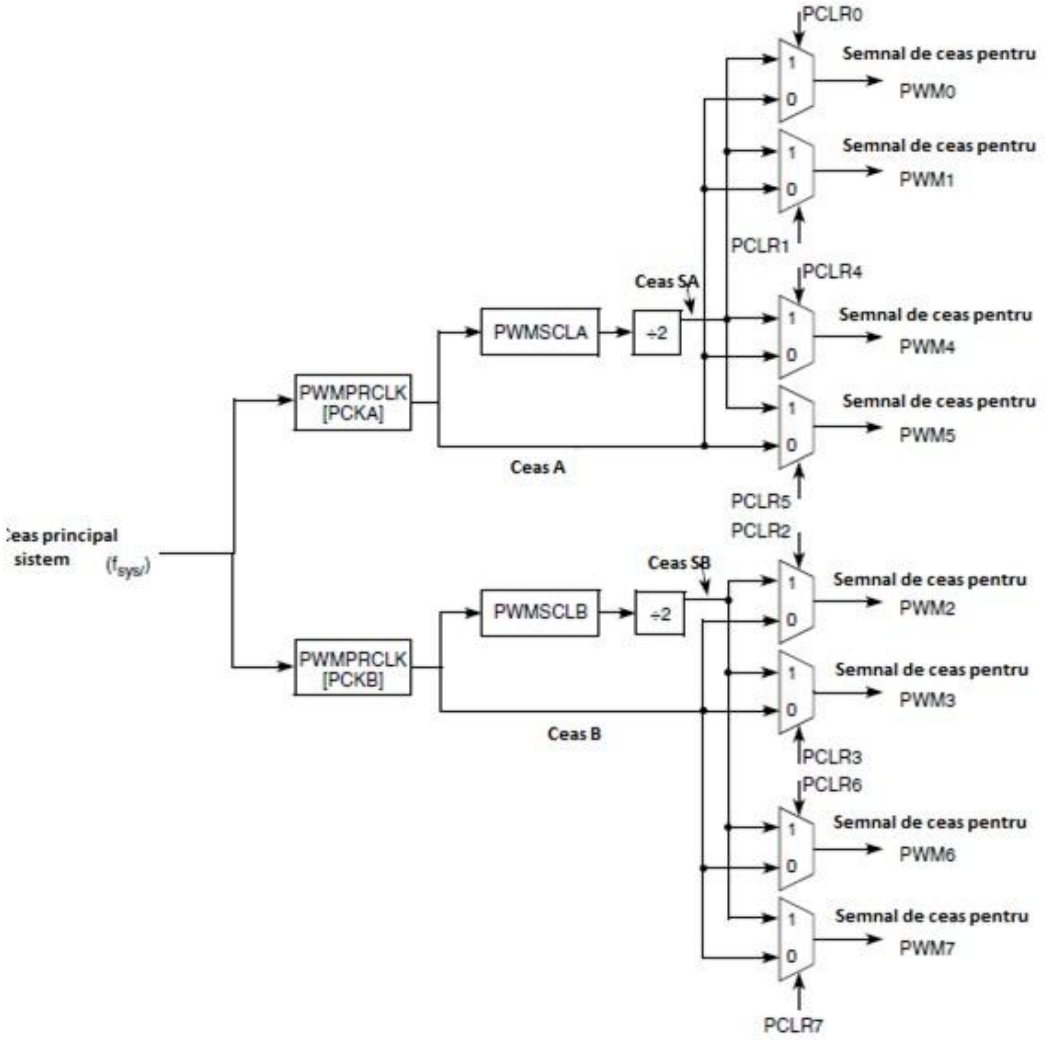


Figura.14 Diagrama bloc pentru selecția semnalului clock PWM

Semnalele de ceas prescalate (A sau B)

Semnalul de ceas principal al sistemului este considerat intrare pentru prescalorul PWM care poate fi dezactivat atunci când dispozitivul este în modul de depanare prin setarea bitului corespunzător al registrului PWMCTL[PFRZ]. Acest procedeu este util pentru reducerea puterii consumate și pentru a putea înghețа semnalul PWM din ieșire intr-o anumită stare în modul de depanare. Semnalul de ceas principal al sistemului nu mai este considerat intrare atunci când toate canalele PWM sunt oprite (PWMEn=0)

Semnalele de ceas A sau B sunt de fapt semnale obținute din semnalul de ceas principal, divizat cu o anumită valoare. Valoarea cu care dorim să divizăm semnalul principal de ceas va fi 1, 1/ 2, … , 1/128. Valoarea selectată pentru semnalele de ceas A și B este determinată debiții PWMPRCLK [PCKAn] și respectiv PWMPRCLKB [PCKBn].

Semnalele de ceas scalate (SAsau SB)

Semnalul de ceas scalat SAsau SB folosesc caintrări semnalele de ceas A respectiv B, și le divid în continuare cu o valoare programabilă de utilizator, și după aceea împart totul la 2. Frecvența disponibilă pentru semnalul de ceas SApoate fi selectată cafiind frecvența de funcționare a semnalului de ceas A împărțită la 2, 4, … , 512. Analog se desfășoară pentru semnalul de ceas scalat SB.

Semnalul de ceas scalat SAeste egal cu semnalul de ceas A împărțit la doi ori valoarea din registrul PWMSCLA:

$$Semnal\ ceas\ SA = \frac{Semnal\ ceas\ A}{2 * PWMSCLA}$$

În mod similar, semnalul de ceas SB este generat dupăurmătoareaecuație:

$$Semnal\ ceas\ SB = \frac{Semnal\ ceas\ B}{2 * PWMSCLB}$$

Ca exemplu, să considerăm cazul în care utilizatorul scrie valoarea 0xFF în registrul PWMSCLA. În aceste condiții, semnal de ceas A este selectat că fiind semnalul de ceas principal divizat cu 4. Un puls este generat cu o frecvență de o dată la 255*4 perioade ale semnalului principal.Trecând acest semnal prin două circuite divizoare, avem un semnal de ceas obținut din semnalul principal de ceas divizat cu 2040. În mod similar, o valoare de 0x01 în registrul PWMSCLA înseamnă selectarea semnalului de ceas A casemnal intern de ceas, acesta fiind obținut din semnalul intern principal de ceas, divizat de 8 ori.

Scrierea unor valori în regiștriiPWMSCLA sau PWNSCLB va avea că și consecințareîncărcarea contoarelor pe 8 biți asociate .

Scrierea valorilor în regiștriide scalare cât timp canalele sunt operaționale, poate cauza neregularități în semnalele PWM dinieșire.

Alegerea semnalului de ceas

Fiecare canal PWM prezintă posibilitatea selectării unuia din cele două semnale de ceas disponibile. Pentru canalele 0, 1, 4 și 5 alegerea se poate face dintre semnalele de ceas Ași SA. Pentru canalele 2, 3, 6, și 7 se poate alege unul din semnalele de ceas B sau SB. Selecția semnalului de ceas se face prin setarea biților de control ai registrului PWMCLK [PCLKx]. Modificarea valorilor biților de control când canalele sunt operaționale poate produce neregularități în forma de undă de la ieșire.

Temporizatoare pentru canalele PWM

Partea cea mai esențiala a modulului PWM sunt temporizatoarele în sine. Fiecare canal al temporizatorului are un numărător, un registru perioadă (registru pentru stabilirea valorii perioadei) și un registru ciclu activ (pentru stabilirea valorii ciclului activ (dutycycle)). Acești regiștrii au dimensiunea de opt biți. Perioada formei de undă din ieșire este controlată de momentul în care valoarea din numărător este egală cu cea din registru perioadă. Ciclul activ este controlat prin momentul în care valoarea din numărător este egală cu cea din registru ciclu activ, ceea ce permite castarea formei de undădinieșire să se modifice de-a lungul unei perioade. Polaritatea de început a semnalului de ieșire este programabilă pentru fiecare canal în parte.

În figură este prezentată o diagrama bloc pentru un temporizator PWM:

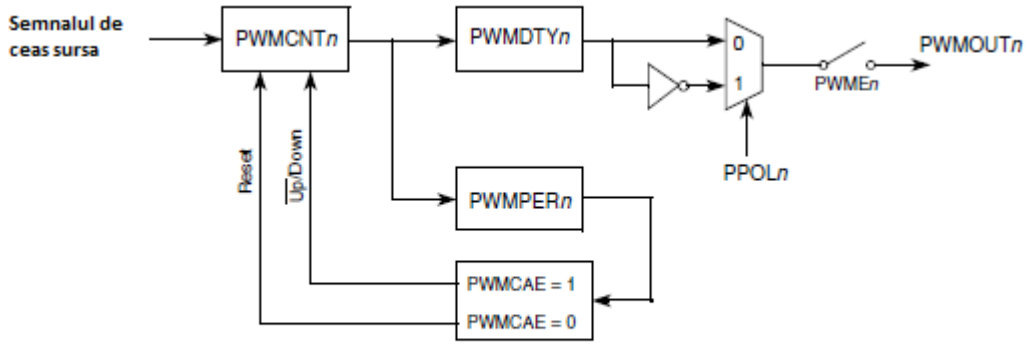
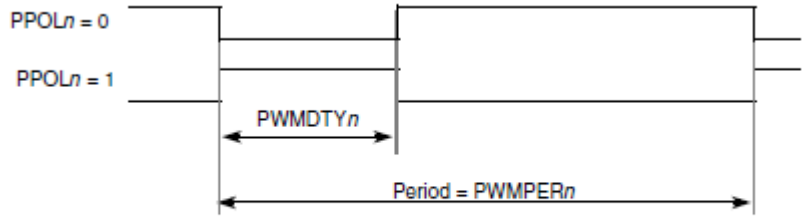


Diagrama bloc a temporizatorului PWM

Figura .15



Iesire PWM de tipul aliniata la stanga

Figura 16

Pentru a calcula frecventa de ieșire a semnalului cu tipul ieșirii pentru un anumit canal aliniată la stânga, vom lua frecvența semnalului de ceas sursă (A, B, SA, SB) și o vom împărți cu valoarea din registrul perioadă a canalului respectiv:

$$frecvența\ semnal\ PWMn = \frac{frecventa\ semnal\ ceas\ (A, B, SA, SB\ )}{PWMPERn}$$

Ciclul activ (procentul de timp dintre cât este semnalul în starea “high “ și durata unei perioade) este exprimat ca:

$$Ciclu\ activ = \left(1 - PWMPOL[POLn] - \frac{PWMDTYn}{PWMPERn}\right) * 100\%$$

Exemplu pentru ieșirea aliniată la stânga

Pentru a lua un scurt exemplu vom considera cazul următor:

Semnalul de ceas sursă = semnalul de ceas principal cu frecvența 40 MHz (o perioadă de 25 ns)

PPOLn= 0, PWMPERn = 4, PWMDTYn = 1

Frecvența PWMn = 40 MHz/4 = 10MHz

Perioada PWMn= 100ns

$$Ciclul\ activ\ PWM = \left(1 - \frac{1}{4}\right) * 100\% = 75\%$$

Forma de undă generată la ieșire este:

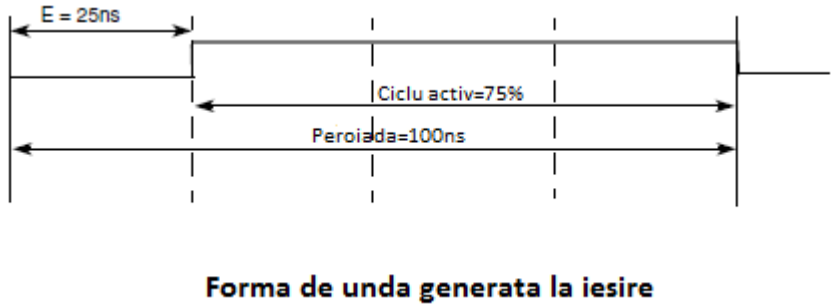


Figura 17

Ieșiri aliniate în centru

Pentru selecția tipului ieșirilor că fiindaliniate în centru, trebuie setat bitul corespunzător din registrul PWMCAE [CAEn]si ieșirea respectivă va fi aliniată în centru.

În acest caz numărătorul pe 8 biținfuncționează canumărător în sens crescător/descrescător și el este setat canumărătorcrescător (în sus) atunci când valoarea acestuia va fi 0x00. Valoarea numărătorului va fi comparată cu valoarea din cei doi regiștrii. Atunci când valoarea numărătorului va fi egală cu valoarea din registrul ciclu activ,bistabilul din ieșireîși va schimba starea, cauzând astfel și schimbarea stării semnalului PWM dinieșire. O egalitate între valoarea numărătorului și aregistrului perioadă va schimba modul de numărare al acestuia de la numărare în sus la numărare în jos .

2.6 Senzor de distanță Sharp 2Y0A21 (10-80cm)

Senzorul reprezintă una dintre cele mai importante inovații ale erei tehnologice.El a apărut o dată cu dezvoltarea microelectronicii, împreună cu alte noțiuni de mare impact, cum ar fi cele de „microprocesor”, „microcontroller”, adăugând o noțiune nouă unei terminologii tehnice având o anumită redundanță.Astfel, o mare parte din elementele tehnice senzitive sunt încadrate în categoria de *traductor*. Un traductor este un dispozitiv care convertește efecte fizice în semnale electrice, ce pot fi prelucrate de instrumente de măsurat sau calculatoare.

În unele domenii, în special în sfera dispozitivelor electro-optice, se utilizează termenul

de *detector* (detector în infraroşu, fotodetector etc.).

Ce este senzorul? Trebuie spus că nu există o definiție unitară şi necontestată a „senzorului", motiv care lasă mult spațiu pentru interpretări, ambiguități şi confuzii. Mulți autori preferă să folosească sintagma *„senzori şi traductoare"*, în cadrul căreia, fie pun pe picior de egalitate senzorul şi traductorul, utilizând, alternativ sau preferențial, unul dintre termeni, fie consideră că unul reprezintă o categorie ierarhică superioară, incluzându-l pe celălalt.

Senzorii IR pot fi active sau pasivi si sunt realizati pentru diferite tipuri de aplicatii dar în general sunt folosiți pentru masurari de distanțe.

În figura de mai jos este prezentat modul de detecție al unui obstacol.

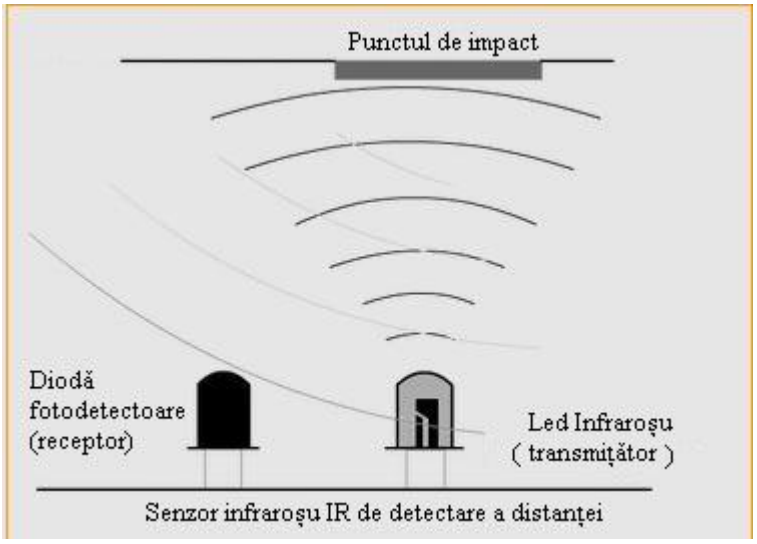


Figura 18

Principiul este simplu Ledul infraroşu transmite un semnal luminos înainte semnal care este apoi este reflectat de un obiect (punctul de impact).Lumina reflectată este captată de dioda fotodetectoare.

Senzorul Sharp 2Y0A21 este un sezor de distanță care măsoară distanța în centimetri.

Tipul de informație pe care o transmite la ieșire este analogică și variază intre 3.1 la distanță de 10 cm și 0.3 la 80 cm.

Figura.19 - Senzor Sharp 2Y0A21

Tabelul de mai jos prezintă caracteristicile senzorului Sharp 2Y0A21

Caracteristici	**Valoare**
Tensiune de alimentare	4.5V- 5.5V
Distanța minimă de măsurare	10cm
Distanța maximă de măsurare	80cm
Curentul mediu de alimentare	30mA
Timpul de răspuns	38 ± 10ms
Greutate	3.5g

Tabel Nr.4 – Cracteristici senzor Sharp 2Y0A21

Figura de mai jos ilustrează diagrama bloc a senzorului Sharp 2Y0A21.

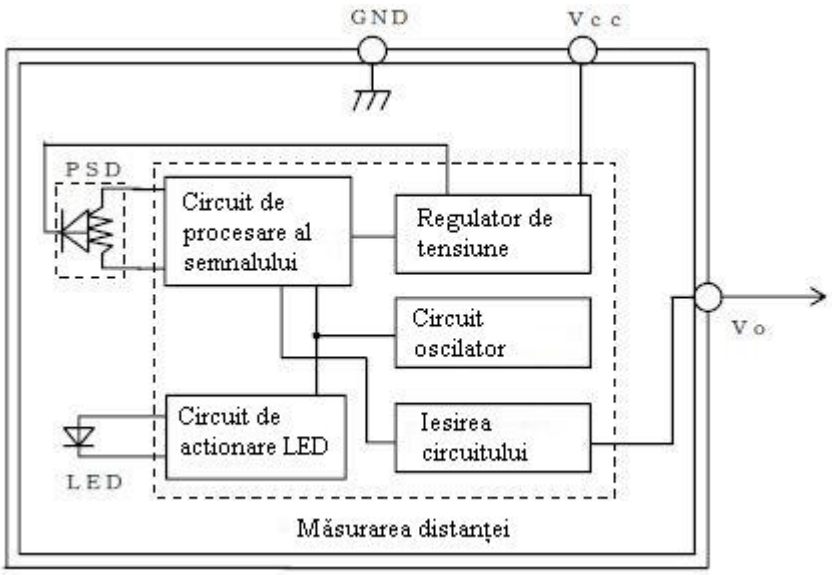


Figura.20 - Diagrama bloc a senzorului Sharp 2Y0A21

Modul de functionare al senzorului este descris în continuare.Circuitul este alimentat de la VCC, tensiunea este stabilizată și redusă pentru activarea întregului ansamblu de blocuri.În urma alimentarii oscilatorului acesta activează circuitul de actionare care comandă în continuare Ledul IR.El transmie un semnal luminos care la contactul cu un obiect se intoarce înapoi și este recepționat de catre dioda fotodetectoare care converteste semnalul luminos în semnal electric.Acesta este preluat și introdus în circuitul de procesare al semnalului unde suferă mici modificări, filtrări ca apoi sa fie transmis la iesire.

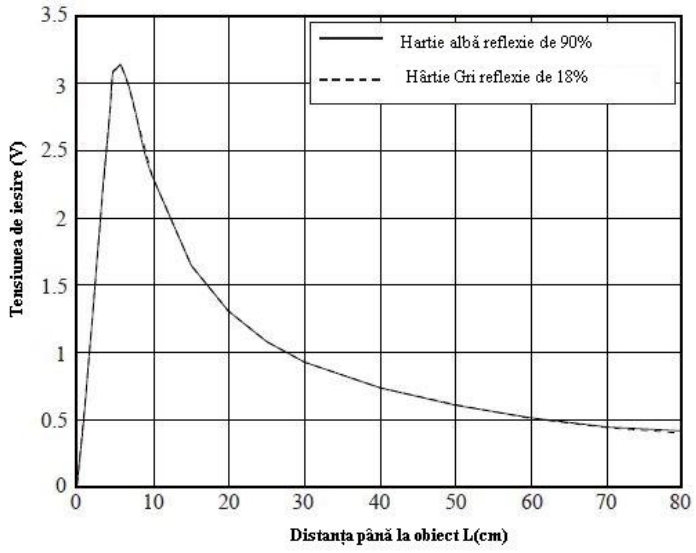


Figura.21 - Diagrama Tensiune funcție de distanță

Formula ce translează vloarea senzorului in distanță este următoarea:

Distanța (cm) = 4800/(Valoarea Senzorului - 20)

Trebuie specificat faptul că scalarea *valorii senzorului* se face intre 80 si 500.

Din diagramă se observă ca la valori ale distanței <8 cm tensiunea scade la fel și la valori >8 cm astfel se obțin tensiuni similare la distanțe diferite.Din această cauză s-a ales ca prag valoarea de 10 cm pentru a înlătura erorile de măsurare.

Tot din diagramă se prezintă dezavantajul major al senzorului IR și anume faptul că la suprafețe GRI coeficientul de reflexie este de 18%, ca la suprafețe negre să tindă spre zero.Ceea ce rezultă a fi o deficiență în detecția de obiecte întunecoase, dezavantaj destul de mare tinându-se seama că este un senzor de distanță.Alt dezavantaj ar mai fi prețul mare.

Avantajele sunt viteza de operare destul de mare și posibilitatea detecției obiectelor în mișcare.

Recomandări ale producătorului sunt să se monteze condesatoare de by-pass intre alimentare si masă pentru a avea o tensiune stabilizată la alimentare și montarea de filtru trece jos pentru obținerea unei valori liniare de tensiune, dar si pentru filtrare de zgomote.

2.7 Regulatorul de tensiune UA7805C

Un regulator de tensiune este proiectat pentru a menține în mod automat un nivel constant de tensiune. Un regulator de tensiune poate fi un simplu "feed-forward" de design sau pot include bucle negative de control feedback. Se poate utiliza un mecanism electromecanic, sau componente electronice. În funcție de design, acesta poate fi folosit pentru a reglementa una sau mai multe tensiuni AC sau DC.

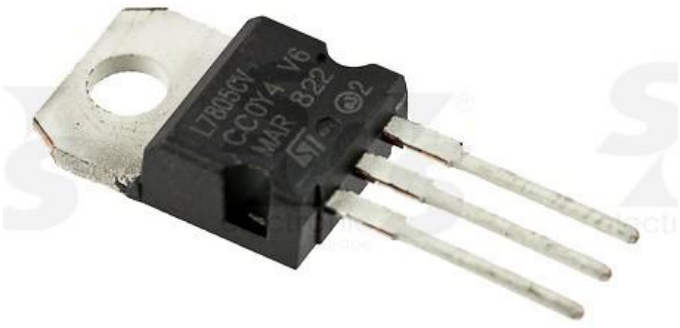


Figura.22 – Regulator 7805

Regulatoare electronice de tensiune se găsesc în dispozitive, cum ar fi sursele de alimentare de calculator pentru a stabiliza tensiuni DC utilizate de către procesor și alte elemente.

Într-un sistem de distribuție a energiei electrice, regulatorul de tensiune poate fi instalat la o substație sau de-a lungul liniilor de distribuție, astfel încât toți clienții primesc constant aceeasi tensiune independent de cât de multa putere trag din linie.

Un regulator de tensiune simplu se poate face dintr-un rezistor în serie cu o diodă (sau o serie de diode). Datorită formei logaritmică de diodei curbe, tensiunea pe dioda schimbă doar ușor datorită modificărilor în curent trase sau la schimbări în intrare. Când nu este important un control de tensiune precis și eficiența, acest design pot lucra bine.

La regulatoare de tensiune feedback-ul funcționează prin compararea tensiunea efectivă de ieșire într-o anumită tensiune de referință fix. Orice diferență este amplificat și utilizat pentru a controla elementul de reglare, astfel încât să se reducă eroarea de tensiune. Aceasta formează o buclă de feedback negativ de control; creșterea câștigul buclă deschisă tinde să crească acuratețea regulamentară, dar reduce de stabilitatea (stabilitatea este evitarea de oscilație, sau de apel, în timpul schimbărilor pasului).

În cazul în care tensiunea de ieșire este prea mică (probabil din cauza tensiunii de intrare reducerea sau încărcarea în curent este în creștere). Elementul de reglementare este poruncit, până la un punct, pentru a produce o putere mai mare de tensiune, prin cădere mai puțină de tensiunea pe intrare sau pentru a trage curent de intrare pentru perioade mai lungi (de tip impuls de comutare de reglementare).Dacă tensiunea de ieșire este prea mare, elementul regulament va fi comandată în mod normal, pentru a produce o tensiune mai mică. Astfel cu acestea cunoscute, multe firme au creat protecție de curent, astfel încât acestea vor opri cu totul aprovizionare curent (sau limitarea curentului într-un fel).Dacă curentul de ieșire este prea mare, unele regulatoare se poate închide.

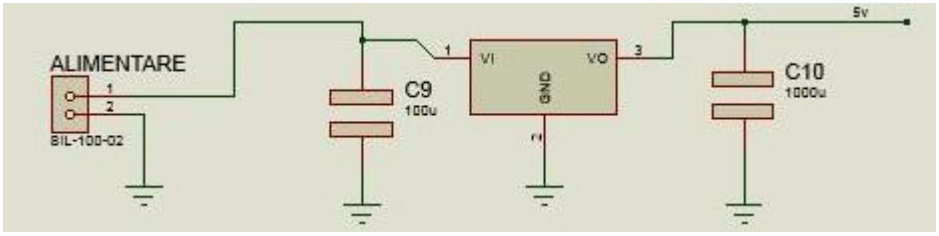


Figura.23 - Schema electrica a circuitului regulator **UA7805C**

Cele două condensatoare sunt electrolitice, și sunt polarizate ambele au borna pozitivă legată de regulator. Primul condensator scoate orice ondulație provenind de la sursă, astfel încât 7805 primește o tensiune de intrare netedă, iar al doilea condensator acționează ca un stabilizator de încărcare pentru a asigura o producție consistentă.

2.8 Interfața neuronală

Într-o rețea neuronală auto-asociativă, stratul de intrare este legat de stratul de ieşire prin intermediul ponderilor asociate. În cadrul procesului de învățare,semnalul aplicat la intrarea rețelei neuronale la un moment dat va fi aceeaşi cu semnalul dorită la ieşirea rețelei. Inițial, setul de ponderi va fi ales aleator iar acesta va fi ajustat de la o epocă la alta pe principiul scăderii erorii obținute între ceea ce se doreşte a se obține la ieşire rețelei neuronale şi ceea ce se obține de fapt prin utilizarea setului curent de ponderi. Procesul de antrenare se va încheia atunci când eroare rețelei va scădea sub un prag acceptat sau când s-a depăşit un anumit număr de epoci de antrenare, setul de ponderi urmând să fie salvat, el reprezentând creierul sistemului.

Din cauza complexității am utilizat o metodă alternativă de creare a unei rețele numită **metoda matricială.**

Se dă formula pentru calculul ponderilor:

Y=X0*W0+X1*W1+...+B

Y – iesirile (motoarele)

X – intrările (senzorii)

W – ponderile

B – pragul (folosit pentru cazul în care senzorii nu reurnează valori)

RR	R	C	L	LL	BL	B	BR	MS	MD	Wo	W1
0	0	0	0	0	0	0	0	1	1	b1=-1	B2=-1
0	0	1	0	0	0	0	0	-1	-1	-2	-2
0	0	0	1	0	0	0	0	0	-1	-1	-2
0	1	0	0	0	0	0	0	-1	0	-2	-1
0	0	0	1	0	0	1	0	0	1	1	-3
0	1	0	0	0	0	1	0	-1	-1	1	-3

2.9 Arduino Bluetooth HC-06

Shiledul Arduino este un circuit integrat care preia informatiile de la microcontroller prin portul serial apoi realizează o conversie in interior ca apoi să transmită sau să primească mai departe datele de la un calculator, telefon sau orice componentă ce are integrat un circuit Bluetooth.

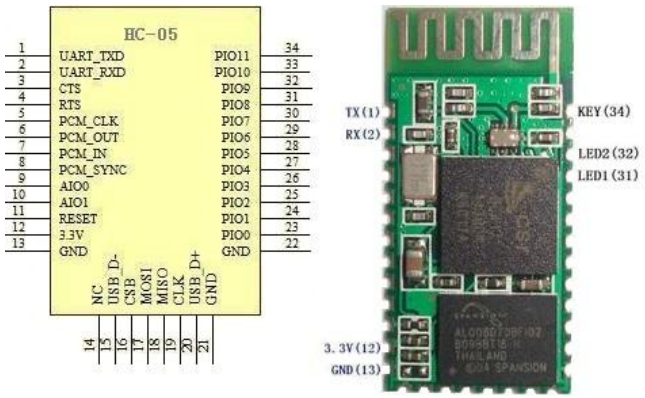


Figura.24 Circuit Arduino Bluetooth HC-06 pinii

Acest circuit se conecteză foarte ușor în diverse aplicații și este foarte ușor de utilizat.Are 4 pini 2 de alimentare 3.3, altul de masă și doi de comunicații RX, TX.

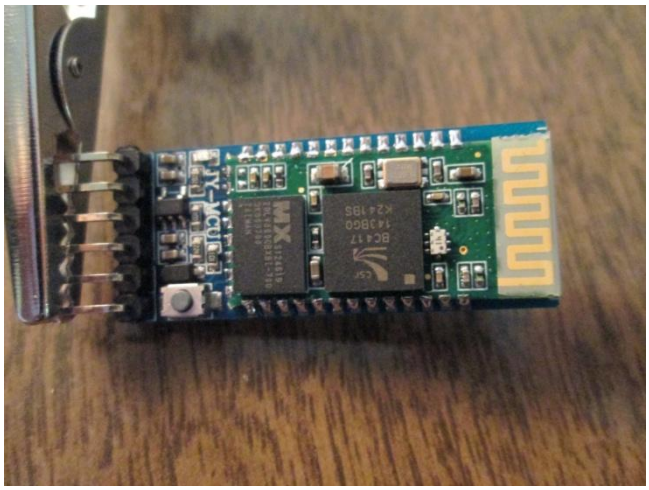


Figura.25 Circuit Arduino Bluetooth HC-06

2.10 Mediul de dezvoltare Freescale CodeWarrior

Freescale CodeWarrior este un program cu ajutorul căruia utilizatorul poate dezvolta

programe pentru microcontrolerele produse de firma FreeScale. Programul se prezintă cu o interfață pentru utilizator tipică unei aplicații Windows, o fereastră cu un meniu în partea superioară de unde utilizatorul poate selecta numeroase opțiuni. Programul conține un editor cu o interfață grafică prietenoasă, un buton de rulare şi un debugger.

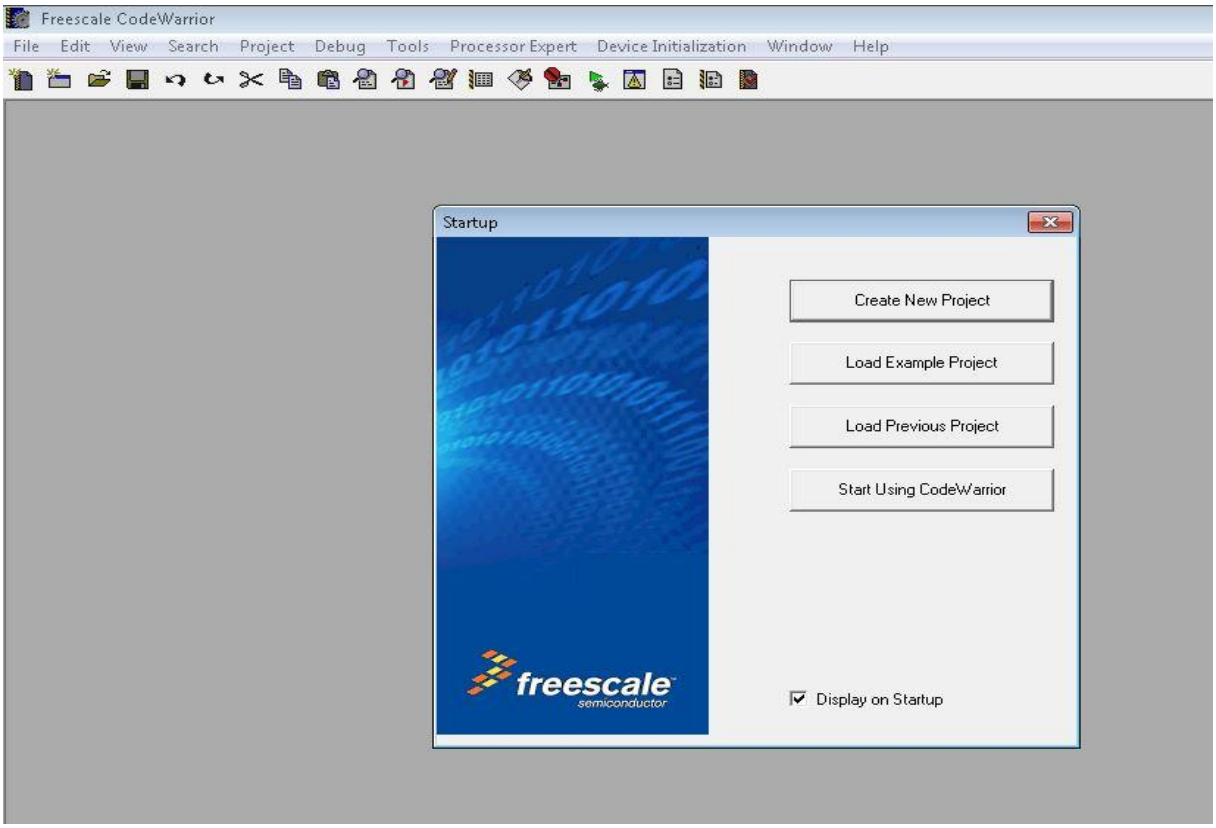
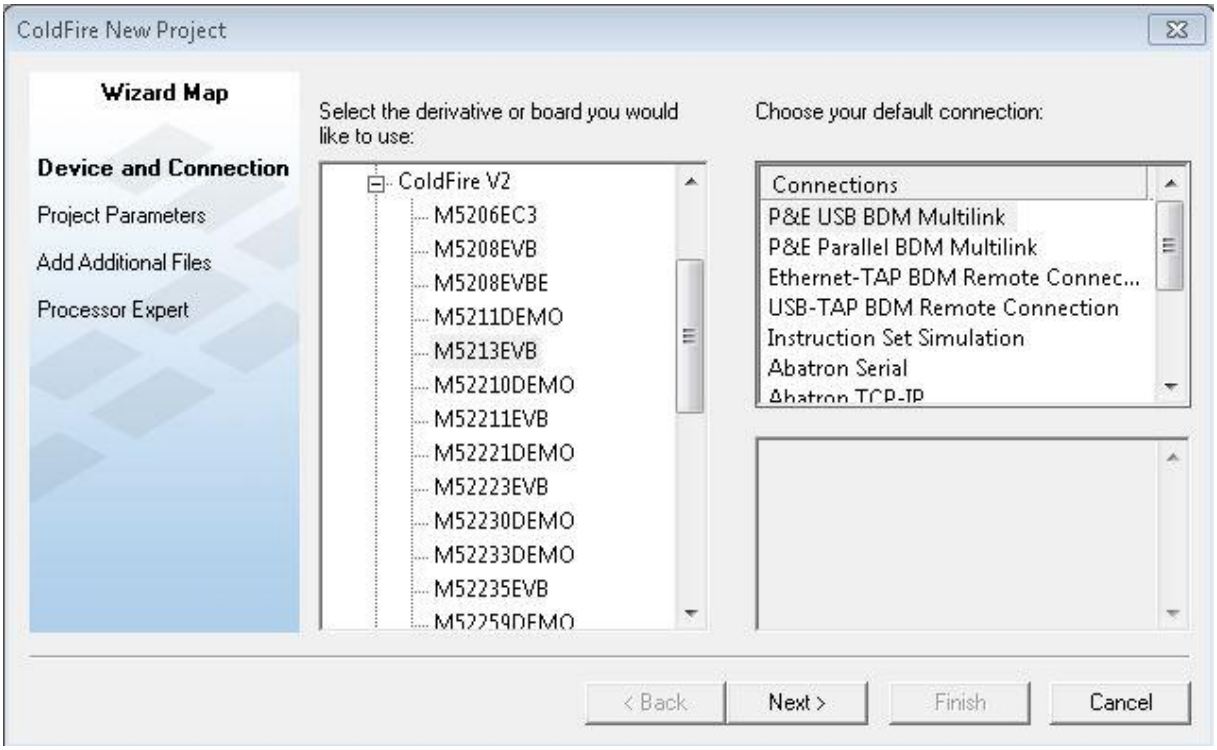


Figura.26 Fereastra principală

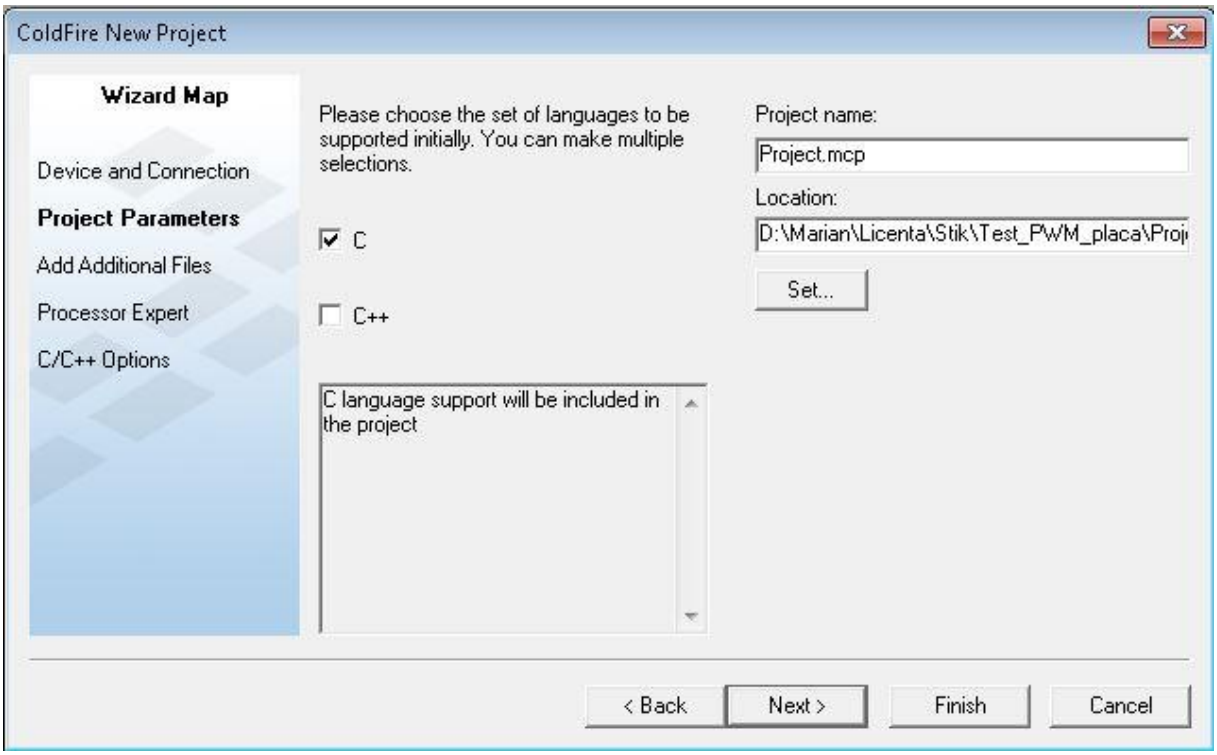
În fereastra principală se observă meniul de start în care se pot alege patru variante și anume:

- Create New Project – Crează un nou proiect
- Load Exaple Project – Programul contine diverse exemple și tutoriale pentru fiecare funcție a microcontrollerelor freescale
- Load Previous Project – Buton pentru încărcarea unui proiect anterior
- Start Using CodeWarrior – Închide meniul principal și il lasă pe utilizator să înceapă de la 0

Pentru a crea un nou proiect putem accesa meniu -> **Create New Project** apare apoi fereastra de mai jos.

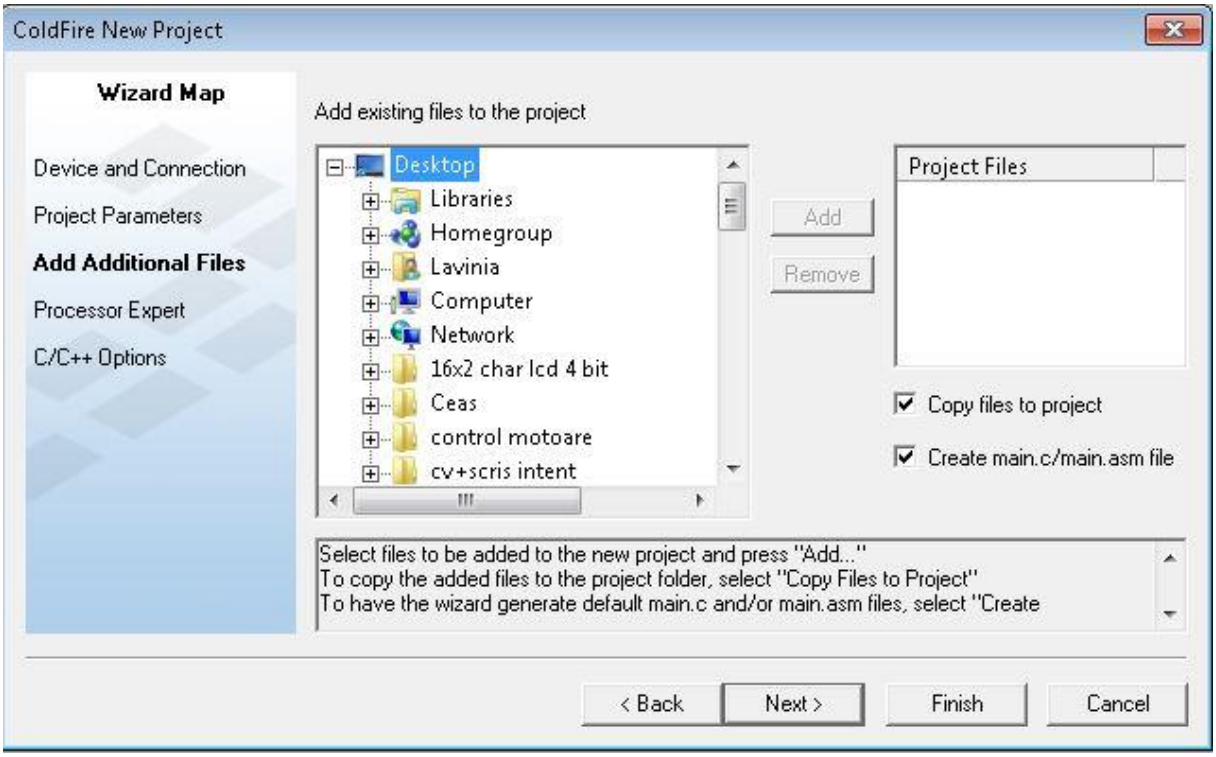


În primul tabel avem toate familiile de microcontrollere care sunt puse la dispoziție de către firma Freescale.Alegem microncotrollerul care se utilizează, apoi se alege tipul de conexiune din tabelul din dreapta apoi se apasă **Next** și apare fereastra următoare**.**

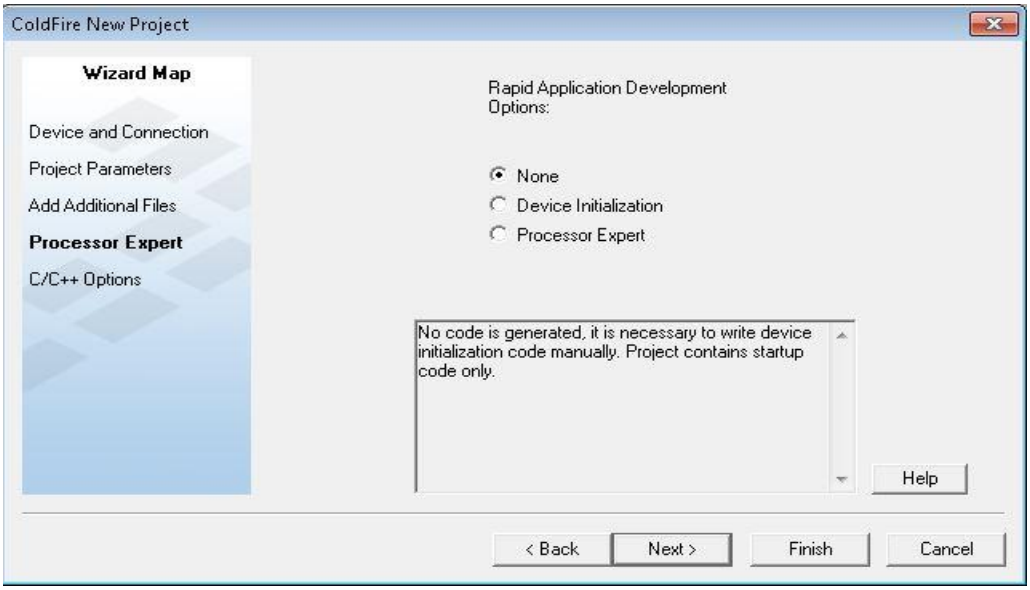


În această fereastră utilizatorul este interogat ce limbaj de programare dorește să folosească. (C sau C++)

În continuare, în aceeași fereastră, se cere să se adauge calea unde se va crea proiectul și numele proiectului apoi se va apăsa butonul **Next.**



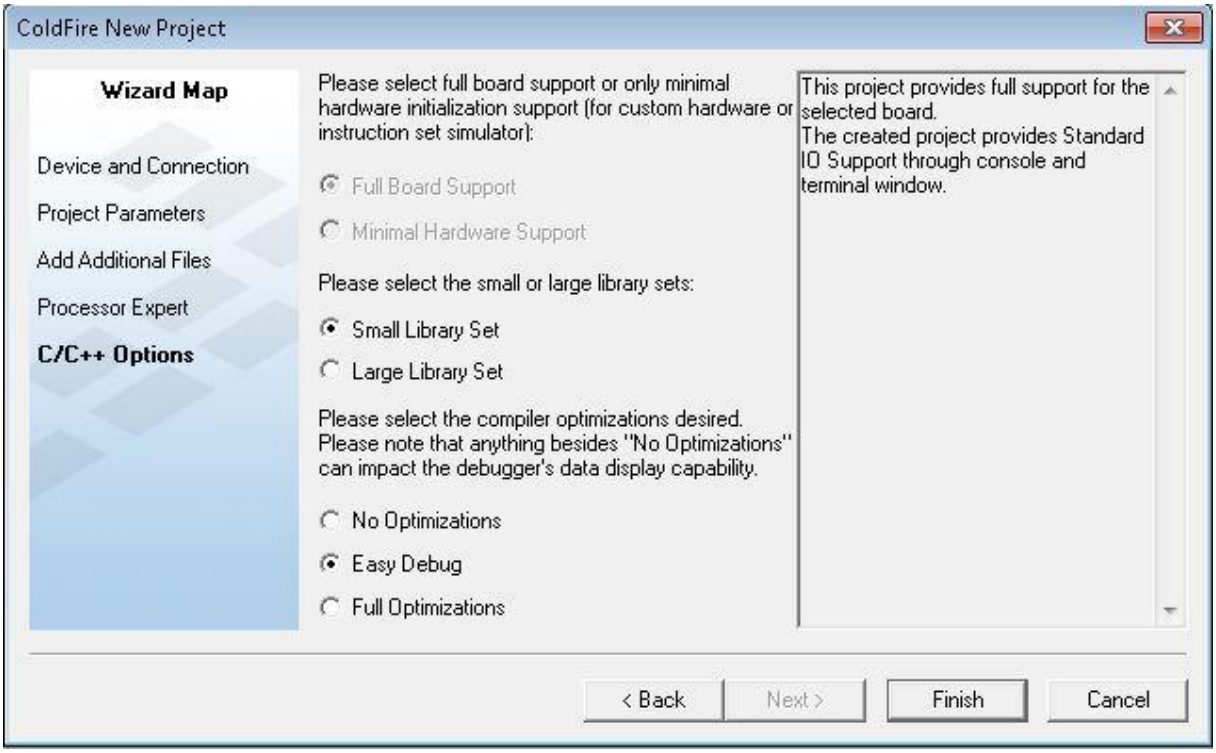
În noua fereastră apărută utilizatorul este întrebat dacă dorește să folosească fișiere deja existente, pe care programul la selectarea acestora le va deschide în noul proiect. Se apasă **Next.**



Un ascpect important al softului Freescale CodeWarrior îl reprezintă această fereastră. Dezvoltatorul a creat un set de aplicații rapide (tooluri) pentru a ușura munca utilizatorului. Astfel avem următoarele opțiuni:

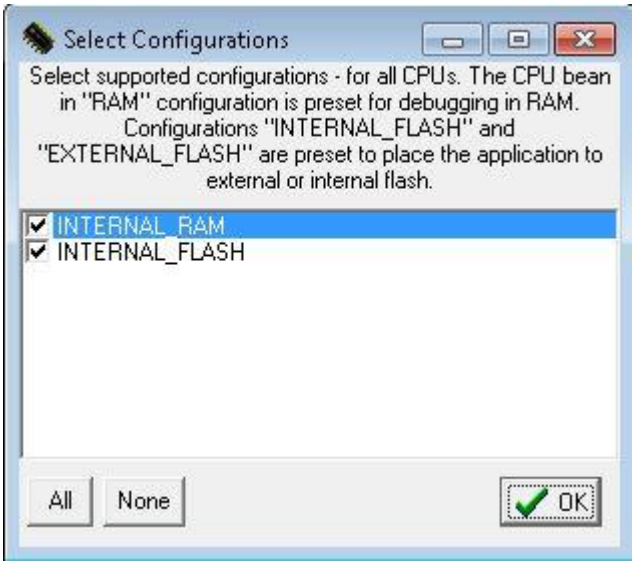
- None – nu se generează nici un cod, este necesar să se scrie manual codul de inițializare al dispozitivului
- Device Inițialization – se generează doar codurile de inițializare
- Processor Expert – toolul generează atât cod de inițializare cât și cod pentru diverse drivere

Se alege și se bifează căsuța Processor Expert apoi se apasă **Next.**



În fereastra de mai sus se selectează tipul de librărie mică sau mare și tipul de Debuging.Se bifeză în funcție de proiect.Apoi se apasă tasta Finish.

În continuare se crează proiectul cu workspace-ul aferent.



Microcontrollerul conține atât un bloc de memorie ram pentru a se realiza cu aceasta un debuging mai rapid, datorită vitezei de lucru, cât un bloc cu memorie flash pentru a încărca softul final.Se apasă **OK.**

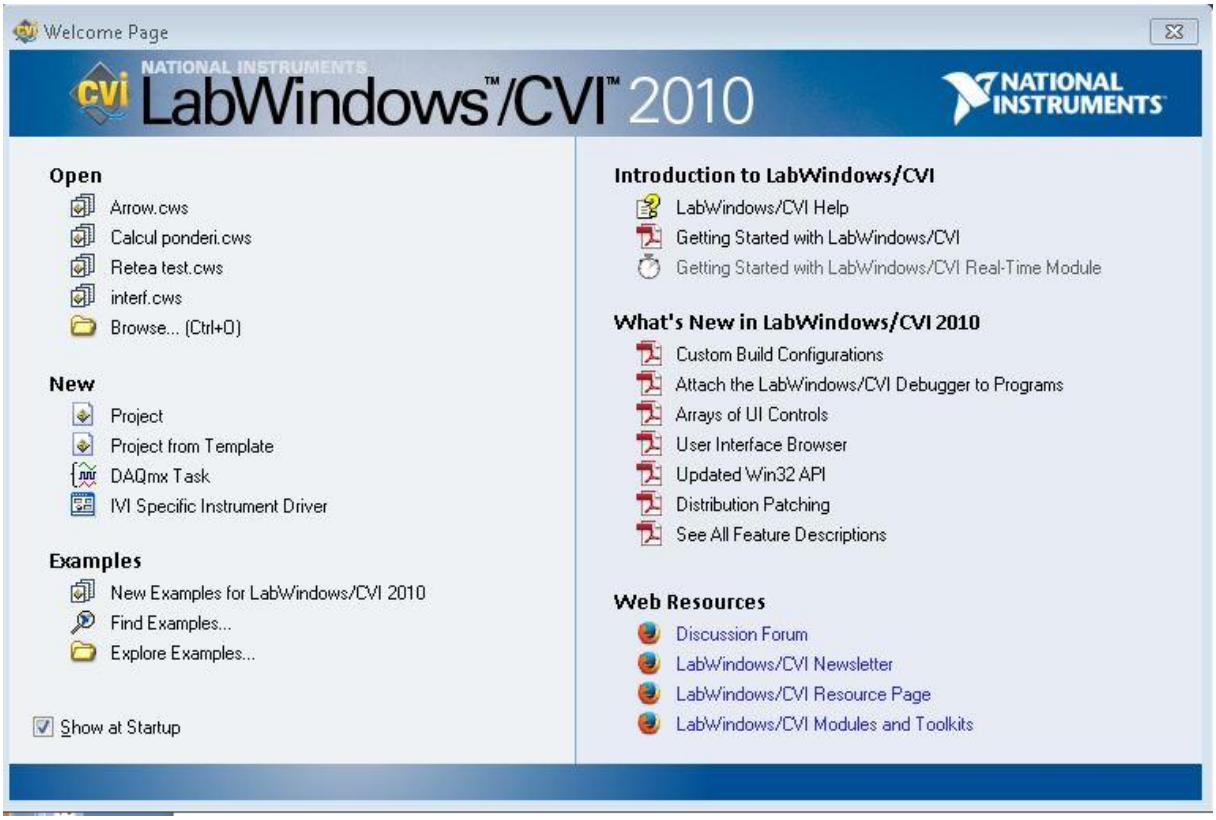
Apare acuma în tabelul din stânga Proiectul.c și se poate trece la lucru.

După scriere unui soft se alege din meniul superior sageata cu gandăcel pentru debuging.Dacă functionează fără vreo problemă atunci codul poate fi încărcat în memoria flash.

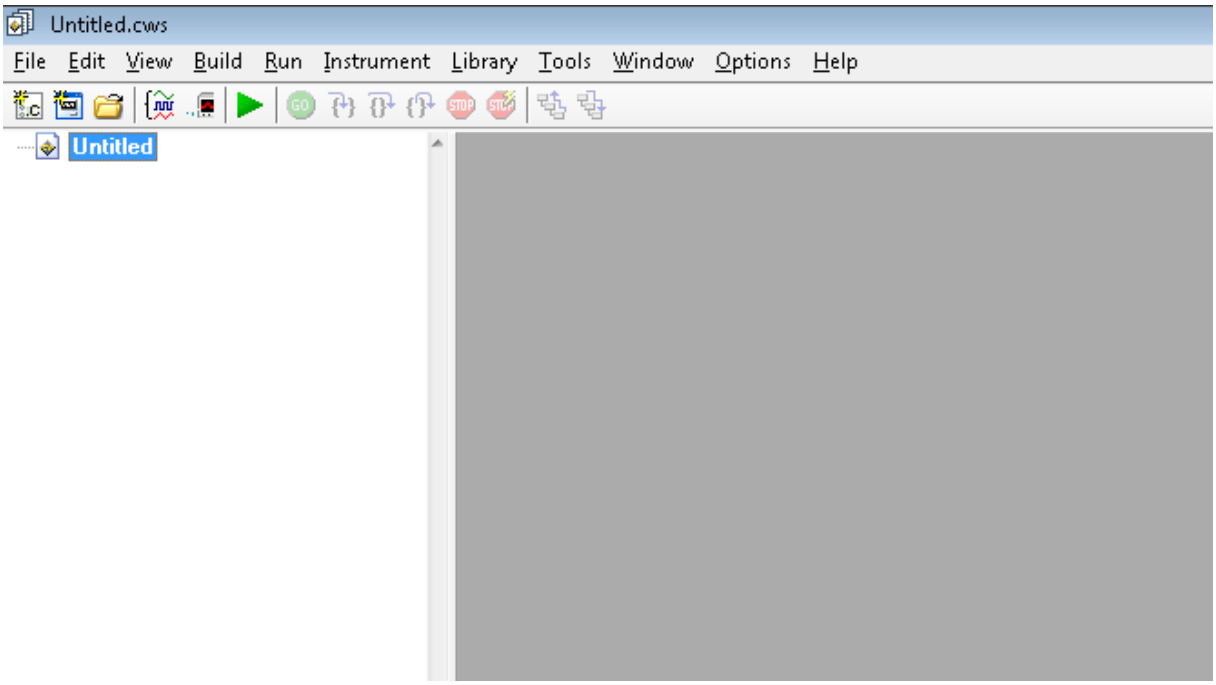
2.11 Mediul de dezvoltare LabWindosCVI

LabWindosCVI este un mediu de programare folosit pentru dezvoltarea unor aplicații pentru sistemele de operare furnizate de către compania Microsoft.Acesta este produs de firma National Instruments Programul vine cu o interfață pentru utilizator prietenoasă, prin intermediul cărora utilizatorul poate accesa cu ușurință resursele puse la dispoziție de acest program.

Chiar de cum lansăm programul în execuție apare o fereastră în care utilizatorul este întrebat dacă doreşte să deschidă o aplicație creată anterior sau dacă doreşte să creeze una nouă sau să utilizeze un exemplu. Pentru a crea un nou proiect programul LabWindosCVI se apasă din lista **New** din partea stângă butonul **Project.**



În continuare apare fereastra principală în care se poate crea workspace sau alceva prin comanda **File ->New...**



Controlul robotului se realizează cu ajutorul acestui program

3. Partea practică - Viziunea de ansamblu asupra proiectului

Proiectul de față prezintă o modalitate de control a unei unități mobile exploratoare cu ajutorul sistemelor embedded. Pentru a atinge obiectivul proiectului, acela de a crea un robot care să învețe să meargă sau folosit o structură electronică pentru controlul motoarelor, cât și o structură pentru preluarea semnalelor de comandă de la senzori.

Unitate mobilă este construită pe un șasiu paralelipipedic, pe care sunt montate patru motoare BDC. Șasiul reprezintă și suport de susținere pentru driverul celor patru motoare, pentru plăcuța de dezvoltare, blocul de comandă motoare, cât și blocul de preluare prelucrare semnale și alimentare.

Robotul lucrează în felul următor.La început se intră pe sistemul de învățare unde din aplicația creată în LabWindowsCVI se transmit comenzi direct motoarelor pentru a deplasa robotul pe un traseu.

În tot acest timp robotul prin intermediul senzorilor și apoi a blocului de prelucrare semnale observă obiectele și salvează în funcție de distanța pană la aceastea limite.

Limite ce sunt transformate în valori binare și anume 1 când coliziunea cu un obstacol este iminentă și 0 când este liber.Aceste limite transformate în valori binare reprezintă doritul idealul pe care robotul trebuie să îl îndeplinească.

La terminarea învățării se comută de pe calculator din soft pe modulul autonom.În cazul acesta utilizatorul nu mai are posibilitatea de a acționa cu robotului decât printr-un buton pentru a selecta iar modul de învățare.

În modul autonom robotul preia valorile de la seznori și le compară cu limitele apoi acestea sunt transmise în funcția de comandă motoare pentru a activa semnalele PWM de comandă.

Se realizează cicli continui în care robotul preia valori de la senzori le verifică cu cele dorite apoi scoate valorile binare pentru funcția de comandă motoare.

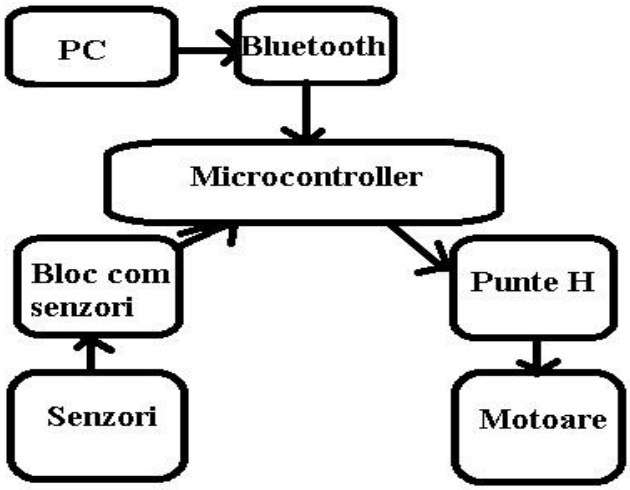


Figura.18 Schema bloc a robotului

Pentru driverul motoarelor s-a folosit circuitul integrat L298, care are 2 punți H, acest lucru înseamnă că ne permite să comandăm 2 motoare BDC. Logica de comandă a circuitului este furnizată de microcontrolerul MCF5213. Circuitul permite si o comandă cu ajutorul semnalelor PWM, furnizate de acelaşi microcontroler amintit anterior. Microcontrollerul primeste informații atât de la calculator prind intermediul transcieverului cât şi de la senzori prin blocul de prelucrare semnale.

3.1.1 Implementarea Hardware

Implementarea hardware se referă la modalitatea prin care s-au realizat struncturile electronice folosite de microntroller.Circuitele electronice au fost realizate prin crearea de cablaje electronice pe care s-au montat componentele electronice.

Un PCB nu este altceva decât un suport mecanic pentru componentele electronice dintr-un circuit. Componentele electronice sunt conectate între ele prin nişte trasee de material conductor, care de cele mai multe ori este din cupru. În funcție de metoda prin care sunt realizate aceste trasee avem PCB-uri obținute prin corodare, prin depunere sau prin frezare mecanică. Metoda corodării plăcii de cupru poate fi aplicată pentru a obține plăci imprimate cu cel mult două straturi de trasee.

Realizarea de cablaje imprimate prin metoda corodării oferă rezultate foarte bune la un preț redus pentru suprafața de circuit imprimat. De obicei această metodă este mai la îndemână decât celelalte două metode anterior menționate.

Prin cablaj înțelegem o placă stratificată care conține cel puțin un strat de material izolator şi cel puțin un strat de material conductor, prin intermediul celui din urmă realizându-se traseele care conectează componentele electronice de pe placă.

Pentru a putea realiza un cablaj prin metoda corodării trebuie mai întâi să imprimăm traseele circuitului pe placă. Traseele se pot proiecta utilizând un program de specialitate cum ar fi Eagle, Proteus sau Cadence.

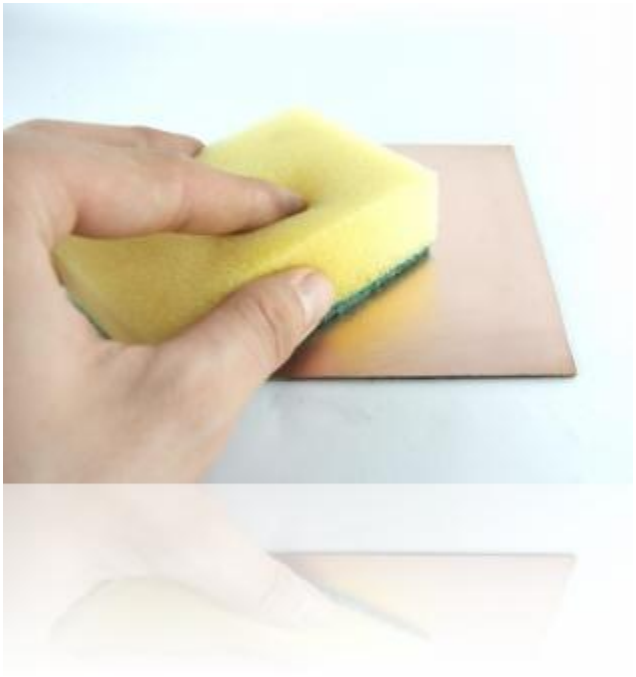
Odată realizată proiectarea traseelor putem trece la imprimarea acestora pe placa de cablaj. Pentru acest lucru putem folosi metoda fotografică sau cea cu transfer termic. Ambele metode ne oferă posibilitatea acoperirii plăcii de cupru, cu un strat rezistent la substanța folosită pentru corodare, pe zonele în care urmează să treacă traseele de conectare a componentelor electronice de pe placă.

Cablajele folosite în cadrul acestui proiect au fost obținute imprimând traseele prin metoda cu transfer termic şi utilizând pentru corodare soluție de clorură ferică. Desi este un proces in multi pasi realizarea unei placi se poate face sub 30 min.

Pași în realizarea unui circuit imprimat sunt:

- Decupați placa la dimensiunile dorite folosind un bonfaier sau un mini circular.

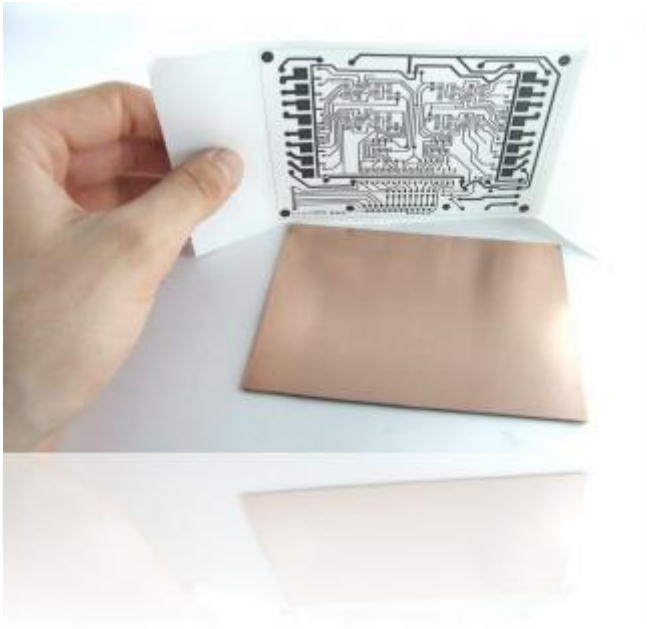
 Lustruiți placa folosind fie hârtie abrazivă cu o granulație mai mare de 600, acesta fiind totuși destul de greu de găsit, o altă soluție fiind folosirea unui burete de vase. Acest pas este foarte important pentru a obtine un rezultat final cât mai bun; cu cât este mai bine lustruită placa cu atat substanța de acoperire este mai lucioasă.



 Curățați cablajul folosind o soluție degresantă, ca de exemplu alcool izotopic, sau spălați placa cu detergent.

 Printați circuitul folosind o imprimată laser, hârtia folosită fiind foarte importantă; cele mai bune rezultate le dă cea cretată cu o densitate mai mare de 120 g/m și cu 2 fețe: una

luciosă și una mată (se găsește destul de greu) dar rezultatele sunt foarte bune; se pot obține ușor trasee de 0.25m.



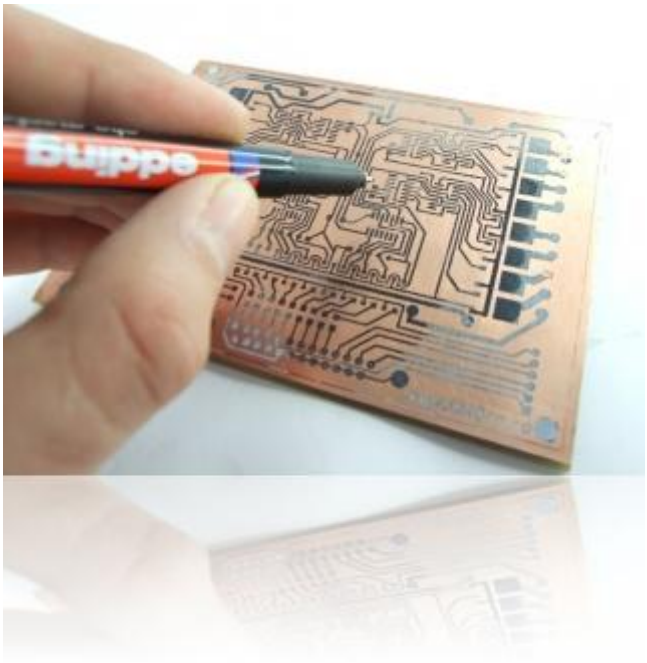
 Decupați circuitul vertical lăsând lateralele mai lungi pentru a putea fi îndoite în spatele placajului pentru o fixare mai bună.

 Folosind un fier de călcat la o temperatură mai mare de 200 grade, apăsați placa aprox. 2 minute, pe urmă frecați apăsând usor folosind o bucata de șervețel; reveniți cu fierul de călcat, frecând usor pentru încă 2 minute.

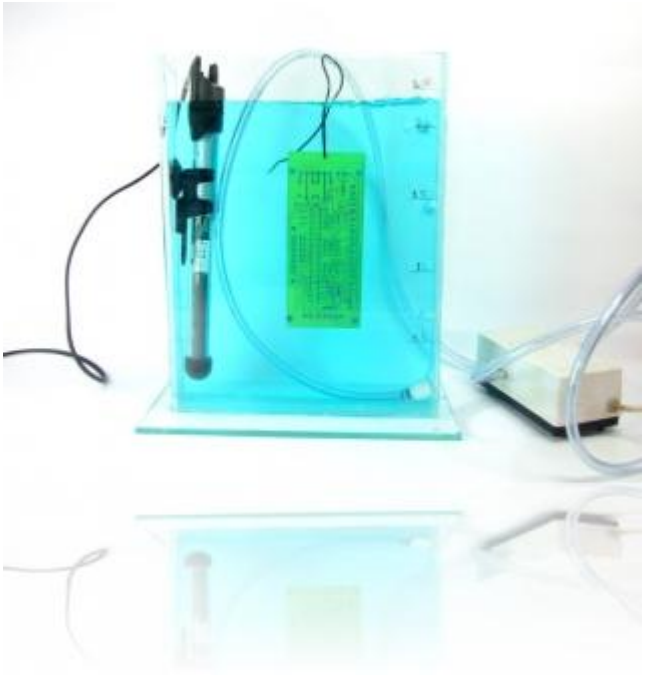
 Odată răcită placa la temperatura camerei, înmuiați-o în apă călduță cu detergent aprox. 5 minute.

 Îndepărtați hârtia, curățați placa folosind o periuța de dinți până când dispare orice urmă de hârtie.

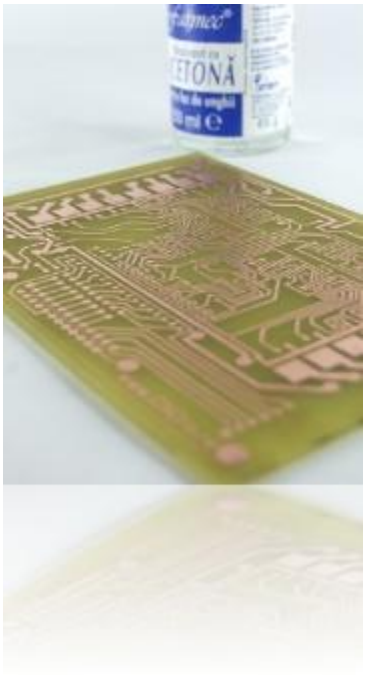
 Refacți eventuale trasee dezlipite folosind un marker rezistent la acid.



 Intoroduceți placa în solutie de corodare - recomand b327 pentru a obține margini mai bine definite + (nu pătează, nu miroase). Încercati pe cât posibil o poziție verticală: aceasta acelerează mult procesul; puteți folosi un “acvariu” .

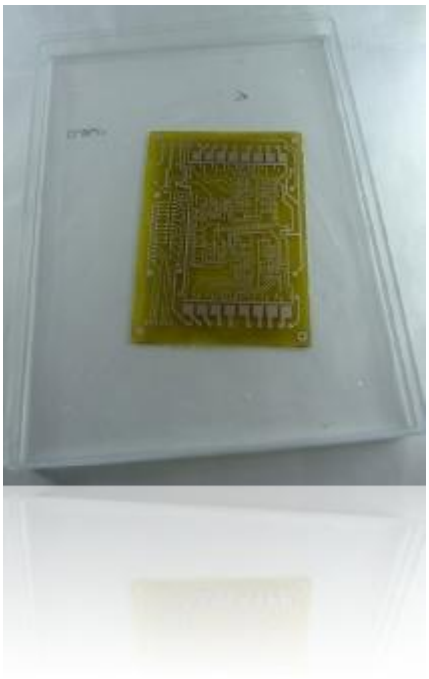


 Curățați placa folosind nitro diluant sau acetonă.



 Verificați la lumina unei lampi eventuale trasee întrerupte sau folosiți un multimetru cu buzeer dacă nu sunteți sigur de continuitatea traseului.

 În acest moment, dacă nu doriți să acoperiți placa și cu stratul de lac protector, puteți introduce placa într-o soluție realizată cu preparatul pentru stanare; după aproximativ 1 min observați magia. O dată acoperită, placa poate fi păstrată pentru perioade lungi de timp fără ca aceasta să oxideze, fiind ușor de cositorit în orice moment.



3.1.2 Circuitul de comandă al motoarelor

Motoarele de curent continuu BDC folosite în acest proiect funcționează la o tensiune de 12 volți și la un curent în sarcină de 400 mA destul de mare.Se spune și că se comandă

două motoare rezultă astfel un consum de 800 mA la mersul în sarcină, din acest motiv nu se pot conecta direct la pinii microntrollerului motoarele.

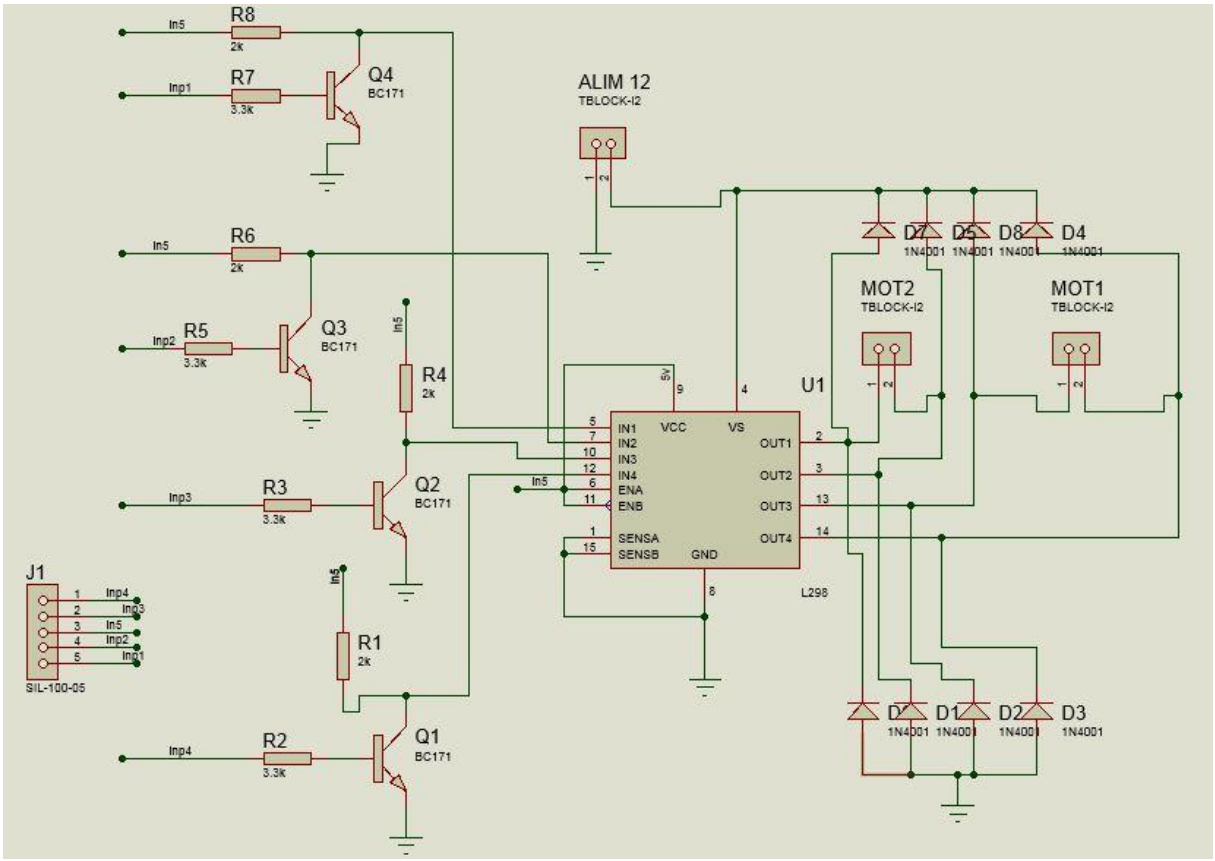
Așa apare necesitatea utilizării unui driver pentru controlul motoarelor, s-a folosit circuitul L298N pnetru a realiza aceasta.

Driverul conține 2 punți H integrate, adică posbilitatea de comandă a două motoare.

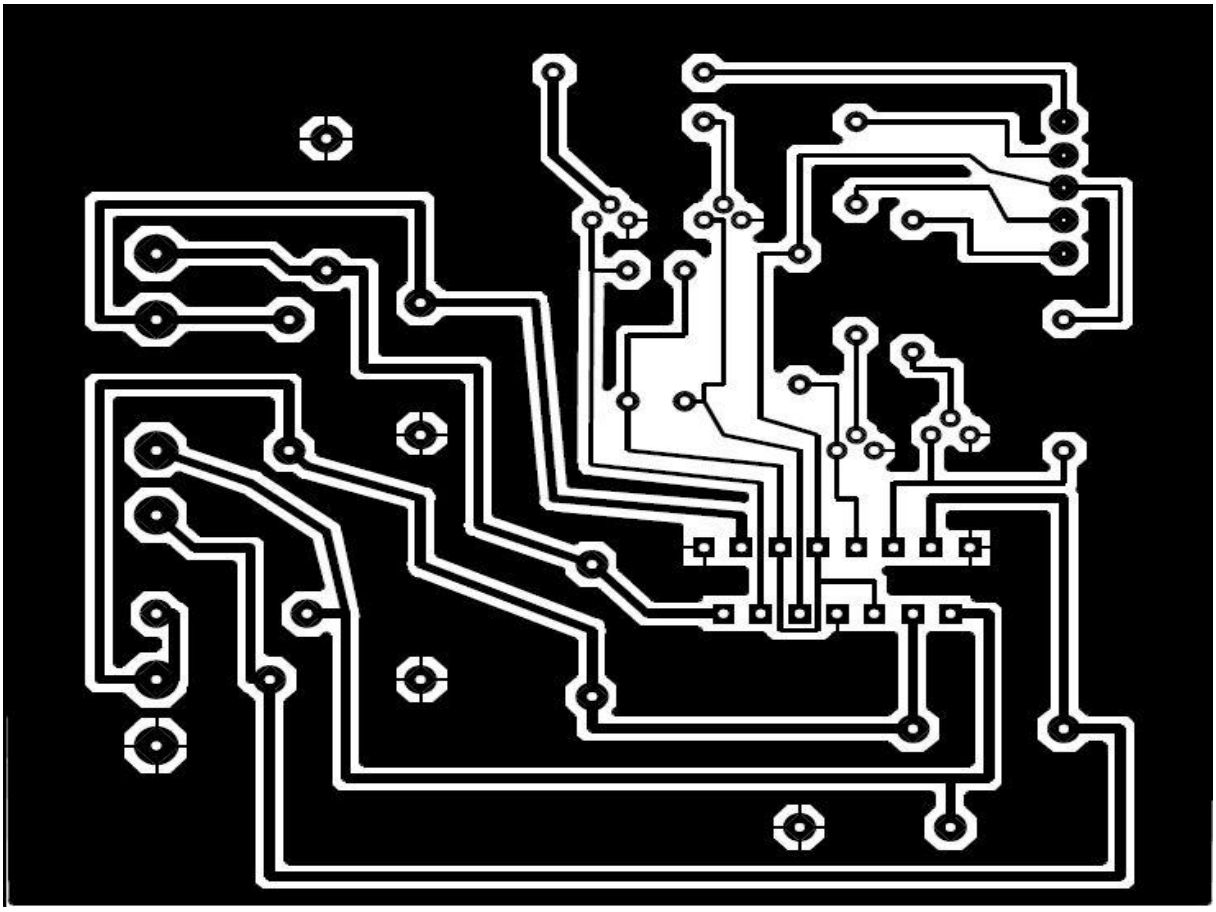
Pentru a comanda driverul se folosește micrcontrollerul M5213EVB.

La rezlizarea cablajului s-a folosit mediul de proiectare ARES inglobat în Proteus.

Schema completă a driverului de comandă pentru motoare este prezentată mai jos.



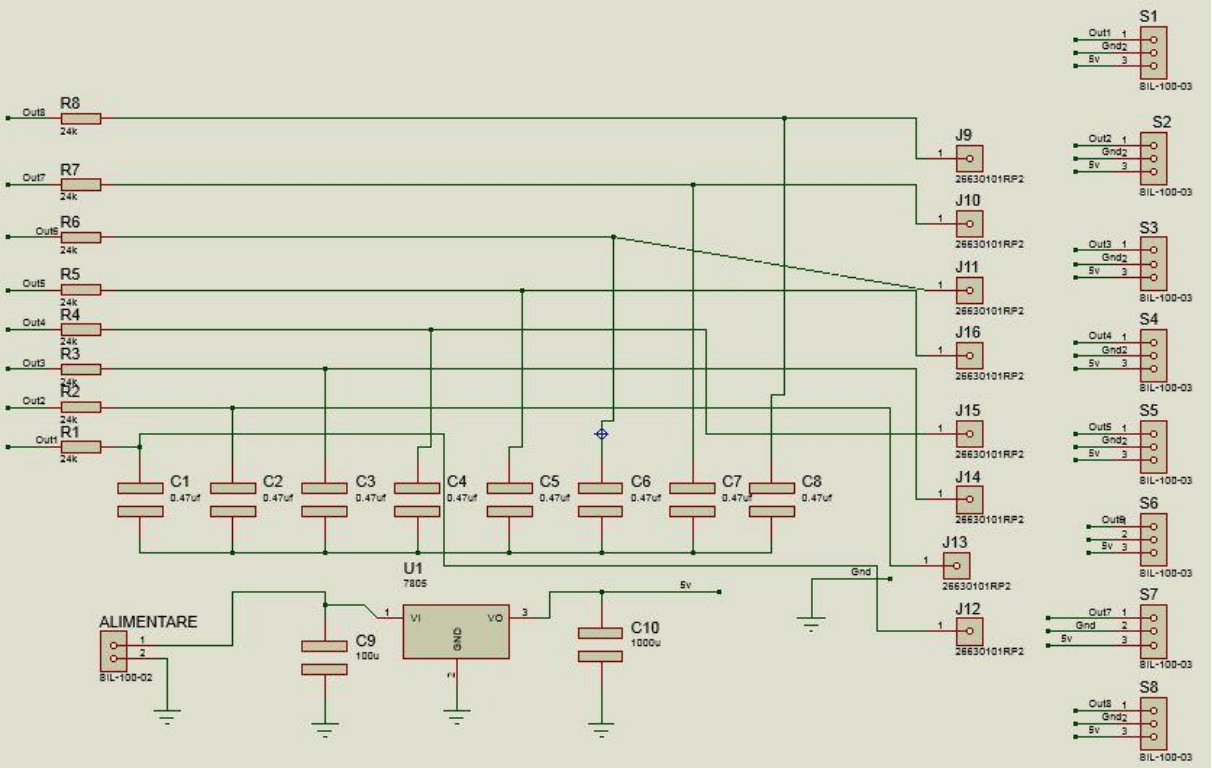
Layout-ul este prezentat mai jos:



3.1.3 Circuitul de alimentare filtrare semnal senzori

Circuitul de alimentare este format din regulatorul UA7805C, care are conectat atât la intrare cât și la ieșire câte un condesator.Condesatoarul de intrare pentru a scoate orice ondulație provenind de la sursă, astfel încât 7805 primește o tensiune de intrare netedă, iar al doilea condensator acționează ca un stabilizator de încărcare pentru a asigura o producție consistentă.

Circuitul de filtrare compus din filtre trece jos elimină zgomotele de la ieșirea de pe senzori.



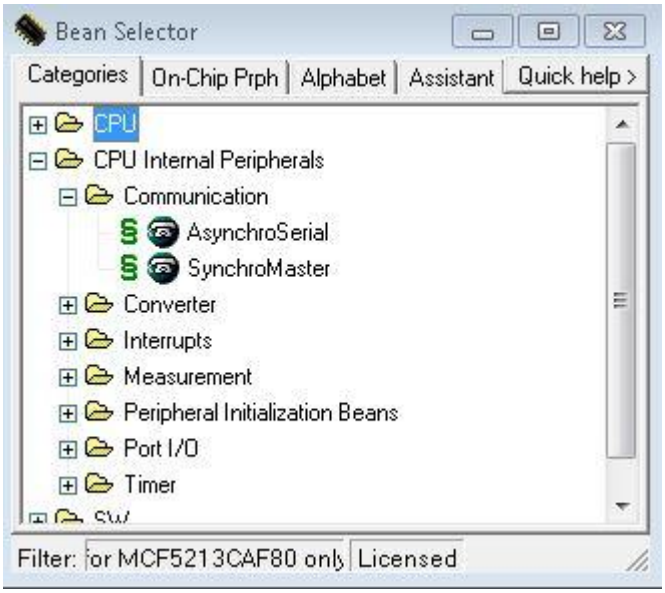
Schema completă de alimentare și prelucrare semnale senzori

3.2.1 Implementarea Software

Soft Microcontroller

Software-ul microcontrolerului a fost dezvoltat cu ajutorul mediului Freescale CodeWarrior microcontrolere din familia MCF52. Programul obținut în urma compilării a fost încărcat în microcontroler prin intermediul programatorului.

De obicei dezvoltarea unui soft pentru un microcontroler începe cu introducerea unei linii de configurare pentru microcontroler. În cazul acestui proiect configurarea microcontrolerului se realizează cu ajutorul opțiunii Bean Selector.**Procesor Expert/View/ Bean selector.**



În continuare se prezintă softul microcontrollerului:

```
char uart_getchar (int channel)
{
    /* Wait until character has been received */
   while (!(MCF_UART_USR(channel) & MCF_UART_USR_RXRDY))
      ;
   return MCF_UART_URB(channel);
}
a=uart_getchar(0);
if(a=='u') l2++;
if(l2%2==0)
mi=1;
else
{
mi=0;
l2=1;
}
```

Functie preluare caracter prin portul serial de la PC plus implementare.

```
//Mod invatare//
while(mi)
{
if(a=='u')mi=0;
a=uart_getchar(0);
   if(a=='w') //a=97
     {
       l1++;
```

```
   setL_engine(l1);
   setR_engine(l1);
    }
else if(a=='a') //a=97
    {
      l1++;
   setL_engine(-l1);
   setR_engine(l1);
    }

else if(a=='s') //a=97
    {
      l1++;
   setL_engine(-l1);
   setR_engine(-l1);
    }

else if(a=='d') //a=97
    {
      l1++;
   setL_engine(l1);
   setR_engine(-l1);
    }
      else
      {if(l1>=0)l1--;
      setL_engine(l1);
      setR_engine(l1);
      }

GetAdcValo(&SenzBL,&SenzB,&SenzBR,&SenzFRR,&SenzFR,&SenzF,&SenzFL,&Senz
FLL);
x0=SenzBL;
x1=SenzB;
x2=SenzBR;
x3=SenzFRR;
x4=SenzFR;
x5=SenzF;
x6=SenzFL;
x7=SenzFLL;
dorit= (x0+x1+x2+x3+x4+x5+x6+x7)>>3;
}
```

Modul de învățare lasă robotul comandat de utilizator și încarcă valorie din senzori in (limite).

```
//Mod autonom//
```

```
a=uart_getchar(0);
if(a=='o') l3++;
if(l3%2==0)
ma=1;
else
{
ma=0;
l3=1;
}
while(ma)
{

GetAdcValo(&SenzBL,&SenzB,&SenzBR,&SenzFRR,&SenzFR,&SenzF,&SenzFL,&Senz
FLL);

y0=ReteaAlternat1(SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL,dorit)
;
y1=ReteaAlternat2(SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL,dorit)
;

setL_engine(y0);
setR_engine(y1);
}
```

Modul autonom este activat după ce au fost încărcate limitele pentru ca robotul să aibă referințe.

```
void GetAdcValo(int *ad0,int *ad1,int *ad2,int *ad3,int *ad4,int *ad5,int *ad6,int *ad7)
{
int data[8][8];




    AD1_Stop();


    AD1_Start();
delayus(2000);
data[0][0]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][0]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][0]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][0]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][0]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][0]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][0]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
```

```
data[7][0]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][1]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][1]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][1]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][1]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][1]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][1]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][1]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][1]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][2]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][2]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][2]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][2]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][2]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][2]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][2]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][2]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][3]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][3]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][3]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][3]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][3]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][3]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][3]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][3]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][4]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][4]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][4]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][4]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][4]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
```

```
data[5][4]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][4]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][4]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][5]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][5]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][5]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][5]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][5]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][5]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][5]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][5]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][6]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][6]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][6]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][6]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][6]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][6]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][6]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][6]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][7]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][7]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][7]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][7]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][7]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][7]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][7]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][7]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
    AD1_Stop();


*ad0=(data[0][0]+data[0][1]+data[0][2]+data[0][3]+data[0][4]+data[0][5]+data[0][6]+data[0]
[7])>>3;
```

```
*ad1=(data[1][0]+data[1][1]+data[1][2]+data[1][3]+data[1][4]+data[1][5]+data[1][6]+data[1]
[7])>>3;

*ad2=(data[2][0]+data[2][1]+data[2][2]+data[2][3]+data[2][4]+data[2][5]+data[2][6]+data[2]
[7])>>3;

*ad3=(data[3][0]+data[3][1]+data[3][2]+data[3][3]+data[3][4]+data[3][5]+data[3][6]+data[3]
[7])>>3;

*ad4=(data[4][0]+data[4][1]+data[4][2]+data[4][3]+data[4][4]+data[4][5]+data[4][6]+data[4]
[7])>>3;

*ad5=(data[5][0]+data[5][1]+data[5][2]+data[5][3]+data[5][4]+data[5][5]+data[5][6]+data[5]
[7])>>3;

*ad6=(data[6][0]+data[6][1]+data[6][2]+data[6][3]+data[6][4]+data[6][5]+data[6][6]+data[6]
[7])>>3;

*ad7=(data[7][0]+data[7][1]+data[7][2]+data[7][3]+data[7][4]+data[7][5]+data[7][6]+data[7]
[7])>>3;
}
```

Funcția ce preia valoare analogică de la fiecare senzor și o convertește digital.

```
void setR_engine (double turatie)
{
   byte tur;

   if ( (turatie < -1.0) || (turatie > 1.0) ) return;
   if (turatie < 0)
      {
   Motor_Dr_PutVal (1);    //pt. mers inapoi
      turatie = -turatie;
      }
   else
      Motor_Dr_PutVal (0);    //pt. mers inainte

   tur = (byte)(turatie*255.0);

   setReg8(PWMDTY5, tur);
}
```

Funcție de comandă motor dreapta este la fel ca cea de comandă motor stânga.

```
int ReteaAlternat1(int x0,int x1,int x2,int x3,int x4,int x5,int x6,int x7,int dorit)
{
```

```
int WInput_n[8],y0=0,o=0,x[7],w=0,i[7];

WInput_n[w]=-1;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=2;
w++;
WInput_n[w]=3;
w++;
WInput_n[w]=2;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=-2;
w=0;

x[o]=x0;
o++;
x[o]=x1;
o++;
x[o]=x2;
o++;
x[o]=x3;
o++;
x[o]=x4;
o++;
x[o]=x5;
o++;
x[o]=x6;
o++;
x[o]=x7;

    for(o=0;o<=7;o++)
  {
  if(x[o]>dorit)
  i[o]=1;
  else
  i[o]=0;
  y0+=WInput_n[o]*i[o]-1;


}
return y0; }
```

Rețeaua ce comandă ieșirile pe motoare.Implementează exact pași inverși folosindu-se de ponderile calculate în tabel.

3.2.2 Soft LabWindowsCvi Calcul ponderi

```
GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON, &v[i]); //0
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_2, &v[i]); //
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_3, &v[i]);
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_4, &v[i]);
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_5, &v[i]);
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_6, &v[i]);
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_7, &v[i]);
            i++;
            GetCtrlVal (panelHandle, PANEL_TOGGLEBUTTON_8, &v[i]);
            GetCtrlVal (panelHandle, PANEL_NUMERIC_3, &m0);
            GetCtrlVal (panelHandle, PANEL_NUMERIC_4, &m1);
```

Se preiau valorile de la fiecare intrare motoare/senzori.

```
for(i=0;i<=7;i++)
    {
      if(v[i]==1)
      {
      a[n]=(m0-c)/v[i];
      w1=a[n];
      n++;
      c=0;
      }
        else
        {
          a[n]=0;
            n++;
        }
    }
    for(n=0;n<=7;n++)
    {
    ponda=ponda+a[n];
    }
      //w1= a[n];
    //w0=ponda-c;
    w0=ponda;
```

```
c=-1;
//ponderi 2
   for(i=0;i<=7;i++)
{
   if(v[i]==1)
   {
   b[m]=(m1-c)/v[i];
   w1=b[m];
   m++;
   c=0;
   }
      else
      {
         b[m]=0;
            m++;
      }
}
for(m=0;m<=7;m++)
{
pondb=pondb+b[m];
}
    //w1= b[m];
w1=pondb;    //-c
b[m]=0;
      SetCtrlVal (panelHandle, PANEL_NUMERIC, w0);
      SetCtrlVal (panelHandle, PANEL_NUMERIC_2,w1);
```

Se parcurg toate intrările și în funcție de aceastea se calculează separat pentru fiecare situație din tabel ponderile.

3.2.3 Soft LabWindowsCvi Interfața de comunicare

```
void SetConfigParms (void)
{
   SetCtrlVal (config_handle, CONFIG_COMPORT, comport);
   SetCtrlVal (config_handle, CONFIG_BAUDRATE, baudrate);
   SetCtrlVal (config_handle, CONFIG_PARITY, parity);
   SetCtrlVal (config_handle, CONFIG_DATABITS, databits);
   SetCtrlVal (config_handle, CONFIG_STOPBITS, stopbits);
   SetCtrlVal (config_handle, CONFIG_INPUTQ, inputq);

   SetCtrlIndex (config_handle, CONFIG_COMPORT, portindex);
}
```

Funcție pentru configurare port serial.

```
int CVICALLBACK fBINARYSWITCH (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{        int binary0;
```

```
   switch (event)
   {
      case EVENT_COMMIT:
         GetCtrlVal (panel, SERIAL_BINARYSWITCH, &binary0);
               if(binary0==1)
               {
         GetCtrlAttribute (panel, SERIAL_Autonom,
         ATTR_CTRL_VAL, buffer);
      stringsize = StringLength (buffer);  //send_data
   bytes_sent = ComWrt (comport, buffer, stringsize);

      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 1); }
               else
      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 0);
         break;
   }
   return 0;
}
```

Funcție care preia valoarea dintr-un siwtch de pe interfața apoi il transmite pentru seta modul de funcționare al robotului.

```
int CVICALLBACK fTasta (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{

   static int last_character;
   static char nou_char;  //0 nou caracter, 1 vechi caracter
   static time_t t_noucharacter;
   time_t t_acum;
   char buffer[3];
   switch (event)
   {
      case EVENT_KEYPRESS:
          // important aici am receptie si transmitere tasta//
         SetCtrlAttribute (panel, SERIAL_TBOX_SEND, ATTR_CTRL_VAL, "");
         character = GetKeyPressEventCharacter  (eventData2);
            memset (buffer, 0, sizeof (buffer));
            CmbSetC (buffer, character);

              SetCtrlAttribute (panel, SERIAL_ORIGINAL,
                  ATTR_CTRL_VAL, buffer);

             stringsize = StringLength (buffer);  //send_data
   bytes_sent = ComWrt (comport, buffer, stringsize);
   //      SetCtrlVal (panel, SERIAL_TBOX_SEND, "");
```

```
//  GetCtrlVal (panel_handle, SERIAL_TBOX_SEND, send_data);
  //  SendAscii ();


RS232Error = ReturnRS232Err ();
if (RS232Error)
   DisplayRS232Error ();
SetCtrlAttribute (panel_handle, SERIAL_BYTES, ATTR_DIMMED, 0);
// SetCtrlVal (panel_handle, SERIAL_BYTES, bytes_sent);

   if (character != 0)
   {
      if (last_character != character)
         {
         nou_char = 0;
         last_character = character;
         t_noucharacter = clock ();

         valw = 0;
         vala = 0;
         vals = 0;
         vald = 0;
         }
      else
         {
         nou_char = 1;
         t_acum = clock ();

         if ( t_acum > (t_noucharacter + 100) )
            {
               if(character==119)
               {

                  valw += 0.01;
                  if  (valw > 1) valw = 1;
                  SetCtrlVal (panel, SERIAL_NUMERIC, valw);
                  t_noucharacter = t_acum;
               }
               if(character==97)
               {

                  vala += 0.01;
                  if  (vala > 1) vala = 1;
                  SetCtrlVal (panel, SERIAL_NUMERIC_2, vala);
```

```
                t_noucharacter = t_acum;
            }
            if(character==115)
            {

                vals += 0.01;
                if  (vals > 1) vals = 1;
                SetCtrlVal (panel, SERIAL_NUMERIC_3, vals);
                t_noucharacter = t_acum;
            }
            if(character==100)   //d
            {

                vald += 0.01;
                if  (vald > 1) vald = 1;
                SetCtrlVal (panel, SERIAL_NUMERIC_4, vald);
                t_noucharacter = t_acum;
            }
        }

    }
}


    break;
  }
  return 0;
}
```

Se scrie o tasta în căsuță, tastă ce va fi transmisă pe portul serial pentru realizarea comenzii motoarelor.

4. Rezumat

Scopul lucrării de față a fost dezvoltarea unei aplicații care, utilizând concepte de interfațare om-calculator şi respectiv sisteme embedded, să uşureze sarcina omului în diverse activități. Unitatea mobilă poate fi controlată pe raza de bătaie a unei rețele bluetooth. În cazul lipsei semnalului, se poate activa modul autonom iar aceasta va găsi singură calea de întoarcere.

Ideea proiectului a pornit de la necesitatea de a integra cunoştințele acumulate în cei patru ani într-un proiect cât mai complet care să corespundă cât mai multor materii parcurse în tot acest timp.

În această lucrare se prezintă pași parcurși pentru a crea un robot autonom cât mai simplu posibil.

În prima parte se prezintă noțiunile teoretice necesare pentru realizarea proiectului.

În partea a doua se relevă transpunerea cunoştințelor teoretice în practică.

Inițial, se prezintă diverse noțiuni despre motoarele de curent continuu cu perii colectoare, modul de comandă al acestora cu ajutorul integratului L298N și al microcontrollerului M5213EVB. Microcontrollerul comunică serial cu PC-ul și primește comenzile pentru a antrena structura de învățare apoi, după comutarea pe modul autonom, microncontrollerul comandă motoarele în funcție de datele achiziționate. Se prezintă și aspecte legate de comunicarea serială datorită comunicării de acest tip cu PC-ul. Apoi se arată detalii despre PWM, pentru că numai așa poate microcontrollerul să comande puntea H.

Dat fiind faptul că pentru funționarea robotului, cât și pentru comunicarea acestuia cu calculatorul, au fost necesare nişte coduri și aplicații software, se prezintă pe scurt mediile de dezvoltare Freescale CodeWarrior și LabWindows CVI în care au fost create aplicațiile.

Partea în care se prezintă implementarea hard începe cu descrierea metodei prin care s-a realizat unul dintre cablaje pentru circuitul de comandă al motoarelor.

După prezentarea hard-ului s-a prezentat și partea soft realizată pentru M5213EVB, Interfața de comandă a robotului și Calculatorul ponderilor.

Descrierea codului pentru programul care rulează pe microcontroler s-a făcut, pentru o mai bună înțelegere, în paralel cu descrierea perifericelor utilizate.

Prin îmbinarea noțiunilor teoretice cu abilitățile practice s-a implementat cu succes unitatea mobilă exploratoare.

5. Concluzii

Practica diferă mult de teorie deoarece multe dintre aspectele teoretice se leagă ușor sau se pot idealiza, cât despre practică se tinde spre idealizare, ceea ce este destul de greu.

Dacă la început proiectul parea o jucărie lucrând la el și întâmpinând diverse obstacole unele mai ușor de trecut altele mai grele îți dai seama ca nu este chiar așa mai ales când scopul final al proiectului este sa copie o caracteristică umană, fundamentală și anume adaptivitatea.

Primele probleme au apărut la partea hard în încercarea de a crea un PCB cât mai clar posibil a trebuit să merg pe încercări succesive, ca într-un final să iasă plăcuța finală care acuma merge ca unsă.

Altă problemă la partea hard a fost când s-a ars un trazistor de pe placă, lucru greu de identificat, dar până la urmă prin încercări succesive și mi-am dat seama că era tranzistorul și nu altceva.

Atât înțelegerea cât și proiectarea unei rețele neuronale sunt dificil de realizat.

O problemă majoră a fost găsirea unei soluții pentru a realiza înlocuirea rețelei neuronale, soluție care a venit tot din partea aceasta dar cu diferențe substanțiale față de o rețea adevărată.

Proiectul sigur va fi continuat la master, posibilitățile unei rețele neuronale sunt infinite și doresc să aprofundez aceste cunoștințe.

Prin realizarea acestui proiect am reușit să integrez mai bine cunoștințele acumulate în decursul celor patru ani de studiu și să îmi dezvolt aptitudini inginerești, care cu siguranță îmi vor fi de folos în cariera pe care doresc să o urmez.

6.Anexe

6.1 Fisierul main.c ColdFire

```
/** ###################################################################
**     Filename  : Leduri_si_serial.C
**     Project   : Leduri_si_serial
**     Processor : MCF5213CAF80
**     Version   : Driver 01.00
**     Compiler  : CodeWarrior MCF C Compiler
**     Date/Time : 25.04.2014, 20:29
**     Abstract  :
**         Main module.
**         This module contains user's application code.
**     Settings  :
**     Contents  :
**         No public methods
**
** ###################################################################*/
/* MODULE Leduri_si_serial */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "AS1.h"
#include "AD1.h"
#include "Motor_Dr.h"
#include "Motor_Stg.h"
#include "Pwm1.h"
//#include "Motor_Stg.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include "Serial.h"
/*Include modul comunicatie*/
#include "CC1k.h"
#include "Functii.h"
#include "Retea1Strat.h"

#include "MCF5213.h"
#include "MCF5213_DTIM.h"

#include<stdio.h>

double turat,ab;
char a;
byte val1;
word val;
int i,l1,l2=1,l3=1,l4,tens,dist,x[8],y[8],n,v,o,vo,g=30,y0,y1,dorit,mi,ma;
int SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL;
int lol,x0,x1,x2,x3,x4,x5,x6,x7;

#define BUFFER_SIZE 15

volatile bool blockReceived = FALSE; /* ISR flag */
double convIR_01 (int val);

                      /* number of received chars */
```

```
void main(void)
{

//setR_engine(turat);
//AD1_Init();
Pwm1_Init();

//i=0.1;
//l1=l2=l3=l4=0;
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!!
***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.
  /* Write your code here */
    /**** Test comunicatie seriala ****/
    stopMotoare();
while(1)
{
a=uart_getchar(0);
if(a=='u') l2++;
if(l2%2==0)
mi=1;
else
{
mi=0;
l2=1;
}
//Mod invatare//
while(mi)
{
if(a=='u')mi=0;
a=uart_getchar(0);
    if(a=='w') //a=97
      {
        l1++;
    setL_engine(l1);
    setR_engine(l1);
     }
else if(a=='a') //a=97
      {
        l1++;
    setL_engine(-l1);
    setR_engine(l1);
     }

else if(a=='s') //a=97
      {
        l1++;
    setL_engine(-l1);
    setR_engine(-l1);
     }

else if(a=='d') //a=97
      {
        l1++;
    setL_engine(l1);
    setR_engine(-l1);
     }
        else
        {if(l1>=0)l1--;
```

```
        setL_engine(l1);
        setR_engine(l1);
        }

GetAdcValo(&SenzBL,&SenzB,&SenzBR,&SenzFRR,&SenzFR,&SenzF,&SenzFL,&SenzFLL)
;
x0=SenzBL;
x1=SenzB;
x2=SenzBR;
x3=SenzFRR;
x4=SenzFR;
x5=SenzF;
x6=SenzFL;
x7=SenzFLL;
dorit= (x0+x1+x2+x3+x4+x5+x6+x7)>>3;
}
//Mod joaca//
a=uart_getchar(0);
if(a=='o') l3++;
if(l3%2==0)
ma=1;
else
{
ma=0;
l3=1;
}
while(ma) //Mod autonom//
{

GetAdcValo(&SenzBL,&SenzB,&SenzBR,&SenzFRR,&SenzFR,&SenzF,&SenzFL,&SenzFLL)
;

y0=ReteaAlternat1(SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL,d
orit);
y1=ReteaAlternat2(SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL,d
orit);

setL_engine(y0);
setR_engine(y1);
}

}
  /*** Don't write any code pass this line, or it will be deleted during
code generation. ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/
//auxiliare
//sprintf(bufText,"\r SenzBL= %d SenzB= %d ,SenzBR= %d ,SenzFRR= %d
,SenzFR= %d ,SenzF= %d ,SenzFL= %d ,SenzFLL= %d
",SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL);// %d.%02d
//a/100,a%100,dist
//  SendBuffTxt (bufText, 0);
//ReteaAlternat(SenzBL,SenzB,SenzBR,SenzFRR,SenzFR,SenzF,SenzFL,SenzFLL,x0,
x1,x2,x3,x4,x5,x6,x7);
//sprintf(bufText,"\r SenzBL= %d SenzB= %d ,SenzBR= %d ,SenzFRR= %d
,SenzFR= %d ,SenzF= %d ,SenzFL= %d ,SenzFLL= %d
",x0,SenzBL,x2,x3,x4,x5,x6,x7);// %d.%02d //a/100,a%100,dist
//  SendBuffTxt (bufText, 0);
```

Fisierul Functii.h

```
#ifndef __Functii__
#define __Functii__
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
#include <stdio.h>
#include "MCF5213.h"
#include "MCF5213_DTIM.h"


/*Include modul comunicatie*/
#include "CC1k.h"
#define SYSTEM_CLOCK_KHZ    (80000)
static uint8 state = 0x80;
char bufText[200];
/****** COMANDA LED-uri ******/
void initLEDs (void);
void LED1 (char stareLED);          //stareLED = 1 - aprinde     LED-ul
corespunzator pozitie in binar
void LED2 (char stareLED);          //stareLED = 0 - stinge
void LED3 (char stareLED);
void LED4 (char stareLED);
void leds_display(uint8 number);
/****** delay *******/
void delayus(unsigned long usecs);
//prototip conversie
double convIR_01 (int val);
void initLEDs (void)
{
    if (state & 0x80)        //verific ca MSB a variab "state" este in 1
            {                //doar daca da atunci pun pe zero LED-urile
            leds_display (0);
            state = 0;
            }
}

void LED1 (char stareLED)
{
    if (state & 0x80) {leds_display (0); state = 0;}

    if (stareLED)
        state |= 0x01;
    else
        state &= 0xFE;

    leds_display(state);
}

void LED2 (char stareLED)
{
    if (state & 0x80) {leds_display (0); state = 0;}

    if (stareLED)
        state |= 0x02;
    else
        state &= 0xFD;

    leds_display(state);
}
```

```
void LED3 (char stareLED)
{
    if (state & 0x80) {leds_display (0); state = 0;}

    if (stareLED)
        state |= 0x04;
    else
        state &= 0xFB;

    leds_display(state);
}

void LED4 (char stareLED)
{
    if (state & 0x80) {leds_display (0); state = 0;}

    if (stareLED)
        state |= 0x08;
    else
        state &= 0xF7;

    leds_display(state);
}

void leds_display(uint8 number)
{
    /* Enable signals as GPIO */
    MCF_GPIO_PTCPAR = 0
        | MCF_GPIO_PTCPAR_DTIN3_GPIO
        | MCF_GPIO_PTCPAR_DTIN2_GPIO
        | MCF_GPIO_PTCPAR_DTIN1_GPIO
        | MCF_GPIO_PTCPAR_DTIN0_GPIO;

    /* Set output values */
    MCF_GPIO_PORTTC = number;

    /* Enable signals as digital outputs */
    MCF_GPIO_DDRTC = 0
        | MCF_GPIO_DDRTC_DDRTC3
        | MCF_GPIO_DDRTC_DDRTC2
        | MCF_GPIO_DDRTC_DDRTC1
        | MCF_GPIO_DDRTC_DDRTC0;
}
void delayus(unsigned long usecs)
{
    //DMA Timer 3 - pornire
    MCF_DTIM3_DTRR = (usecs - 1);
    MCF_DTIM3_DTER = MCF_DTIM_DTER_REF;
    MCF_DTIM3_DTMR = 0
        | MCF_DTIM_DTMR_PS(SYSTEM_CLOCK_KHZ / 1000)
        | MCF_DTIM_DTMR_ORRI
        | MCF_DTIM_DTMR_FRR
        | MCF_DTIM_DTMR_CLK_DIV1
        | MCF_DTIM_DTMR_RST;

    while ((MCF_DTIM3_DTER & MCF_DTIM_DTER_REF) == 0)
    {};

    // Oprire timer
    MCF_DTIM3_DTMR = 0;
}
```

```
//Prototip functii motoare
void setR_engine(double turatie);
void setL_engine(double turatie);
void stopMotoare(void);

void stopMotoare(void)
{
    setReg8(PWMDTY3, 0xFF); //motor
    setReg8(PWMDTY5, 0xFF); //motor dreapta



}

void setL_engine (double turatie)
{
    byte tur;

    if ( (turatie < -1.0) || (turatie > 1.0) ) return;
    if (turatie < 0)
        {
        Motor_Stg_PutVal (1);   //pt. mers inapoi
        turatie = -turatie;
        }
    else
        Motor_Stg_PutVal (0);   //pt. mers inainte

    tur = (byte)(turatie*255.0);
    setReg8(PWMDTY3, tur);
}

// turatie - valoare negativa determina ca motorul sa se invarta invers
//         - valoare pozitiva determina o rotatie a motorului  in sensul
inainte
//         - intervalul permis [-1, 1]
//         - cu cat mai aproape de 1 sau -1 turatia este mai mare

void setR_engine (double turatie)
{
    byte tur;

    if ( (turatie < -1.0) || (turatie > 1.0) ) return;
    if (turatie < 0)
        {
    Motor_Dr_PutVal (1);    //pt. mers inapoi
        turatie = -turatie;
        }
    else
        Motor_Dr_PutVal (0);    //pt. mers inainte

    tur = (byte)(turatie*255.0);

    setReg8(PWMDTY5, tur);
}
///Preluare senzori ADC///
void GetAdcValo(int *ad0,int *ad1,int *ad2,int *ad3,int *ad4,int *ad5,int
*ad6,int *ad7);

void GetAdcValo(int *ad0,int *ad1,int *ad2,int *ad3,int *ad4,int *ad5,int
*ad6,int *ad7)
{
int data[8][8];
```

```
     AD1_Stop();


     AD1_Start();
delayus(2000);
data[0][0]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][0]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][0]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][0]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][0]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][0]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][0]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][0]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][1]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][1]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][1]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][1]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][1]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][1]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][1]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][1]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][2]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][2]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][2]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][2]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][2]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][2]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][2]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][2]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][3]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][3]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][3]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][3]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][3]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][3]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][3]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][3]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][4]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][4]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][4]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][4]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][4]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
```

```
data[5][4]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][4]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][4]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][5]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][5]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][5]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][5]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][5]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][5]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][5]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][5]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][6]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][6]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][6]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][6]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][6]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][6]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][6]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][6]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();

  AD1_Start();
delayus(2000);
data[0][7]=(int)((getReg16(ADRSLT0) + 0x00) >> 5);
data[1][7]=(int)((getReg16(ADRSLT1) + 0x00) >> 5);
data[2][7]=(int)((getReg16(ADRSLT2) + 0x00) >> 5);
data[3][7]=(int)((getReg16(ADRSLT3) + 0x00) >> 5);
data[4][7]=(int)((getReg16(ADRSLT4) + 0x00) >> 5);
data[5][7]=(int)((getReg16(ADRSLT5) + 0x00) >> 5);
data[6][7]=(int)((getReg16(ADRSLT6) + 0x00) >> 5);
data[7][7]=(int)((getReg16(ADRSLT7) + 0x00) >> 5);
     AD1_Stop();


*ad0=(data[0][0]+data[0][1]+data[0][2]+data[0][3]+data[0][4]+data[0][5]+dat
a[0][6]+data[0][7])>>3;

*ad1=(data[1][0]+data[1][1]+data[1][2]+data[1][3]+data[1][4]+data[1][5]+dat
a[1][6]+data[1][7])>>3;

*ad2=(data[2][0]+data[2][1]+data[2][2]+data[2][3]+data[2][4]+data[2][5]+dat
a[2][6]+data[2][7])>>3;

*ad3=(data[3][0]+data[3][1]+data[3][2]+data[3][3]+data[3][4]+data[3][5]+dat
a[3][6]+data[3][7])>>3;

*ad4=(data[4][0]+data[4][1]+data[4][2]+data[4][3]+data[4][4]+data[4][5]+dat
a[4][6]+data[4][7])>>3;

*ad5=(data[5][0]+data[5][1]+data[5][2]+data[5][3]+data[5][4]+data[5][5]+dat
a[5][6]+data[5][7])>>3;

*ad6=(data[6][0]+data[6][1]+data[6][2]+data[6][3]+data[6][4]+data[6][5]+dat
a[6][6]+data[6][7])>>3;
```

```
*ad7=(data[7][0]+data[7][1]+data[7][2]+data[7][3]+data[7][4]+data[7][5]+dat
a[7][6]+data[7][7])>>3;
}
//subrutina converteste un int preluat de la ADC
//intr-un numar in intervalul [0, 1] - la intrarea RMA
double convIR_01 (int val)
{
    //din punct de vedere practic am observat ca
    //valorile primite de la convertor se gasesc
    //in domeniul [0 .. 961] cu un "shift right"
    //suplimentar de 2

    if (val > 950) val = 950;
    if (val < 100) val = 100;

    return (double) (val - 100)/850.0;
}



#endif /* __Functii__ */
```

Fisierul Retea1Start.h

```
int ReteaAlternat1(int x0,int x1,int x2,int x3,int x4,int x5,int x6,int
x7,int dorit);
int ReteaAlternat2(int x0,int x1,int x2,int x3,int x4,int x5,int x6,int
x7,int dorit);
//retea alternativa caz care merge 100%
/*int ReteaAlternat(int x0,int x1,int x2,int x3,int x4,int x5,int x6,int
x7)
{
int i0,i1,i2,i3,i4,i5,i6,i7,y0;
    if(x0>dorit)
    i0=1;
    else
    i0=0;
        if(x1>dorit)
    i1=1;
    else
    i1=0;
        if(x2>dorit)
    i2=1;
    else
    i2=0;
        if(x3>dorit)
    i3=1;
    else
    i3=0;
        if(x4>dorit)
    i4=1;
    else
    i4=0;
        if(x5>dorit)
    i5=1;
    else
    i5=0;
        if(x6>dorit)
    i6=1;
```

```
    else
    i6=0;
        if(x7>dorit)
    i7=1;
    else
    i7=0;
    for(k=0;k<=7;k++)
        for(l=0;l<=1;l++)
    y0=int WInput_n[k][l]+;

return y0;

}*/
//ponderi



int ReteaAlternat1(int x0,int x1,int x2,int x3,int x4,int x5,int x6,int
x7,int dorit)
{
int WInput_n[8],y0=0,o=0,x[7],w=0,i[7];

WInput_n[w]=-1;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=2;
w++;
WInput_n[w]=3;
w++;
WInput_n[w]=2;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=-2;
w=0;

x[o]=x0;
o++;
x[o]=x1;
o++;
x[o]=x2;
o++;
x[o]=x3;
o++;
x[o]=x4;
o++;
x[o]=x5;
o++;
x[o]=x6;
o++;
x[o]=x7;

        for(o=0;o<=7;o++)
    {
    if(x[o]>dorit)
    i[o]=1;
    else
    i[o]=0;
    y0+=WInput_n[o]*i[o]-1;
```

```
}
return y0;
}
int ReteaAlternat2(int x0,int x1,int x2,int x3,int x4,int x5,int x6,int
x7,int dorit)
{

int WInput_n[8],y0=0,o=0,x[7],w=0,i[7];

WInput_n[w]=-1;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=2;
w++;
WInput_n[w]=3;
w++;
WInput_n[w]=2;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=1;
w++;
WInput_n[w]=-2;
w=0;

x[o]=x0;
o++;
x[o]=x1;
o++;
x[o]=x2;
o++;
x[o]=x3;
o++;
x[o]=x4;
o++;
x[o]=x5;
o++;
x[o]=x6;
o++;
x[o]=x7;


    for(o=0;o<=7;o++)
    {
    if(x[o]>dorit)
    i[o]=1;
    else
    i[o]=0;
    y0+=WInput_n[o]*i[o]-1;


}
    return y0;
}
```

6.2 Fisierul main.c LabWindowsCVI

```
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "interf.h"
#include <stdio.h>

#include <rs232.h>
#include <utility.h>
#include <formatio.h>
#include <string.h>
#include "serial.h"
#include "multikey.h"
#include <toolbox.h>
/*---------------------------------------------------------------------------*/
/* Defines                                                                   */
/*---------------------------------------------------------------------------*/

#define TEXT_LENGTH     2000
#define QuitHelp        1
#define InputqHelp      2

//#define  SERIAL_SEND                10  //comanda trimitere
//#define  SERIAL_TBOX_READ           13
//#define  SERIAL_READ                14
//#define  SERIAL_CLEARBOX            15



/*---------------------------------------------------------------------------*/
/* Module-globals                                                            */
/*---------------------------------------------------------------------------*/
int valbut,
    panel_handle,
    config_handle,
    comport,
    baudrate,
    portindex,
    parity,
    databits,
    stopbits,
    inputq,         /* Sets input queue length in OpenComConfig */
    outputq,        /* Sets output queue length in OpenComConfig */
    xmode,
    ctsmode,
    stringsize,
```

```
    bytes_read,
    RS232Error,
    config_flag,
    breakstatus,
    port_open,
    com_status,
    send_mode,
    send_byte,
    send_term_index,
    read_term_index=0,
    read_term,
    inqlen,        /* Stores result from GetInQLen */
    outqlen;       /* Stores result from GetOutQLen */
    val;
    a;
    int L;
    int character;
short read_cnt=9;
double timeout,bytes_sent;
char devicename[30],
     send_data[TEXT_LENGTH],
     read_data[TEXT_LENGTH],
     tbox_read_data[TEXT_LENGTH],
     com_msg[500],
     msg[100];
char send_dataCH;
  char buffer[3];
  int characterr;

static double valw,vals,vala,vald;

/*---------------------------------------------------------------------------*/
/* Internal function prototypes                                              */
/*---------------------------------------------------------------------------*/
void DisplayRS232Error (void);
void SetConfigParms (void);
void GetConfigParms (void);
void DisplayHelp (int);
void EnablePanelControls (int);
void DisplayComStatus (void);
void ActivateSendControls (int);
void SendAscii (void);
void SendByte (void);

  /*---------------------------------------------------------------------------*/
```

```
/* Set the port configuration parameters.                                */
/*-----------------------------------------------------------------------*/
void SetConfigParms (void)
{
   SetCtrlVal (config_handle, CONFIG_COMPORT, comport);
   SetCtrlVal (config_handle, CONFIG_BAUDRATE, baudrate);
   SetCtrlVal (config_handle, CONFIG_PARITY, parity);
   SetCtrlVal (config_handle, CONFIG_DATABITS, databits);
   SetCtrlVal (config_handle, CONFIG_STOPBITS, stopbits);
   SetCtrlVal (config_handle, CONFIG_INPUTQ, inputq);

   SetCtrlIndex (config_handle, CONFIG_COMPORT, portindex);
}

/*-----------------------------------------------------------------------*/
/* Get the port configuration parameters.                                */
/*-----------------------------------------------------------------------*/
void GetConfigParms (void)
{
   GetCtrlVal (config_handle, CONFIG_COMPORT, &comport);
   GetCtrlVal (config_handle, CONFIG_BAUDRATE, &baudrate);
   GetCtrlVal (config_handle, CONFIG_PARITY, &parity);
   GetCtrlVal (config_handle, CONFIG_DATABITS, &databits);
   GetCtrlVal (config_handle, CONFIG_STOPBITS, &stopbits);
   GetCtrlVal (config_handle, CONFIG_INPUTQ, &inputq);
  // GetCtrlVal (config_handle, CONFIG_OUTPUTQ, &outputq);
  // GetCtrlVal (config_handle, CONFIG_CTSMODE, &ctsmode);
   //GetCtrlVal (config_handle, CONFIG_XMODE, &xmode);
   //GetCtrlVal (config_handle, CONFIG_TIMEOUT, &timeout);
   GetCtrlIndex (config_handle, CONFIG_COMPORT, &portindex);
   #ifdef _NI_unix_
      devicename[0]=0;
   #else
      GetLabelFromIndex (config_handle, CONFIG_COMPORT, portindex,
                devicename);
   #endif
}

/*-----------------------------------------------------------------------*/
/* Let the user configure the port.                                      */
/*-----------------------------------------------------------------------*/
int CVICALLBACK ConfigCallBack (int panel, int control, int event,
                        void *callbackData, int eventData1,
                        int eventData2)
{
   switch (event)
```

```
        {
        case EVENT_COMMIT:
            config_handle = LoadPanel (panel_handle, "interf.uir", CONFIG);
            InstallPopup (config_handle);

            /*  If user already has done configuration, then
                display those new parameters.  If entering
                configuration for 1st time, set config_flag
                and use default settings.
            */
            if (config_flag)    /* Configuration done at least once.*/
                SetConfigParms ();
            else                /* 1st time.*/
                config_flag = 1;
            break;
        }
    return(0);
}

/*---------------------------------------------------------------------------*/
/* close the configuration panel.                                            */
/*---------------------------------------------------------------------------*/
int CVICALLBACK CloseConfigCallback (int panel, int control, int event,
                                     void *callbackData, int eventData1,
                                     int eventData2)
{
    switch (event)
        {
        case EVENT_COMMIT :
            port_open = 0;  /* initialize flag to 0 - unopened */
            GetConfigParms ();
            DisableBreakOnLibraryErrors ();
            RS232Error = OpenComConfig (comport, devicename, baudrate, parity,
                                        databits, stopbits, inputq, outputq);
            EnableBreakOnLibraryErrors ();
            if (RS232Error) DisplayRS232Error ();
            if (RS232Error == 0)
                {
                port_open = 1;
                //GetCtrlVal (config_handle, CONFIG_XMODE, &xmode);
                SetXMode (comport, xmode);
                //GetCtrlVal (config_handle, CONFIG_CTSMODE, &ctsmode);
                SetCTSMode (comport, ctsmode);
                //GetCtrlVal (config_handle, CONFIG_TIMEOUT, &timeout);
                SetComTime (comport, timeout);
                EnablePanelControls (0); /* Enable: no errors */
```

```
            }
          else
            EnablePanelControls (1); /* Disable: errors found */
          DiscardPanel (config_handle);
          break;
        }
    return(0);
}
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
void EnablePanelControls (int enable)
{
    SetCtrlAttribute (panel_handle, SERIAL_SEND, ATTR_DIMMED, enable);
    SetCtrlAttribute (panel_handle, SERIAL_READ, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_READ_COUNT, ATTR_DIMMED, enable);
    SetCtrlAttribute (panel_handle, SERIAL_TBOX_READ, ATTR_DIMMED, enable);
    SetCtrlAttribute (panel_handle, SERIAL_BYTES, ATTR_DIMMED, enable);
    SetCtrlAttribute (panel_handle, SERIAL_ERROR, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_FLUSHINQ, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_FLUSHOUTQ, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_GETINQ, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_GETOUTQ, ATTR_DIMMED, enable);
    SetCtrlAttribute (panel_handle, SERIAL_COMSTATUS, ATTR_DIMMED, enable);
    SetCtrlAttribute (panel_handle, SERIAL_READTERM, ATTR_DIMMED, enable);
   // SetCtrlAttribute (panel_handle, SERIAL_SENDMODE, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_RCV_HELP_MSG, ATTR_DIMMED, enable);
    //SetCtrlAttribute (panel_handle, SERIAL_TRANS_HELP_MSG, ATTR_DIMMED,
enable);
    SetCtrlAttribute (panel_handle, SERIAL_CLEARBOX, ATTR_DIMMED, enable);
    ActivateSendControls (enable);
}

/*---------------------------------------------------------------------------*/
/*  Activate or deactivate the Send controls.  For activate, enable = 0,     */
/*  for deactivate, enable = 1, since 0 is not dimmed and 1 is dimmed.       */
/*---------------------------------------------------------------------------*/
void ActivateSendControls (int enable)
{

      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED,enable);
      SetCtrlAttribute (panel_handle, SERIAL_SENDTERM, ATTR_DIMMED, enable);
     //  SetCtrlAttribute (panel_handle, SERIAL_TRANS_HELP_MSG,
ATTR_DIMMED,enable);


}
```

```
/*---------------------------------------------------------------------------*/
/* Clear the character display.                                              */
/*---------------------------------------------------------------------------*/
int CVICALLBACK ClearBoxCallBack (int panel, int control, int event,
                                  void *callbackData, int eventData1,
                                  int eventData2)
{
   if (event == EVENT_COMMIT)
      ResetTextBox (panel_handle, SERIAL_TBOX_READ, "\0");
   return 0;
}



/*---------------------------------------------------------------------------*/
/* Send the appropriate data to the port.                                    */
/*---------------------------------------------------------------------------*/
int CVICALLBACK SendCallBack (int panel, int control, int event,
                              void *callbackData, int eventData1,
                              int eventData2)
{
   if (event == EVENT_COMMIT)
      {
         GetCtrlVal (panel_handle, SERIAL_TBOX_SEND, send_data);
         SendAscii ();

      RS232Error = ReturnRS232Err ();
      if (RS232Error)
         DisplayRS232Error ();
      SetCtrlAttribute (panel_handle, SERIAL_BYTES, ATTR_DIMMED, 0);
      SetCtrlVal (panel_handle, SERIAL_BYTES, bytes_sent);
      }
   return 0;
}

/*---------------------------------------------------------------------------*/
/* Send ASCII characters to the port.                                        */
/*---------------------------------------------------------------------------*/
void SendAscii (void)
{          char valoare;
      // sprintf(valoare,"%d",character) ;
   // if(character==119)
   //       valoare = '119';

switch (send_term_index)
```

```
        {
        case 1:
           strcat(send_data, "\r");
           break;
        case 2:
           strcat(send_data, "\n");
           break;
        }

    stringsize = StringLength (send_data);  //send_data
    bytes_sent = ComWrt (comport, send_data, stringsize);

//ComWrtByte (comport, send_data);
}

/*---------------------------------------------------------------------------*/
/* Read data from the COM port.                                              */
/*---------------------------------------------------------------------------*/
int CVICALLBACK ReadCallBack (int panel, int control, int event,
                              void *callbackData, int eventData1,
                              int eventData2)
{

    switch (event)
        {
        case EVENT_COMMIT:
           read_data[0] = '\0';
         //  GetCtrlVal (panel_handle, SERIAL_READ_COUNT, &read_cnt);
           //GetCtrlIndex (panel_handle, SERIAL_READTERM, &read_term_index);
           switch (read_term_index)
              {
              case 0:
                 read_term = 0;
                 break;
              case 1:
                 read_term = 13;
                 break;
              case 2:
                 read_term = 10;
                 break;
              }
           if (read_term)
              bytes_read = ComRdTerm (comport, read_data, read_cnt,
                                      read_term);
           else
              bytes_read = ComRd (comport, read_data, read_cnt);
```

```
            /*  Copy subset of read_data into tbox string for display.
               ComRdTerm does not automatically put null byte after
               number of bytes read into read_data string. */
            CopyString (tbox_read_data, 0, read_data, 0, bytes_read);
            SetCtrlVal (panel_handle, SERIAL_TBOX_READ, tbox_read_data);
            RS232Error = ReturnRS232Err ();
            if (RS232Error)
               DisplayRS232Error ();
            break;
         case EVENT_RIGHT_CLICK :
            break;
         }
      return 0;
}



int main (int argc, char *argv[])
{
   if (InitCVIRTE (0, argv, 0) == 0)
      return -1;  /* out of memory */
   if ((panel_handle = LoadPanel (0, "interf.uir", SERIAL)) < 0)
      return -1;
   DisplayPanel (panel_handle);
   RunUserInterface ();
   DiscardPanel (panel_handle);
   return 0;
}

int CVICALLBACK Ftimer (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{
   switch (event)
   {
      case EVENT_TIMER_TICK:

            read_data[0] = '\0';
         GetCtrlVal (panel_handle, SERIAL_READ_COUNT, &read_cnt);
         GetCtrlIndex (panel_handle, SERIAL_READTERM, &read_term_index);
         switch (read_term_index)
            {
            case 0:
               read_term = 0;
               break;
            case 1:
               read_term = 13;
```

```
            break;
         case 2:
            read_term = 10;
            break;
         }
      if (read_term)
         bytes_read = ComRdTerm (comport, read_data, read_cnt,
                                 read_term);
      else
         bytes_read = ComRd (comport, read_data, read_cnt);
      /*  Copy subset of read_data into tbox string for display.
         ComRdTerm does not automatically put null byte after
         number of bytes read into read_data string. */
      CopyString (tbox_read_data, 0, read_data, 0, bytes_read);
      SetCtrlVal (panel_handle, SERIAL_TBOX_READ, tbox_read_data);
      RS232Error = ReturnRS232Err ();
      if (RS232Error)
         DisplayRS232Error ();
      break;
   }
   return 0;
}


int CVICALLBACK fPanel (int panel, int event, void *callbackData,
      int eventData1, int eventData2)
{
   switch (event)
   {
      case EVENT_GOT_FOCUS:

         break;
      case EVENT_LOST_FOCUS:

         break;
      case EVENT_CLOSE:
         QuitUserInterface (0);
         break;
   }
   return 0;
}

int CVICALLBACK fTimTast (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{
   switch (event)
```

```
    {
      case EVENT_TIMER_TICK:

                valw -= 0.01;
                if (valw < 0) valw = 0;
                SetCtrlVal (panel, SERIAL_NUMERIC, valw);


                vala -= 0.01;
                if (vala < 0) vala = 0;
                SetCtrlVal (panel, SERIAL_NUMERIC_2, vala);


                vals -= 0.01;
                if (vals < 0) vals = 0;
                SetCtrlVal (panel, SERIAL_NUMERIC_3, vals);


                vald -= 0.01;
                if (vald < 0) vald = 0;
                SetCtrlVal (panel, SERIAL_NUMERIC_4, vald);

          break;
    }
    return 0;
}

int CVICALLBACK fTasta (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)
{

    static int last_character;
    static char nou_char;  //0 nou caracter, 1 vechi caracter
    static time_t t_noucharacter;
    time_t t_acum;
    char buffer[3];
    switch (event)
    {
      case EVENT_KEYPRESS:
          // important aici am receptie si transmitere tasta//
          SetCtrlAttribute (panel, SERIAL_TBOX_SEND, ATTR_CTRL_VAL, "");
          character = GetKeyPressEventCharacter  (eventData2);
             memset (buffer, 0, sizeof (buffer));
             CmbSetC (buffer, character);

              SetCtrlAttribute (panel, SERIAL_ORIGINAL,
```

```
                ATTR_CTRL_VAL, buffer);

          stringsize = StringLength (buffer);  //send_data
bytes_sent = ComWrt (comport, buffer, stringsize);
//      SetCtrlVal (panel, SERIAL_TBOX_SEND, "");


   //  GetCtrlVal (panel_handle, SERIAL_TBOX_SEND, send_data);
     //  SendAscii ();


   RS232Error = ReturnRS232Err ();
   if (RS232Error)
      DisplayRS232Error ();
   SetCtrlAttribute (panel_handle, SERIAL_BYTES, ATTR_DIMMED, 0);
  // SetCtrlVal (panel_handle, SERIAL_BYTES, bytes_sent);

      if (character != 0)
      {
         if (last_character != character)
            {
            nou_char = 0;
            last_character = character;
            t_noucharacter = clock ();

            valw = 0;
            vala = 0;
            vals = 0;
            vald = 0;
            }
         else
            {
            nou_char = 1;
            t_acum = clock ();

            if ( t_acum > (t_noucharacter + 100) )
               {
                  if(character==119)
                  {

                     valw += 0.01;
                     if  (valw > 1) valw = 1;
                     SetCtrlVal (panel, SERIAL_NUMERIC, valw);
                     t_noucharacter = t_acum;
                  }
                  if(character==97)
```

```
                    {

                        vala += 0.01;
                        if  (vala > 1) vala = 1;
                        SetCtrlVal (panel, SERIAL_NUMERIC_2, vala);
                        t_noucharacter = t_acum;
                    }
                    if(character==115)
                    {

                        vals += 0.01;
                        if  (vals > 1) vals = 1;
                        SetCtrlVal (panel, SERIAL_NUMERIC_3, vals);
                        t_noucharacter = t_acum;
                    }
                    if(character==100)   //d
                    {

                        vald += 0.01;
                        if  (vald > 1) vald = 1;
                        SetCtrlVal (panel, SERIAL_NUMERIC_4, vald);
                        t_noucharacter = t_acum;
                    }
                }

            }
        }


        break;
    }
    return 0;
}
int CVICALLBACK QuitCallback (int panel, int control, int event,
                              void *callbackData, int eventData1, int eventData2)
{
    if (event == EVENT_COMMIT)
        QuitUserInterface (0);
    return 0;
}
  /*---------------------------------------------------------------------------*/
/* Display error information to the user.                                    */
/*---------------------------------------------------------------------------*/
void DisplayRS232Error (void)
{
    char ErrorMessage[200];
```

```
   switch (RS232Error)
      {
      default :
         if (RS232Error < 0)
            {
            Fmt (ErrorMessage, "%s<RS232 error number %i", RS232Error);
            MessagePopup ("RS232 Message", ErrorMessage);
            }
         break;
      case 0  :
         MessagePopup ("RS232 Message", "No errors.");
         break;
      case -2 :
         Fmt (ErrorMessage, "%s", "Invalid port number (must be in the "
                           "range 1 to 8).");
         MessagePopup ("RS232 Message", ErrorMessage);
         break;
      case -3 :
         Fmt (ErrorMessage, "%s", "No port is open.\n"
            "Check COM Port setting in Configure.");
         MessagePopup ("RS232 Message", ErrorMessage);
         break;
      case -99 :
         Fmt (ErrorMessage, "%s", "Timeout error.\n\n"
            "Either increase timeout value,\n"
            "       check COM Port setting, or\n"
            "       check device.");
         MessagePopup ("RS232 Message", ErrorMessage);
         break;
      }
}

int CVICALLBACK fBINARYSWITCH (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{          int binary0;
   switch (event)
   {
      case EVENT_COMMIT:
         GetCtrlVal (panel, SERIAL_BINARYSWITCH, &binary0);
               if(binary0==1)
               {
         GetCtrlAttribute (panel, SERIAL_Autonom,
         ATTR_CTRL_VAL, buffer);
      stringsize = StringLength (buffer);  //send_data
   bytes_sent = ComWrt (comport, buffer, stringsize);
```

```
      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 1); }
            else
      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 0);
        break;
   }
   return 0;
}

int CVICALLBACK fBINARYSWITCH_2 (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{       int binary0;

   switch (event)
   {
      case EVENT_COMMIT:
        GetCtrlVal (panel, SERIAL_BINARYSWITCH_2, &binary0);
                    if(binary0==1)  {
        GetCtrlAttribute (panel, SERIAL_Invatare,
        ATTR_CTRL_VAL, buffer);
      stringsize = StringLength (buffer);  //send_data
   bytes_sent = ComWrt (comport, buffer, stringsize);
      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 0);
                    }
            else
      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 1);
        break;
   }
   return 0;
}

int CVICALLBACK fBINARYSWITCH_3 (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{        int binary0;
   switch (event)
   {
      case EVENT_COMMIT:
        GetCtrlVal (panel, SERIAL_BINARYSWITCH_3, &binary0);
                    if(binary0==1)    {
          GetCtrlAttribute (panel, SERIAL_Liber,
        ATTR_CTRL_VAL, buffer);
      stringsize = StringLength (buffer);  //send_data
   bytes_sent = ComWrt (comport, buffer, stringsize);



      SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 0);  }
```

```
            else
        SetCtrlAttribute (panel_handle, SERIAL_TBOX_SEND, ATTR_DIMMED, 1);
            break;
    }
    return 0;
}
```

7. Bibliografie

1. L298N Datasheet
2. http://ro.wikipedia.org
3. http://en.wikipedia.org
4. http://ro.math.wikia.com/wiki/Motor_de_curent_continuu
5. https://www.google.ro/
6. http://cncro.ro/lang/ro-ro/realizare-cablaje/
7. Cursuri EPIOC
8. LabWindows CVI example – help
9. https://translate.google.com