

## Cuprins

Capitolul 1.Introducere .....	5
1.1.Memoriu justificativ .....	5
1.2.Rezumat .....	8
Capitolul 2. Fundamentare teoretică .....	9
2.1.Senzori .....	9
2.1.1.Senzor de atingere .....	10
2.1.2. Senzor de culoare.....	11
2.1.3.Senzorul ultrasonic .....	12
2.1.4.Senzorul giroscop .....	13
2.2. Servomotorul.....	14
2.3.Componentele de legătură .....	18
2.4. Unitatea central de procesare .....	19
2.4.1. Ecranul LCD .....	21
2.4.2. Interfața digitală și Bluetooth.....	21
2.4.3. Comunicație Bluetooth .....	21
2.4.4. Sistemul de operare .....	22
2.4.5. RobotC.....	23
2.5.Conectori.....	25
Capitolul 3.Parrea practică .....	26
3.1.Necesarul de a construi acest robot .....	26
3.2.Parrea mecanică a robotului .....	27
3.3.Cum funcționează acest robot .....	30
3.4.Înterpretarea datelor pe calculator .....	33
3.4.1.Calibrarea giroscopului.....	36
3.4.2.Comandă de viraj.....	37
3.4.3.Afișarea valorilor senzorului giroscop .....	38
3.4.4. Controlul giroscopului .....	39
3.4.5.Distanța față de obiect și afișare.....	40
3.4.6.Comunicarea volorii distanței celuialt NXT .....	41

3.4.7.Ștergerea valorilor .....	42
3.4.8.Comanda de viraj pentru NXT2 .....	43
3.4.9.Recepționarea mesajului de la NXT1 .....	45
3.4.10.Controlul motoarelor în funcție de mesajul recepționat .....	45
3.5.Pseudocodul programului .....	46
Concluzie.....	51
Bibliografie.....	52
Anexă .....	53

# Capitolul 1.Introducere

## 1.1.Memoriu justificativ

Pentru mai mult de 100 de ani, oamenii au gândit, visat și scris despre roboți. Astăzi, roboții sunt foarte reali și înlocuiesc o mare varietate de locuri de muncă importante în diferite domenii.

Domeniile unde se întâlnesc sunt: construcții, comerț, transport, circulația mărfurilor, agricultură, supraveghere, inspecție, protecție, medicină, gospodarie, etc. Roboții sunt utilizați în procesele de fabricație, ca manipolatoare pe adâncul mării, în diverse laboratoare unde se lucrează cu substanțe radioactive sau există temperaturi ridicate sau alte locuri periculoase. Unii dintre ei sunt mobili, având sistem de locomotie încorporat, putându-se deplasa pe uscat, pe apa, în aer.

Există atât de multe tipuri de roboți folosiți pentru diverse servicii, încât o enumerare a tuturor tipurilor acestora ar fi aproape imposibilă. Pot exista mai multe tipuri de roboți care pot fi :

✚ *Androizi,(fig.1)* sunt roboți care încearcă să mimeze comportamentul și înfățișarea umană ;



Fig.1 Androizi

✚ *Roboți statici,(fig.2)* roboții care sunt folosiți în diferite fabrici și laboratoare ca de exemplu brațe robot ;



Fig.2 Braț robotic

- ✚ *Tele-roboti*, sunt roboții care pot fi ghidați prin diferite dispozitive cum ar fi telecomanda de un operator uman;
- ✚ *Roboți mobili*, roboți care se deplasează într-un anumit mediu fără intervenție umană și realizează anumite obiective;
- ✚ *Roboți autonomi*, (fig.3) roboți care îndeplinesc sarcinile fără indiciu din partea unui operator uman și își obțin energia necesară funcționării din mediul înconjurător;



Fig.3 Robot autonom

Un robot este format din elemente mecanice, motoare, senzori și module electrice cu ajutorul cărora robotul poate fi controlat. Pentru realizarea sistemului de conducere a roboților industriali se propune aplicarea unor tehnici ale inteligenței artificiale pentru realizarea nivelelor ierarhice superioare și a unor metode avansate, de control predictiv, în materializarea nivelului ierarhic inferior. Roboții primesc informații atât de la senzorii externi, despre mediul înconjurător în care aceștia funcționează, cât și de la senzorii interni care măsoară poziția și orientarea elementelor mobile cu care sunt echipați.

În această lucrare îmi propun să fac un robot care să urce cu ușurință obstacolele întâlnite. Pe lângă aceasta vreau să mențin și un giroscop în poziție perpendiculară față de suprafața pământului.

## 1.2.Rezumat

Această lucrare are patru capitole, după cum urmează:

- ✚ **Capitolul 1.Introducere** , în acest capitol găsim Momoriul justificativ și Rezumatul acestei lucrări.
- ✚ **Capitolul 2.Fundamente teoretice**, unde sunt descrise în detaliu părțile componente ale Platformei Lego Mindstorms, cum ar fi senzorii care poate folosi această platformă, servomotoarele, unitatea centrală de procesare unde este stocat soft-ul cu care controlăm senzorii și motoarele, cablurile USB cu care se face legătura senzorilor la unitate și piesele ce le folosim.
- ✚ **Capitolul 3. Partea practică**, aici se găsește partea practică a aceste lucrări, adică cum a fost realizat acest robot și de ce anume așa și nu altfel. La primul punct din din aceste capitol, adică “3.1. Necesarul de a construi acest robot” , pentru construcția primului bloc este necesar de următoare piese: două unități de procesare, cinci servomotoare, un senzor giroscop, un senzor ultrasonic, opt roti, patru sinile, cabluri de transmiterea datelor, piesele de legătură. Tot aici găsim descris soft-ul acestui robot care îl întâlnim la punctul “3.4.”, unde este detaliată fiecare bucățică de cod.
- ✚ **Capitolul 4.Evaluarea rezultatelor**, aici vom avea Concluzia, Bibliografia și Anexa acestei lucrări.

## Capitolul 2. Fundamentare teoretică

În această lucrare se va folosi Platforma Lego Mindstorms pentru a prezenta un robot ce este capabil să treacă peste obstacolele și să mențină senzorul giroscop de pe platformă perpendicular față de suprafața pământului. Avantajele acestei platforme sunt reconfigurabilitatea, adică folosirea a aceleiași componente în diferite structuri, ușurința de montare a pieselor și o interfață de programare prietenoasă. Nu în ultimul rând aceasta are și dezavantaje, fiind fragilitatea și precizia scăzută a structurilor construite cu componentele Lego.

Platforma Lego Mindstorms include:

- + unitatea centrală de procesare NXT cu patru intrari si trei iesiri;
- + senzori;
- + servomotoare;
- + cabluri USB pentru a conecta unitatea centrală la calculator;
- + componentele de legătură;

### 2.1.Senzori

Controlerul NXT are patru porturi pentru conectarea senzorilor. Cu acești senzori putem să aflăm mai multe informații despre mediul înconjurător ca de exemplu temperatură, greutate, distanță, mărime, rezistență fizică, etc. În setul Lego Mindstorms NXT găsim patru senzori cum ar fi: sensor de culoare, sensor de atingere și sensor de ultrasonic.

Pe lângă senzorii ce intră în acest set mai există și alți senzori compatibili cum ar fi:

- + mecanici: senor gyroscopic, sensor de accelerație
- + optici: sensor de culoare, sensor de lumină
- + ambientali: sensor de temperatură, sensor de sunet
- + orientat: sensor de distanță, sensor busolă



Fig. 4 Senzori Lego

### 2.1.1.Senzor de atingere

Funcționează ca un întrerupator având doua setări apăsat sau eliberat, când este atins circuitul se închide și curentul electric trece prin el. După cum se vede în fig.5.

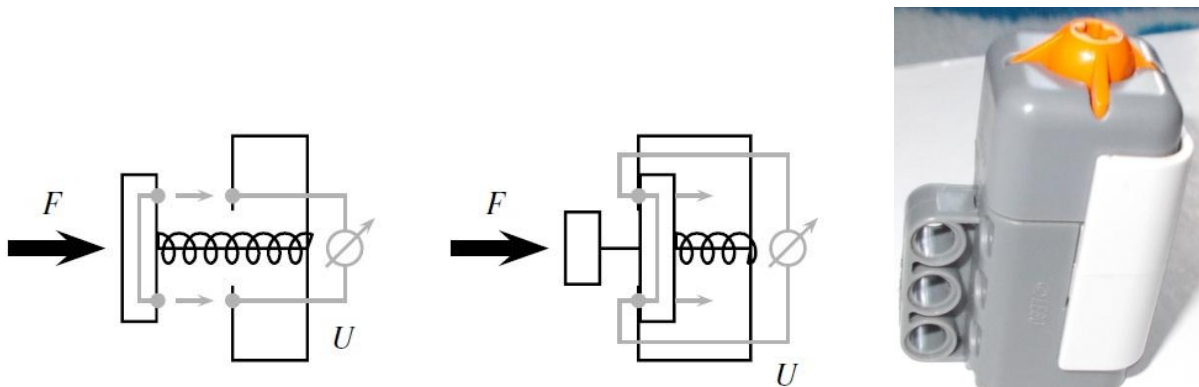


Fig.5 a)apăsarea butonului      b) deschiderea butonului      c) senzor de atingere

O altă comandă al senzorului este de amortizare care este o modalitate mai ușoară de a interacționa cu mediul înconjurător permițând astfel robotului să detecteze obstacole când senzorul se atinge și să-și schimbe orientarea.

Deci pe lângă cele doua setări apăsat și lăsat mai avem și o a treia setare de amortizare. Toate aceste setări sunt importante deoarece nu știi cum va reacționa robotul în diferite situații.



### 2.1.2. Senzor de culoare



Fig.6 Senzor de culoare

Acest sensor este fotosensibil și poate să distinga patru culori sau variații a intensității luminii ambiante care cade pe o fotodiodă. El are trei Led-uri, emițătoare de lumină roșie, verde și albastră.

Senzorul poate fi utilizat în trei moduri:

- modul inactiv (denumit și modul lumină ambientă), atunci când senzorul doar măsoară intensitatea de lumină care cade pe el.
- modul activ (numit și modul lumină reflectată), atunci când senzorul acționează ca o sursă de lumină, folosind lumina roșie a ledului, utilizându-se atunci când senzorul este aproape de o suprafață ca să poată determina cât de întunecată este această suprafață. Cu cât valoarea măsurată este mai mare cu atât suprafața este mai întunecată.
- modul senzorului de culoare , are aceeași funcționalitate ca modul inactiv. Diferența este că poate detecta 6 culori diferite: negru, alb, albastru, verde, galben și roșu.

Senzorul constă din două blocuri, care sunt Light Sensor Interface și Light Sensor Read. Primul bloc este blocul de ieșire a interfeței controlerului NXT și cel de-al doilea bloc se folosește pentru primirea datelor de la senzorul de culoare. Acest sensor măsoară intensitatea de lumină luând valori pe un interval de la 0 (foarte întunecat) până la 100 (foarte luminos).

### 2.1.3.Senzorul ultrasonic

Senzorul cu ultrasunete cunoscut sub numele emițător-receptor sau de traductor, funcționează similar ca un radar sau sonar.



Fig.7 Senzorul ultrasonic

Este cel mai întâlnit sensor ce folosește semnale acustice pentru o măsurare mai precisă a distanțelor față de alte obiecte fixe sau mobile din jur. Emițătorul sonar al senzorului transmite semnal acustic în mediu la care reflexia acestuia este recepționată de componenta detector al sensorului. Sunetul transmis de către sonar sunt din spectrul sunetelor ultrasonice și au o frecvență foarte înaltă pentru a nu putea fi depistate de urechea umană.

Principiul de funcționare:

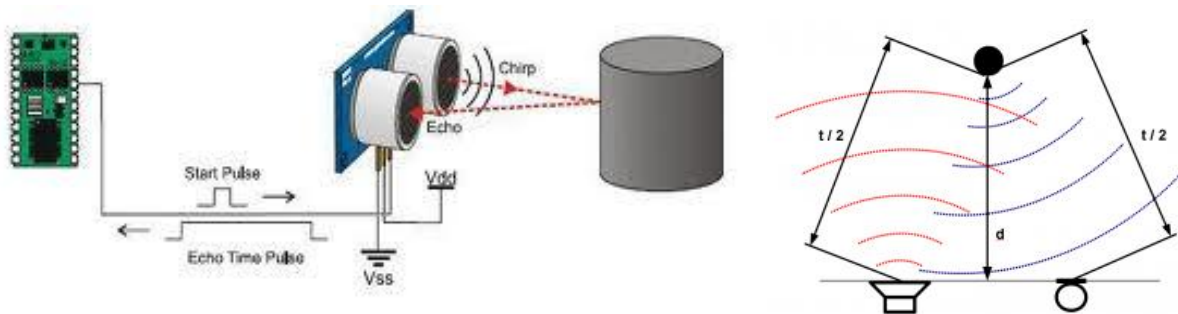


Fig.8 Funcționarea senzorului

Pentru a măsurare distanța trebuie să măsurăm timpul de propagare.

$$d=v*t ,$$

d-distanța, v-viteza de propagare a undei, t-timpul măsurat

Cu cât timpul măsurat este mai mare cu atât distanța față de un obiect este mai mare. Distanța depinde de indicele de refracție al mediului care el depinde la rândul său de temperatură, presiune, lungime de undă, umiditatea aerului. Senzorul ultrasonic poate măsura o distanță între 0 și 255 centimetri cu o precizie de +/-3cm. Citirea sensorului primind valori de la 0 la 100. Măsurătorile care sunt sub 4cm nu se pot face, din cauza timpului necesar pentru întoarcerea valorii. Iar măsurătorile de până la 20cm sunt relativ precise. Receptorul acestui sensor este plasat pe stânga și transmițătorul este plasat pe dreapta. De aceea măsurătorile efectuate pe stânga sunt mai puțin precise și apar astfel erori. Între 20cm și 80cm, măsurătorile au o marjă de eroare de 8%. O problemă este în detectarea unor obiecte, cum ar fi cele moi care absorb ultrasunetele sau oglinda care reflectă ultrasunetele.

#### 2.1.4.Senzorul giroscop

Un senzor giroscop sau giroscopul este un titirez, care este montat într-o suspensie cardanică pentru a conserva momentul cinetic ca să-și poată menține orientarea în spațiu. Atunci când o forță externă încearcă să rotească axa de rotație a giroscopului, creează un cuplu care poate fi măsurat. Într-un montaj cardanic, deviația poate fi determinată de codificator în cele trei direcții spațiale montate pe osii. Giroscopul rotativ menține orientarea într-un montaj cardanic după cum vedeți în figura de mai jos

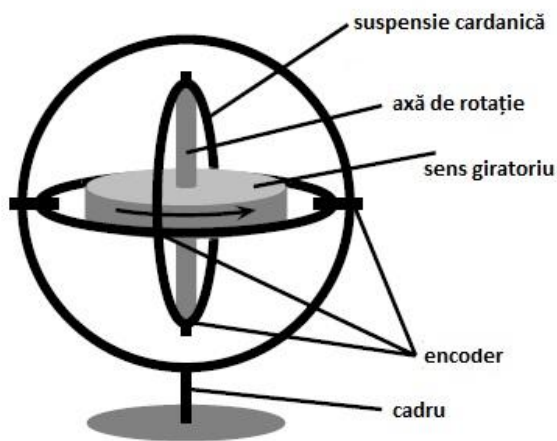


Fig.10 Giroscop



Fig.11 Senzorul HiTechnic giroscop

Senzorul HiTechnic permite să detectăm mișcări de rotație ale vehiculului NXT cu viteza de rotație de  $\pm 360^\circ$  pe secundă. Valoarea măsurată reprezintă viteza de rotație în grade pe secundă. Rata de rotație poate fi citită de aproximativ 300 de ori pe secundă. Axa de măsurare a senzorului giroscop este în plan vertical, adică poziționat cu capacul cu capăt negru în sus după cum este arătat în figura de mai jos.

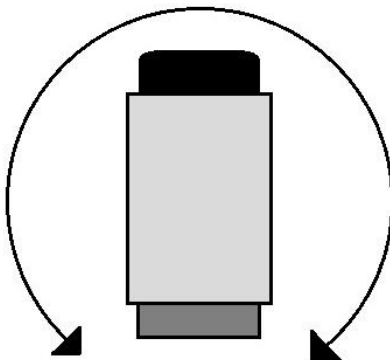


Fig.12 Functionarea

În timpul funcționării, se poate întâmpla ca senzorul să detecteze o viteză chiar și în repaus de derivă. Pentru a evita deteriorarea valorilor, trebuie să recalibrăm senzorul regulat. Pentru a obține rezultate bune. O problema a senzorilor giroscopici reprezintă dezechilibre mici de giroscop. Această problemă se numește Gyro derivă.

## 2.2. Servomotorul

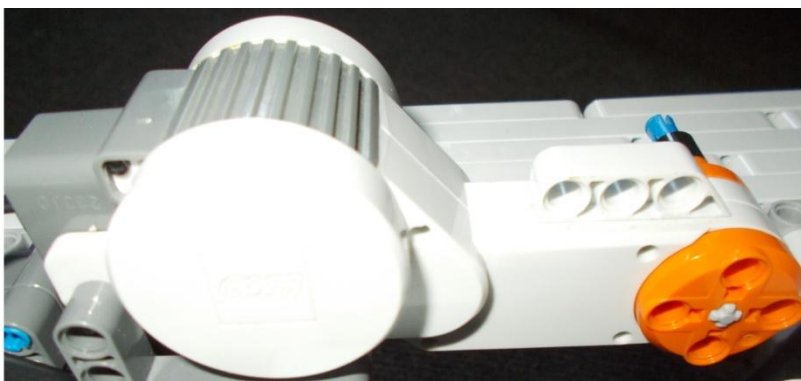


Fig.13 Motorul Lego

NXT folosește trei motoare electrice de curent continuu și pot fi alimentate la 9V sau 12V. Ele sunt transductoare electromagnetice care transformă energia electric în energie mecanică. Motorul este format din două părți, una fiind statorul iar cealaltă parte este rotorul. Statorul poate avea mai mulți poli magnetici, dar rotorul având o bobină. O tensiune oarecare  $U$  este aplicată la o așa-numită armatură, care funcționează ca o bobină. Această tensiune produce un câmp magnetic care interacționează cu câmpul magnetic permanent și înconjoară armatură se află în mișcare. Modificarea turației motorului necesită modificarea tensiunii aplicată pe înfășurarea rotorului. După cum avem în figura de mai jos (Fig.14).

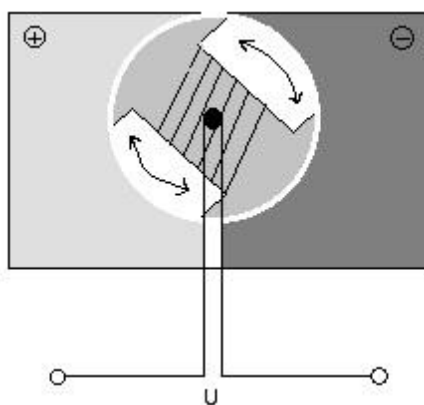


Fig.14 Funcționarea motorului

Pentru a modifica sensul rotației se realizează prin modificarea polarității tensiunii de alimentare sau altfel, adică modificarea sensului câmpului magnetic. Polaritatea tensiunii de alimentare se modifică cu ajutorul punților H. Această punte H fiind un circuit electronic are 4 “întrerupătoare”, care schimbă sensul curentului ce trece prin motor. Cu cât tensiunea aplicată este mai mare cu atât motorul se mișcă mai repede reducând cuplul. Ușurința cu care se manipulează aceste motoare oferă un avantaj. Motorul Lego este o combinație de motor electric, de transmisie și sensor de rotație. Impulsurile de curent ce generează o frecvență predefinită, face ca lungimea platoului să determine cuplu și viteză. Tensiune se schimbă mereu între tensiunea de alimentare  $V_{dd}$  (“high”) și masă (“low”). Puterea de ieșire a motorului exprimată în procente corespunde mediei aritmetice:

$$p = t_1 * f_{pwm} = t_1 / \lambda_{pwm}$$

,unde frecvența PWM este:  $f_{pwm} = 1/\lambda_{pwm}$  , iar  $t_1$  este durata impulsului, după cum se vede și în fig.15.

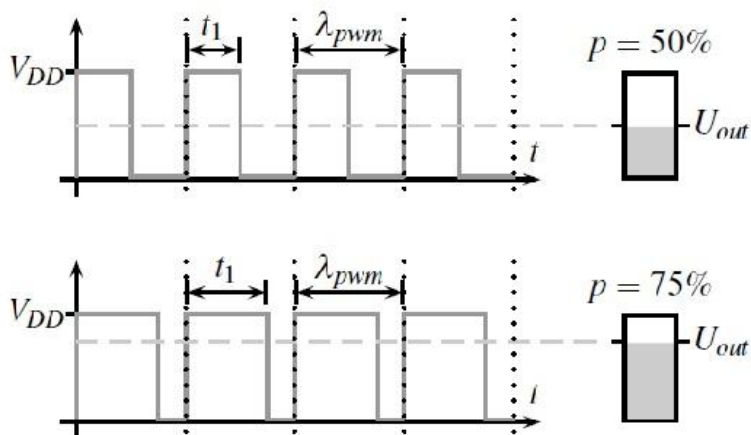


Fig.15 Puterea ieșită de la motor

La motoarele NXT, frecvența PWM este de 200Hz, ceea ce corespunde la o lățime maximă a impulsului de 5ms. Față de motorul prezentat mai sunt și motoare NXT cu senzori de rotație, cu care se poate măsura rotația motorului.

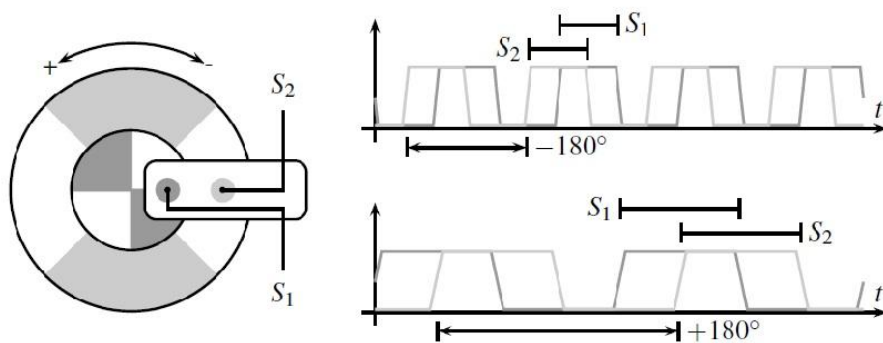


Fig.16 Motoare NXT cu senzori de rotație

Un servomotoar Lego este format din:

- ✚ corpul propriu-zis a motorului
- ✚ angrenaj cu roți dinațate

✚ senzorul de rotație

Aceste motoare au o precizie de rotație de un grad și pot măsura aceste rotații în grade sau în rotații complete. Ele au o serie de avantaje unul din ele fiind protecția la supracurent și supratensiune. Dar pe lângă avantaje mai sunt și dezavantaje cum ar fi turație mică la ieșire, disponibil o gamă mică de puteri, posibilitatea de a se uza la sarcini mari a roților dințate care sunt din plastic.

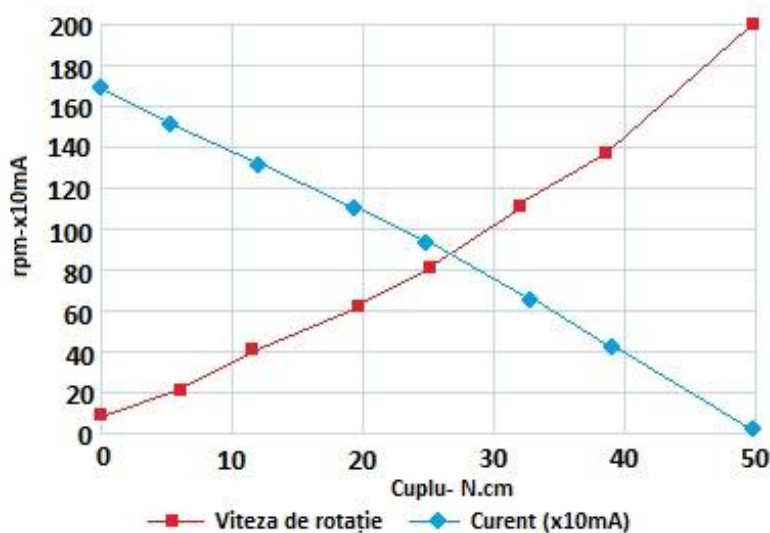


Fig.17 Curentul și viteza de rotație a motorului

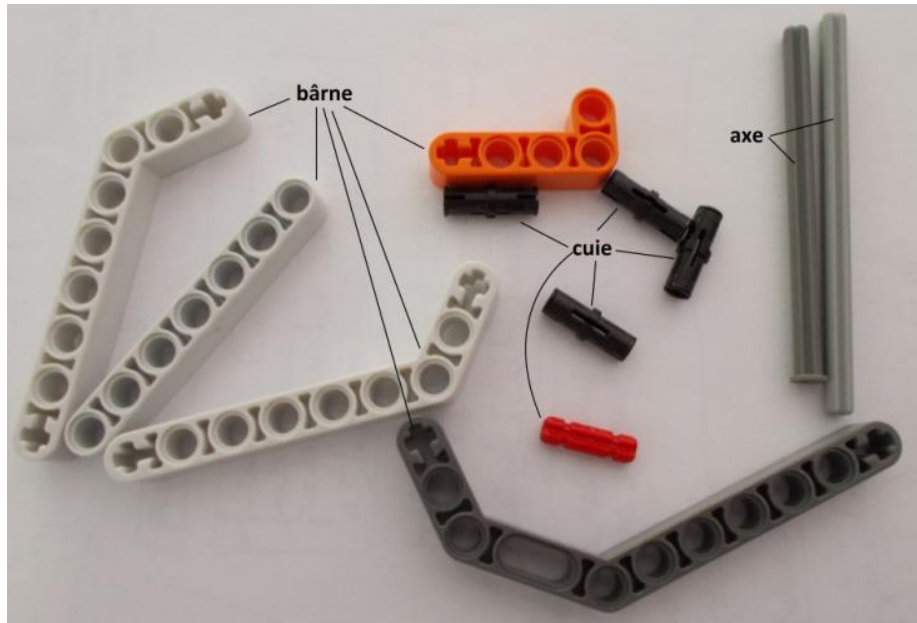
Servomotorul are următoarele specificații după cum se vede în tabelul de mai jos:

Tensiunea motorului NXT	Cuplu	Viteza de rotație	Curent	Puterea mecanică	Puterea electrică	Eficiența
4,5V	16,7 N.cm	33rpm	0,6A	0,58W	2,7W	21,4%
7V	16,7 N.cm	82rpm	0,55A	1,44W	3,85W	37,3%
9V	16,7 N.cm	117rpm	0,55A	2,03W	4,95W	41%
12V	16,7 N.cm	177rpm	0,58A	3,10W	6,96W	44,5%

Tabelul.1 Specificațiile motorului NXT

## 2.3.Componentele de legătură

Partea tehnică a chitului Lego este diversificată și include bârne, cuie, axe, angrenaje, benzi de cauciuc, pneumatice, arcuri.



Bârnele sunt niște blocuri lungi cu un rând de gauri rotunde. Sunt destul de multe dar ele și de dimensiuni diferite fiind structura de bază a sistemului tehnic.

Cuiele sunt piese cilindrice mici care sunt montate în gaurile bânelor și cu ajutorul lor prindem bârnele între ele. Acest chit are în componența sa mai multe tipuri de cuie care sunt negre, albastre, gri, aurii și rosii. Cuiele negre și cele albastre țin țeapăn montarea bânelor, însă cuiele gri și aurii formează între bârne balamale libere.

Axele cunoscute și sub numele de axe transversale, sunt niște tije în formă de cruce. Ele sunt foarte des folosite în piesele rotative și sunt de diferite dimensiuni și de câteva culori.

Angrenajele sunt cilindri plane cu gauri în centru și au dinți în jurul marinelor. Majoritatea din ele pot fi montate pe axe.



## 2.4. Unitatea central de procesare

Unitatea centrală de procesare este un element de bază al limbajului grafic LEGO Mindstorms NXT și centrul operațional al sistemului. Ea execută programele utilizatorului și controlează comunicarea cu senzorii, servomotoarele, cu PC-ul sau cu alte unități NXT.

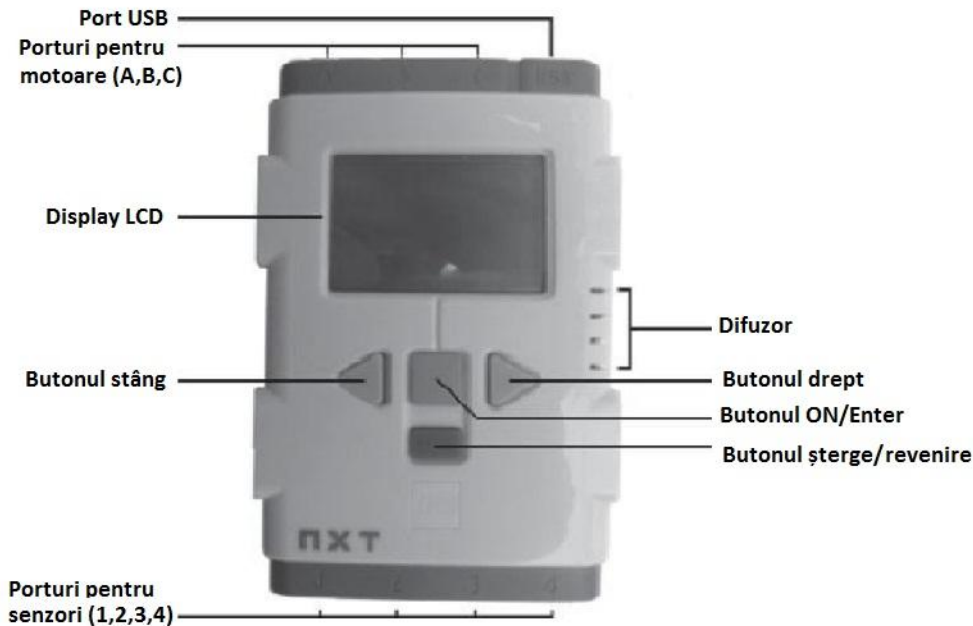


Fig.18 Unitatea de procesare și componentele ei

Această unitate are următoarele specificații tehnice:

- ✚ microcontroler Atmel AT91SAM7 pe 32 de biți, cu memorie RAM de 64KB și memorie Flash de 256KB;

Există o mare varietate de microcontrolere AT91 cu memorie flash și un miez ARM7TDMI. Viteza clock-ului acestor chip-uri este de 60MHz.

- ✚ microcontroler ATmelga48 pe 8 biți cu o frecvență de 4MHz, având o memorie RAM de 512 biți și memorie Flash de 4KB;
- ✚ comunicații fără fir Bluetooth (clasa a II-V2.0 compatibil), având 47 Kocteți memoria RAM și 8Mbți memorie flash;

- ✚ Port USB (12 Megabiți pe secundă);
- ✚ 4 porturi de intrare;
- ✚ 3 porturi de ieșire;
- ✚ Un ecran LCD de 100x64 pixeli;
- ✚ Difuzor-8KHz (sunet de calitate), o rezoluție de 8 biți și rată de eșantionare de 2-16KHz;
- ✚ Sursă de alimentare (6 baterii AA);

Sistemul de procesare al robotului are două microcontrolere, principal și auxiliar. Microcontrolerul principal execută programele utilizatorului, iar cel auxiliar controlează servomotoarele prin PWM și menține viteza în funcție de sarcină.

Sub carcasa unității centrale de procesare, interfețe de comunicare cu senzorii, afișajul cu cristale lichide, microcontrolerul de comunicare Bluetooth, porturile de conectare cu senzorii și servomotoarele, butoane de navigare prin meniu, un difuzor.

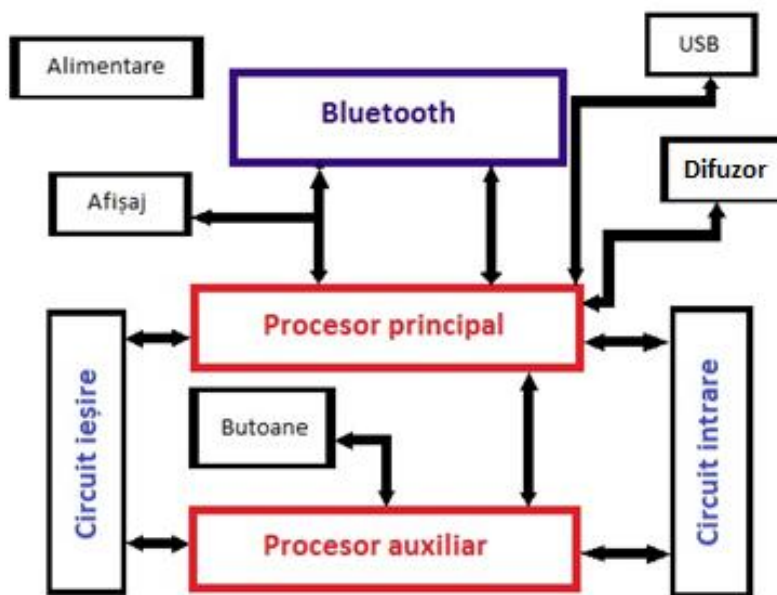


Fig.19 Diagrama sistemului de procesare

### **2.4.1. Ecranul LCD**

La caramida NXT ecranul LCD este mult mai mare decât procesorul său. El este o matrice monocrom de 100x64 pixeli suficient pentru a afișa opt linii de text cu 16 caractere pe linie. Firmware-ul asigură textul, precum și sprijinul la nivel de pixel pentru a face un desen pe acest ecran. Se poate specifica, de asemenea, numele unui fisier-pictogramă care poate fi pus ca o imagine pe ecranul NXT-ului. În funcție de mediul de programare în care se lucrează există mai multe sau mai puține funcții de afișarea textului pe ecran. Având aceste funcții se poate alege un font mare sau mai mic, afișarea acestui text în orice punct al ecranului, de a desena linii, dreptunghi, cercuri, elipse.

Se poate crea de asemenea, propriile fișiere și încarcate pe NXT. În cadrul programului NXT se poate crea astfel de fișiere care oferă numeroase posibilități pentru implementa diferite jocuri. Funcțiile de afișare NXT sunt utile pentru a face grafice și diagrame.

### **2.4.2. Interfața digitală și Bluetooth**

Pentru a comunica cu senzorii, motoarele și alte dispozitive periferice atașate la NXT, sistemul a introdus interfațe I2C și USB, dar pentru comunicații fără fir NXT are o interfață Bluetooth. Interfața USB este utilizată pentru a comunica cu calculatorul și I2C este utilizat pentru a comunica cu senzorii și motoarele. Software-ul NXT vine cu driverele USB care face din NXT un dispozitiv “Plug and Play”.

Sistemul USB constă dintr-un dispozitiv gazdă care poate fi conectat la mai multe dispozitive periferice, iar unul periferic se poate conecta doar cu o singură gazdă. Interfața I2C cunoscută sub numele de IIC este utilizată într-o gamă foarte largă de dispozitive, chiar în unele articole de uz casnic. Un exemplu frecvent a utilizării acestor dispozitive sunt telefoanele mobile.



















### **2.4.3. Comunicație Bluetooth**

Față de interfețele cu fir prezentate anterior, NXT are o interfață fără fir. Aceasta are o îmbunătățire majoră față de interfața infraroșu, în ceea ce privește viteza, consumul de energie și standardizare. Bluetooth-ul dintre NXT-uri poate opera în oricare dintre modulele “master” sau “slave”, în schimb la conectarea NXT cu PC, NXT va fi mereu “slave”. Conexiunea Bluetooth se

poate face reciproc între patru NXT, unul fiind “master” și ceilalți ”slaves”. Pentru ca NXT-urile să poată comunica între ele trebuie ca fiecare NXT să aibă un nume unic.

## 2.4.4. Sistemul de operare

Aproximativ jumătate din cele 256KB din memoria flash este utilizată pentru a stoca sistemul de operare al NXT-ului. NXT are propriul sistem de operare, la fel ca un PC care are un sistem de operare Windows sau Linux. Acest sistem administrează toate dispozitivele (senzorii, motoarele, butoanele, ecranul LCD, memoria, timpii) care fac parte din NXT. Software-ul ce este scris într-un program anume are instrucțiuni pe care procesorul NXT nu le poate executa. Pentru aceasta sistemul are o aplicație “interpret” care convertește instrucțiunile într-un cod funcțional. Sunt foarte multe programe create pentru unitatea centrală NXT. Software-ul de programare standard este NXT-G bazat pe programare grafică LabVIEW. Dar se pot folosi și alte programe software, ca de exemplu:

-  LabVIEW Toolkit
-  Lego NXT
-  Ada
-  BricxCCNext Byte Codes & Not eXactly C
-  ROBOTC
-  RoboMind
-  Enchanting
-  nxtOSEK
-  ICON
-  LibNXT
-  NXTGCC
-  URBI
-  leJOS NXJ
-  MATLAB și Simulink
-  Lua
-  FLL NXT Navigation
-  ruby-nxt
-  Robotics.NXT

- ✚ C\_NXT
- ✚ PyNXT
- ✚ NXT-Python
- ✚ LEGO Mindstorms EV3 Software
- ✚ Physical Etoys

Ca mediu de dezvoltare software am folosit programul RobotC.

## 2.4.5. RobotC

Acest program utilizează un limbaj de programare C care este standard în industrie fiind cel mai popular și cel mai utilizat limbaj de programare. RobotC a fost dezvoltat de către Academia de Robotica de la Universitatea Carnegie Mellon. Programarea în RobotC este conceptual la fel ca și programarea în NXT-G.



Fig.20 Pornirea programului RobotC

Se scrie programul pe PC și după ce este complet se compilează acest program, după care este descărcat pe NXT. O dată ce sa descărcat programul pe modulul NXT se poate rula sau executa. Diferența între NXT-G și RobotC este că NXT-G este un limbaj de programare grafic în timp ce RobotC este un limbaj de programare bazat pe text. RobotC are capacitatea de a urmări în mod automat, pe PC, executarea programului. Se poate suspenda temporar programul pentru a vedea execuția logică sau o line de cod la un moment dat. Acest program are de asemenea avantajul că, fiind un limbaj textual, acesta are o reprezentare foarte compactă.

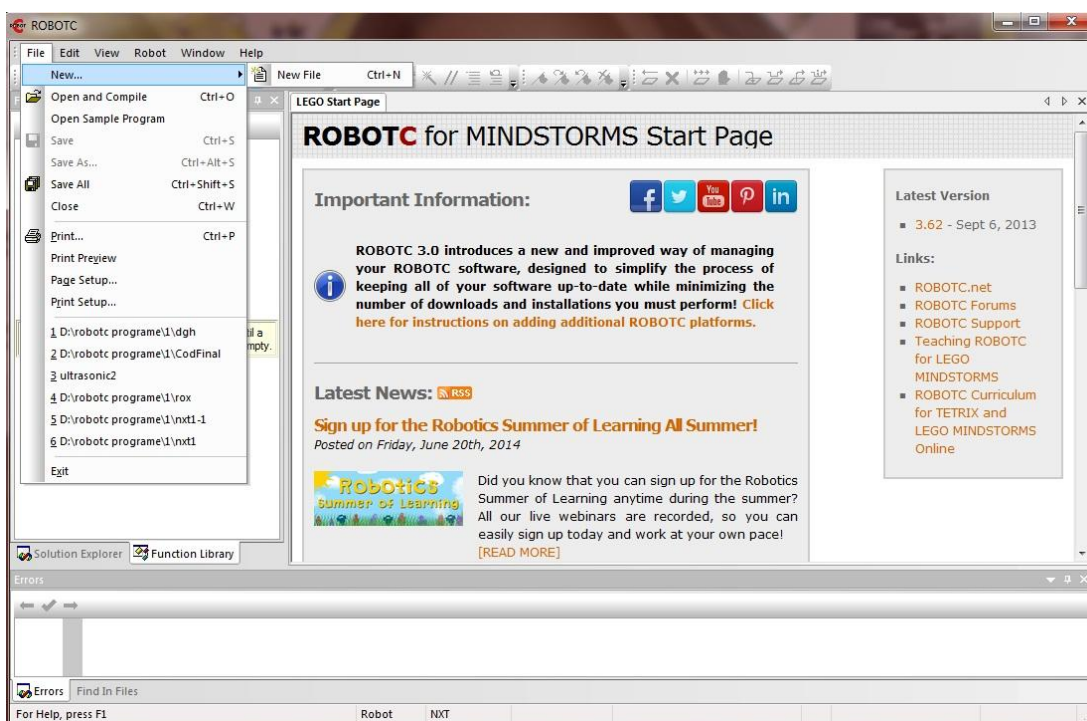


Fig.21 Deschiderea unei ferestre pentru a programa

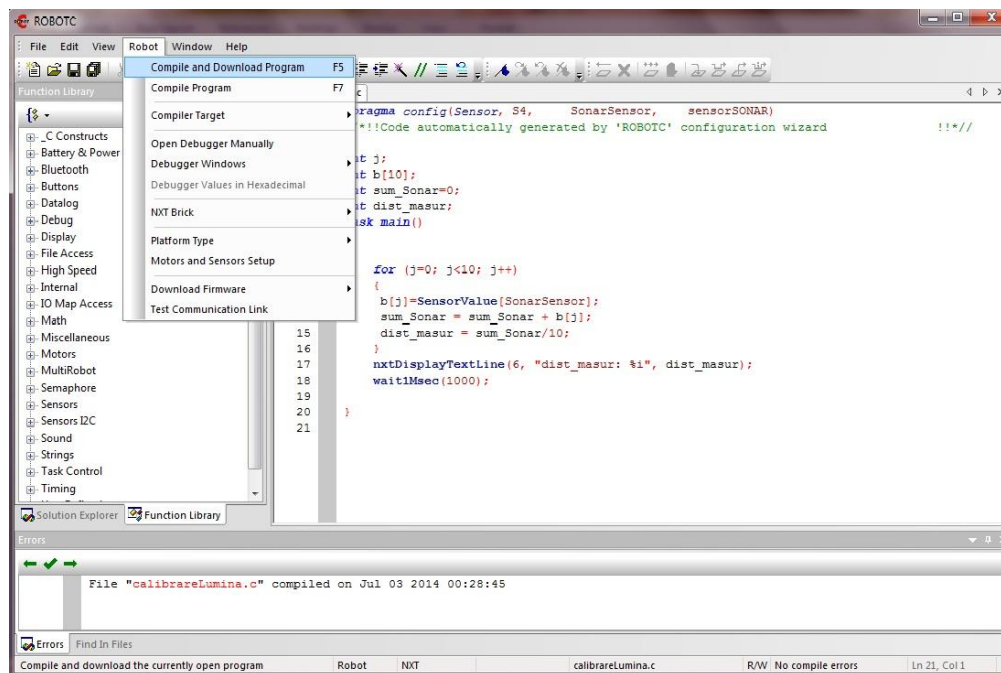


Fig.22. Compilarea și descărcarea programului pe chitul robotic

## 2.5.Conectori

În acest set avem două tipuri de conectori care sunt:

- ✚ Cabluri de conexiune care ne permit să conectăm senzorii și servomotoarele la porturile unității de control NXT. Ele au ca terminație conectori modulatori cu șase fire de tip RJ12.
- ✚ și un cablu USB cu care permite să conectăm unitatea de control NXT la portul USB al calculatorului pentru a transmite date, adică să descărcăm programul scris dar în primul rând Firmware-ul.

## Capitolul 3. Partea practică

În această lucrare vor fi prezentate etapele de proiectare și de implementarea unui robot ce trece peste obstacole și menține senzorul giroscop, ce este poziționat pe o platformă, la aproximativ 90° față de suprafața pământului pe care se află senzorul.

Etapele ce se vor parcurge în continuare:

1. Necesarul de a construi acest robot;
2. Partea mecanică a robotului;
3. Cum funcționează acest robot;
4. Interpretarea datelor pe calculator;
5. Pseudocodul;

### 3.1. Necesarul de a construi acest robot

Pentru ca să fac acest robot eu am folosit piesele Lego Mindstorms și anume:

- ✚ Două unități de procesare, una din ele fiind master iar cealaltă slave
- ✚ Cinci servomotoare, unde vom folosi un servomotor pentru giroscop și patru pentru platformă;
- ✚ Un senzor giroscop;
- ✚ Un senzor ultrasonic;
- ✚ Opt roti;
- ✚ Patru sinile;
- ✚ Cabluri de transmiterea datelor;
- ✚ Piesele de legătură;

În continuare vom folosi aceste piese pentru construcția robotului.



### 3.2. Partea mecanică a robotului

Pentru început vom lua două motoare, o unitate de procesare, patru roti, două șinile și piese de a lega toate aceste componente ca să obținem figura de mai jos:



Fig. 23 Fața blocului



Fig. 24 Blocul în lateral

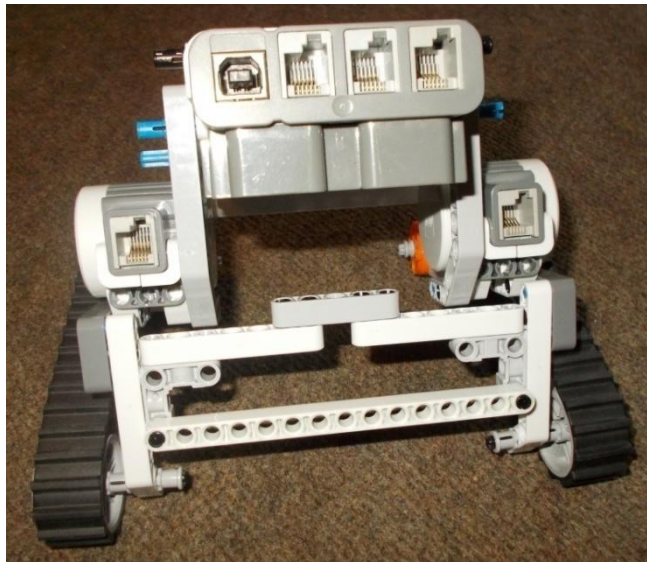


Fig. 25 Spatele blocului

Aceiași construcție o facem încă o dată pentru a obține două construcții pe care va trebui să așezăm platforma pe care se află senzorul giroscop. Doar că una din construcții care se va numi master vom așeza senzorul ultrasonic jos sub unitatea de procesare după cum e în fig.27.



Fig.26 Cele doua blocuri ale robotului

Am ales această construcție deoarece doresc ca aceste două blocuri să poată urca sau coborâ trepte sau alte obstacole întâlnite. La fiecare șină am pus câte un motor pentru a controla fiecare șină în parte, adică pentru a putea face viraj la stânga sau la dreapta.

Construcția care are în componența sa și un senzor ultrasonic va fi pus cu fața în față, pentru ca acest senzor să poată detecta obstacolele. Însă cel de-al doilea bloc va fi montat cu invers față de primul bloc, bineînțeles că motoarele acestui bloc vor merge cu o viteză negativă față de motoarele celui de-al doilea bloc.



Fig.27 Senzorul ultrasonic de pe primul bloc

Mai departe voi construi platforma unde ne va trebui piese de legătură, un servomotor și senzorul giroscop. Platforma va fi de o lungime dublă față de lungimea axului pe care este montat giroscopul și fixat de motor.

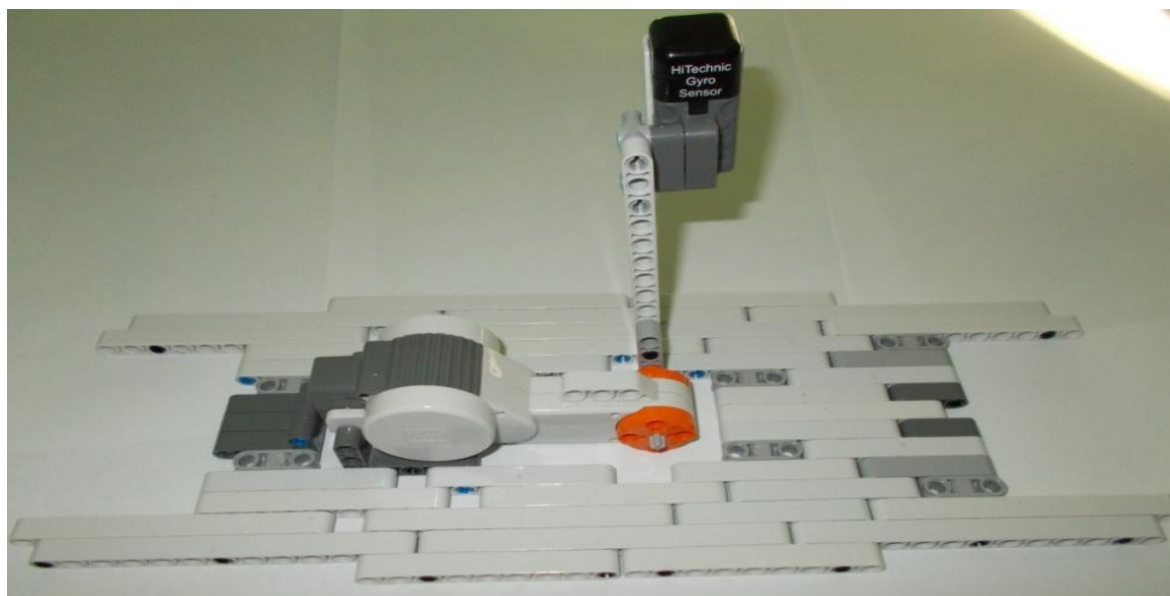


Fig.28 Platforma robotului

Lungimea acestei platforme am ales-o în așa mod, fiindcă în cazul în care senzorul giroscopic montat pe axă va trebui să fie mișcat de motor pentru a fi perpendicular pe suprafața pământului să nu fie oprit de alte piese ale robotului. Acum vom uni cele două blocuri construite mai sus din fig.26 împreună cu platforma din fig.28. După care vom fixa cablurile de legătură între



servomotoare și unitatea de procesare, senzori și unitatea de procesare. Și vom obține următorul robot:

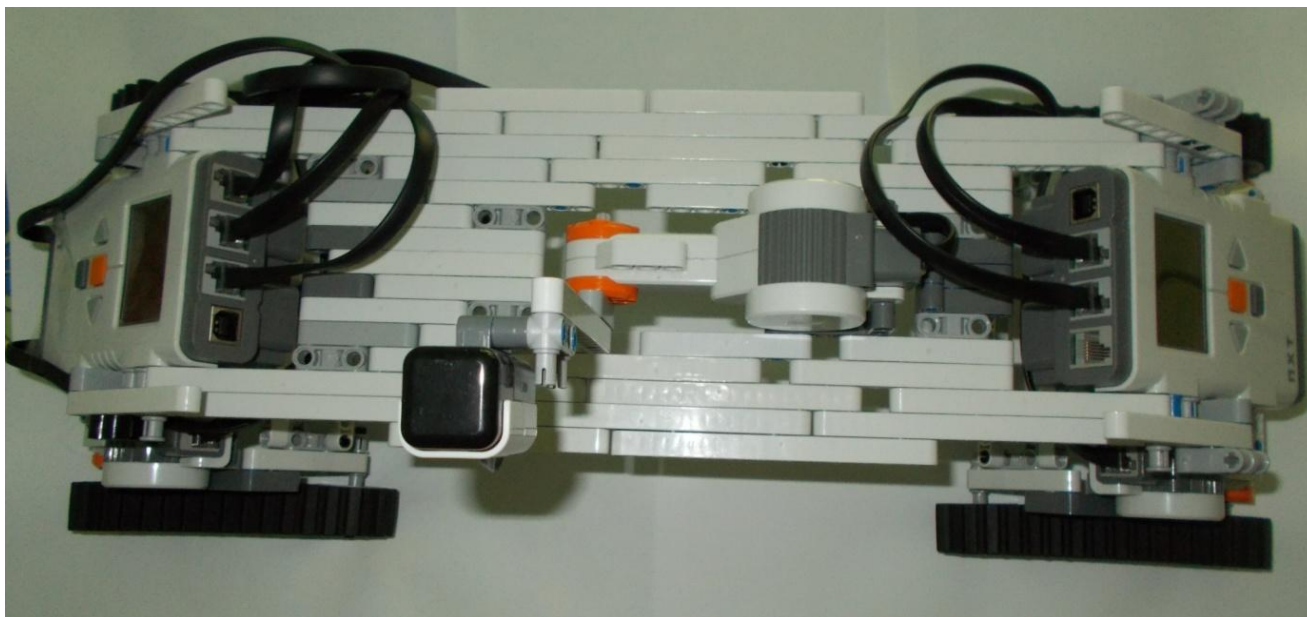


Fig.29 Robotul final

### 3.3.Cum funcționează acest robot

În această lucrare am dorit să fac un robot care să urce sau să coboare diferite obstacole. Pentru a ușura urcarea sau coboarărea acestor obstacole am folosit șinile.

După cum am menționat mai sus, vom avea două construcții, care una din ele va fi master iar cealaltă va fi slave. Masterul va avea în construcția sa pus și un senzor ultrasonic care va detecta obstacolele.

Mai avem un senzor giroscopic care este plasat pe platforma așezată între cele două construcții, master și slave. Indiferent de situație, că urcă sau coboară robotul, acest senzor trebuie să stea la aproximativ  $90^0$  față de suprafața pământului. Servomotorul care este la fel plasat pe această platformă, va menține acest senzor în plan perpendicular față de suprafața pământului.

Acest senzor măsoară vîrta unghiulară, cu alte cuvinte măsoară cât de repede se rotește senzorul. El returnează o valoare care reprezintă numărul de rotații în grade pe secundă. În această aplicație senzorul giroscop returnează o valoare atunci când robotul este în mișcare și transmite

aceste valori la unitatea de procesare master, dupa care aceasta la rândul său transmite informațiile necesare la motorul care ține acest giroscop, pentru al aduce la poziția verticală față de axa pământului pe senzor. Pentru a putea aduce senzorul giroscop la poziția cerută trebuie facute câteva calcule.

În primul rând când senzorul giroscop nu se rotește, adică nu este în mișcare, returnează o valoare de aproximativ 600. Această valoare se numește offset.

Pentru a afla rata de rotație a senzorului trebuie să scădem acest offset din valoarea ce o returnează giroscopul. Să presupunem că offsetul este 580, iar valoarea pe care o returnează senzorul giroscop este 670, deci senzorul are o rotație în sensul acelor de ceasornic de 90 de grade pe secundă. Dar acest giroscop este la fel ca orice alt senzor un pic zgomotos, în realitate el are o rotație de 89 sau 91 de grade pe secundă.

Offset-ul ce trebuie aflat este diferit pentru fiecare senzor în parte și se poate schimba din cauza unor factori externi. Din acest motiv, este foarte important să se determine valoarea offset-ului înainte de al utiliza în diferite aplicații. Valoarea care este citită de la senzorul giroscop, atâta timp cât acesta nu este mișcat, este offset-ul.

După cum am menționat mai sus, că apar unele zgomote la citirea giroscopului. Pentru a elimina acest zgomot nu ar trebui să ne bazăm pe o singură citire a senzorului, dar trebuie să luăm în considerare media aritmetică a mai multor citiri ale senzorului. Giroscopul întoarce valori întregi, dar în realitate, rata de rotație nu este un întreg. Valorile citite de la senzor sunt transformate din semnale analogice, de către NXT, în valori digitale de tip intreg. Aceasta face ca la fiecare lectură a giroscopului, să apară o mică eroare.

Offset-ul nu este o valoare constantă fiindcă poate fi afectată de către temperatură sau de tensiune. Acest lucru este numit derivă. Pe termen lung, acest lucru va afecta calitatea informațiilor de la giroscop. De aceea trebuie corectată această eroare.

Aceasta are următorul calcul: se citește senzorul giroscop de  $n$  ori, după care se face o medie a tuturor valorilor citite, acesta fiind offset-ul

$$\text{Offset} = (\sum \text{n. citiri ale senzorului})/n$$

adevarata valoare a zenzorului este:

$$\text{Valoarea reala a senzorului} = \text{Valoarea citită de la senzor} - \text{Offset}$$

Revenind la senzorul ultrasonic care este așezat la unitatea de procesare master și care ne ajută să calculăm distanța la un eventual obstacol. Senzorul citește aceste distanțe până la obiectul respectiv prin trimiterea unui semnal acustic de către emițător. O dată ajunsă această undă la obiect, ea este reflectată și recepționată de către receptorul senzorului. Timpul de dus și de întors a acestei unde determină distanța față de acest obiect. Distanța ce este calculată de acest senzor ne va fi de folos mai departe la controlul motoarelor ambelor blocuri.

Primul bloc, adică master este o unitate de procesare principală care comunică prin Bluetooth cu cealaltă unitate de procesare, adică slave. Ca să putem face comunicația prin Bluetooth a acestor două unități trebuie să-i dăm un nume fiecărei unități. Numele masterului este “NXT1”, iar pentru slave “NXT2” .

Am realizat această comunicare între unități ca să pot controla puterea specifică pentru mers a servomotoarelor. Datele citite de la senzorul ultrasonic sunt procesate de unitatea de control și totodată transmise aceste date și unității de procesare secundare. Informațiile ajunse la cea de-a doua unitate de procesare, ne ajută să controlăm servomotoarele în momentul când senzorul ultrasonic depistează un obstacol. La o distanță mai mică de 15cm față de obstacol, prima unitate va face ca motoarele ei să meargă cu o putere mai mare pentru ca să poată urca obstacolul întâlnit. În schimb unitatea a doua dă comandă servomotoarelor să meargă cu o putere mai mică, pentru a nu forța prea mult urcarea primului bloc. Dacă întradevar vom forța urcarea primului bloc s-ar putea dezmembra șinelele.

În următoarea figură vedem cum se transmit datele de la unitățile de procesare și chiar comunicarea Bluetooth între cele două unități.

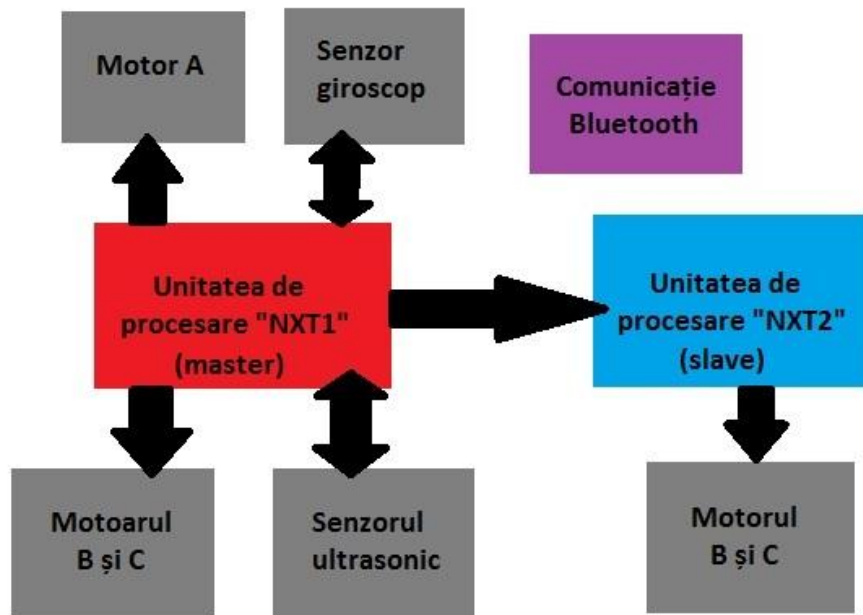





Fig.30 Comunicarea datelor între elementele robotului

### 3.4.Înterpretarea datelor pe calculator

Limbajul de programare folosit pentru scrierea acestei aplicații este RobotC. Pentru ca să putem lucra în acest mediu trebuie mai întâi să descărcăm Firmware-ul pe unitatea de control NXT, ca să putem pune robotul în funcție. Pentru ca să descarci acest program pe unitate este necesar de a conecta această unitate la calculator cu cablul USB, după care se va selecta din meniul programului RobotC [Robot](#) --> [Download Firware](#) --> [Standard File](#). Descărcarea nu va dura foarte mult și după ce a terminat, unitatea de control se va aprinde singură după care poți să programezi în acest program.

Acest program are trei nivele de selecție a funcțiilor pentru librărie. Aceste nivele sunt:

-  Basic
-  Expert
-  Super User

Primul nivel Basic conține noua fișiere de unde poți alege funcția necesară și acestea sunt:

\_C Constructs, Battery & Power Control, Bluetooth, Math, Motors, Sensors, Sound, Timing, User Defined.

Cel de-al doilea nivel are un pic mai multe fișiere având posibilitatea de a face programe mai complexe. Pe lângă fișierele care le are primul nivel cel de-al doilea nivel mai are în plus următoarele fișiere: Buttons, Datalog, Debug, Display, File Access, Miscellaneous, MultiRobot, Semaphore, Sensors I2C, Strings, Task Control.

Ultimul nivel, adică cel de-al treilea are cu trei fișiere mai mult ca cel de-al doilea nivel și acestea sunt: High Speed, Internal, IO Map Access.

Dacă dorim să testăm codul scris ca să vedem ca funcționează, adică nu are nici o eroare selectăm **Robot** → **Compile Program**, dar dacă dorim să verificăm codul în același timp să se și descarce pe unitatea de procesare, dacă este fără erori desigur atunci selectăm **Robot** → **Compile and Download Program**.

Din cauză că folosesc două unitați de procesare pentru a conduce acest robot voi avea un software compus din două programe. Primul program va fi mai complex și este scris pentru unitatea de procesare master și cel de-al doilea program, pentru unitatea de procesare slave.

Deci mai întâi voi descrie functionarea primului program, adică pentru blocul master.

Înainte de a scrie programul, în primul rând trebuie să configurăm senzorii și motoarele care vor fi utilizați:

```
#pragma config(Sensor, S1, Gyro_Sensor, sensorI2CHiTechnicGyro)
#pragma config(Sensor, S4, Sonar_Sensor, sensorSONAR)
#pragma config(Motor, motorA, Gyro_Motor, tmotorNormal, PIDControl, encoder)
#pragma config(Motor, motorB, Right_Motor, tmotorNormal, PIDControl, encoder)
#pragma config(Motor, motorC, Left_Motor, tmotorNormal, PIDControl, encoder)
//**Code automatically generated by 'ROBOTC' configuration wizard **//
```

Prima linie de cod este configurarea senzorului giroscop unde am ales portul S1 pentru al programa. Am dat nume acestui senzor Gyro\_Sensor și l-am ales de tip Hi Technic Gyro.



Aceasta linie “#pragma config(Sensor, S4, Sonar\_Sensor, sensorSONAR)” este configurarea senzorului ultrasonic. Pentru al programa am ales portul S4, cu un nume Sonar\_Sensor și este de tip Sonar.

Următoarele trei rânduri sunt configurații ale servomotoarelor și au următoarele nume Gyro\_Motor pe care l-am folosit pentru a menține și controla senzorul giroscop, Right\_Motor care este motorul drept al constructorului nostru și Left\_Motor care este motorul stâng al constructorului.

Aceste linii de cod sunt date automat de către programul RobotC, doar că trebuie bifate aceste configurații și le găsim în [Robot--> Motor and Sensor Setup](#).

Mai întâi de toate, înainte de a începe programul principal sunt declarate variabilele globale:

```
//---declararea variabilelor globale-----  
float grade=0;  
  
int Gyro_value;  
int i;  
int a[50];  
int sum_Gyro=0;  
  
int dist;  
int d;  
int j;  
int n;  
int b[50];  
int offset;  
//-----
```

Declararea acestor variabilele s-a făcut la începutul programului, astfel încât programul să știe despre ele înainte de a începe să execute bucla principală. Unele din aceste variabile sunt inițializate chiar când am declarat acestea. Gradele și sum\_Gyro (care este suma tuturor valorilor citite de la giroscop) le-am inițializat cu zero, doar ca gradele le-am declarat de tip real iar suma giroscopului de tip întreg.

Am declarat de tip întreg valoarea giroscopului (Gyro\_value), distanța până la obiect (dist), doua valori i și d care sunt folosite pentru citirea a doi vectori, a și b. Acești vectori vor stoca 50 de valori ale senzorului giroscop, respectiv ale senzorului ultrasonic.

După ce am declarat variabilele, va urma programul principal care va conține:

- + Calibrarea giroscopului
- + Comanda de viraj
- + O buclă, unde vom avea:
  - Afișarea valorilor senzorului giroscop
  - Controlul giroscopului
  - Distanța față de obiect și afisarea pe ecran
  - Comunicarea valorilor distantei celui alt NXT
  - Ștergerea valorilor

### 3.4.1. Calibrarea giroscopului

Este necesară pentru a afla valoarea reală a giroscopului, adică a offset-ului după cum urmează în codul următor.

```
//--calibrarea giroscopului-----  
for (i=0; i<50; i++)                //  
{                                    //  
    a[i]= SensorValue[Gyro_Sensor]; //  
    sum_Gyro = sum_Gyro + a[i];      //  
}                                    //  
    offset = sum_Gyro / 50;          //  
                                    //  
//-----
```

Înainte de a da start execuției programului în sine trebuie să calibrăm senzorul giroscop, căci cum am menționat anterior, senzorul giroscop trebuie citit în starea nemișcată înainte de a fi folosit.

În această buclă se iau 50 de valori citite de la senzorul giroscop, când acesta este nemiscat și făcută o medie aritmetică. Media aritmetică este offset-ul. Acestă valoare adică offset-ul va fi folosit mai departe pentru cunoașterea adevăratei valori ale senzorului giroscop. La un moment dat, s-ar putea să nu mai avem senzorul giroscop chiar atât de perpendicular față de suprafața pământului, deoarece apare o eroare de la motor. Această eroare ce apare de la motor nu este de la codul scris

sau citirea lui, dar de la partea mecanică a motorului, mai exact de la axul portocalui (fig.13) unde este fixată axa giroscopului.

### 3.4.2.Comandă de viraj

Folosit motor la fiecare şină în parte, putem face şi vedea cum robotul face viraj la stânga sau la dreapta, pentru aceasta am făcut două bucle. Prima buclă este pentru virajul la stânga şi are următoarele linii de cod:

```
//---comanda de viraj la stanga-----  
for(j=0; j<50; j++)                //  
{                                    //  
    nxtDisplayBigTextLine(1, "j=%i", j); //  
    motor[Right_Motor]=50;           //  
    motor[Left_Motor]=30;            //  
    wait1Msec(100);                  //  
}                                     //  
eraseDisplay();                      //  
//-----
```

Această buclă este scrisă pentru 50 de valori ale variabilei j. Linia trei şi patru din cod comandă pornirea motoarelor cu o anumită putere. Motorul drept (Right\_Motor) va merge cu o putere de 50%, iar cel de la stânga (Left\_Motor) cu o putere de 30%. La fiecare încrementare a lui j apare un timp de aşteptare de 100 milisecunde.

Pentru a face viraj la stânga i-am dat putere mai mare motorului din dreapta, însă pentru virajul la dreapta vom da putere mai mare motorului din stânga. Puterea de control a celor două motoare va fi inversă pentru cea de-a doua buclă, adică motorul drept va merge de data aceasta cu o putere de 30%, iar cel stâng cu 50%. După terminarea fiecărei bucle sunt şterse valorile lui j şi a lui n de pe ecranul unităţii de control. Mai jos este prezentat codul pentru virajul la dreapta.

```

//---comanda de viraj la dreapta---
for(n=0; n<50; n++)           //
{                               //
    nxtDisplayTextLine(2, "n=%i", n); //
    motor[Right_Motor]=30;      //
    motor[Left_Motor]=50;      //
    wait1Msec(100);            //
}                               //
eraseDisplay();                //
//-----

```

La ambele bucle putem vedea pe ecranul unității de control încrementarea variabilelor j și n, pe linia unu respectiv pe linia doi a ecranului. Afișarea se face cu ajutorul funcției `nxtDisplayTextLine`, specificând mai întâi linia, apoi formatul textului ce trebuie să apară pe ecran și în cele din urmă adresa textului.

### 3.4.3. Afișarea valorilor senzorului giroscop

Liniile de cod ce urmează reprezintă afișarea datelor de la senzorul giroscop:

```

//---afisarea valorilor senzorului giroscop-----
nxtDisplayTextLine(1, "offset: %i", offset);           //
nxtDisplayTextLine(2, "SensorValue: %i", SensorValue[Gyro_Sensor]); //
Gyro_value=SensorValue(Gyro_Sensor)-offset;           //
nxtDisplayTextLine(3, "Gyro_value: %i", Gyro_value);   //
grade=grade+Gyro_value*.001;                           //
nxtDisplayTextLine(4, "grade: %.3f", grade);           //
wait1Msec(100);                                         //
//-----

```

După ce am calibrat senzorul giroscop putem afișa pe ecranul unității de control, valorile calculate sau citite de la senzor. Pe linia 1 se va afișa calculul ce s-a făcut la calibrare pentru offset, iar pe linia 2 este afișată valoarea ce este citită de la senzor nefiind în stare de repaus. Aceste două valori sunt folosite pentru a calcula valoarea reală a giroscopului, adică din valoarea returnată de senzor pus în funcțiune se scade valoarea offset-ului (linia 3 din cod), după care aceasta este afișată pe ecran pe linia 3.

Pentru a vedea câte grade are valoarea reală a senzorului giroscop într-o milisecundă, fiindcă valoarea returnată de senzorul giroscop este în grade pe secundă, atunci valoarea senzorului într-o milisecundă este:

$$\text{Valoarea reală a senzorului} \cdot 0,001 = \text{grade}$$

Am afișat această valoare pe linia 4 a ecranului. Calculul acestei valori l-am folosit pentru a aduce giroscopul la înclinarea care trebuie. Aplicând aceasta în practică nu dădea un rezultat prea bun, adică la un moment dat eroarea devenea prea mare pentru a o corecta. De aceea am folosit valoarea reală a senzorului în grade pe secundă pentru a aduce axul giroscopului la starea care trebuie.

### 3.4.4. Controlul giroscopului

Valoarea citită de la senzorul giroscop este trimisă unității de control, unde se calculează valoarea reală a lui. În funcție de această valoare trebuie să se de-a comandă motorului ce menține giroscopului pe o axă. Inițial poziția axului pe care este poziționat giroscopul este perpendicular pe suprafața pământului și cu capul negru în sus.

După ce am poziționat perpendicular putem vedea ce valori ne returnează senzorul, atunci când se mișcă robotul. Acesta valoare citită poate fi negativă sau pozitivă, în funcție de sensul mișcării acestui senzor și cât de repede este mișcat. Inițial când am poziționat senzorul vertical, fiind nemișcat, avem o valoare de zero grade pe secundă. La mișcare însă dacă avem o valoare negativă, va trebui să adunăm această valoare cu semn schimbat ca să aducem senzorul în poziția zero. Dar dacă vom avea valori pozitive, va trebui să scădem aceiași valoare pentru a obține la fel poziția zero.

Ca să scădem sau să adunăm valoarea necesară pentru a aduce senzorul pe poziția perpendiculară, ne vom folosi de motorul ce ține giroscopul. Când giroscopul va avea o oarecare valoare diferită de zero, atunci motorul va duce senzorul în sensul invers a acestei valori.

Pentru a controla giroscopul de la robotul meu am folosit următorul cod:

```

//---controlul giroscopului-----
if (Gyro_value > 10)                //
    motor[ Gyro_Motor]=3;           //
wait1Msec(50);                      //
if (Gyro_value <-10)                //
    motor[ Gyro_Motor]=-3;          //
wait1Msec(50);                      //
if (Gyro_value<10 && Gyro_value>-10)//
    motor[ Gyro_Motor]=0;           //
wait1Msec(50);                      //
//-----

```

În prima linie de cod am menționat că dacă valoarea giroscopului este mai mare decât 10, atunci motorul ce menține senzorul (Gyro\_Motor) să pornească în sensul opus valorii cu o putere de 3%. Dacă valoarea giroscopului este mai mică decât -10, atunci motorul va porni la fel în partea opusă cu o putere de -3%. Pentru valorile cuprinse în intervalul [10,-10] motorul se va opri din funcționare.

Am ales o putere de 3% sau -3% pentru motor, deoarece o putere mai mare de control va mișca mai repede axul pe care este poziționat senzorul, adică senzorul va avea o viteză mai mare de mișcare (de rotație), ceea ce va duce la valori mult mai mari ale senzorului, la un control nu tocmai prea bun și la erori mai mari. O valoare mai mică decât 3% pentru puterea motorului la fel nu este prea bună, fiindcă motorul se mișcă prea încet și nu reușește să readucă cât decît senzorul la poziția cerută.

Când senzorul este nemișcat uneori citește o valoare diferită de zero, aceasta fiind o mică eroare. Pentru eliminarea acestei erori am testat practic de mai multe ori și am decis că 10 este o valoare mai bună pentru compararea valorii giroscopului.

### 3.4.5.Distanța față de obiect și afișarea

Calcularea distanței față de obiect se face cu senzorul ultrasonic. Am folosit acest senzor cu scopul de a obține date în privința distanței și de a folosi aceste date mai departe la comunicarea Bluetooth. Senzorul ultrasonic poate măsura până la 255cm, dar are o marjă de eroare, precum și la distanțe mai mici de patru centimetri el determină valori foarte mari, care nu sunt reale. Cel mai bun

interval de calcul a distanței este cuprins între 10 și 20 centimetri. De aceea în codul de mai jos vom vedea cum s-a făcut calculul distanței:

```
//---distanța fata de obiect si afisarea---  
int sum_dist=0; //  
for(d=0; d<50; d++) //  
{ //  
    b[d]=SensorValue(Sonar_Sensor); //  
    sum_dist=sum_dist+b[d]; //  
} //  
dist=sum_dist/50; //  
nxtDisplayTextLine(5, "dist= %i", dist); //  
wait1Msec(300); //  
//-----
```

În prima linie a acestui cod este inițializată suma tuturor distanțelor citite de la senzor cu zero. Pentru a fi siguri de valoarea returnată de senzor, îl vom citi de 50 de ori.. Liniile 2, 3, 4, 5, 6 ale codului alcătuiesc o buclă unde sunt stocate 50 de valori citite ale senzorului. La sfârșitul acestei bucle este făcută media tuturor valorilor citite de la senzor și afișată pe linia cinci a ecranului. Am ales să fac această medie, fiindcă este foarte important de a ști distanța exactă a obstacolului. Media aritmetică a acestor distanțe este folosită mai departe la controlul motoarelor a ambelor unități de control.

### 3.4.6.Comunicarea volorii distanței celuiilalt NXT

Pentru a comunicarea ambelor unități de control între ele este necesar de a activa Bluetooth-ul. Una dintre unități trebuie să conducă și cealaltă unitate de control. Unitatea principală este masterul care are numele “NXT1” transmite mesaje prin Bluetooth celeilalte unități de control slave cu numele “NXT2”. Înainte de a trimite mesaj de la NXT1 celuiilalt NXT2 trebuie specificat câte mesaje vor fi transmise și care este mesajul respectiv. Conectarea Bluetooth-ului se face la un port menționat în codul primului NXT, cât și la celălalt NXT. În codul pricipal trebuie menționat numele prietenului cu care se face conexiunea Bluetooth, adică numele celui care îi trimiți mesajul respectiv.

În continuare sunt prezentate liniile de cod a conexiunii Bluetooth dintre NXT1 și NXT2:

```

//---comunicarea valorii distantei celuiilalt NXT-----
ubyte mesaj_Trimis1[1]={dist}; //
btConnect(1, "NXT2"); //
cCmdMessageWriteToBluetooth(mesaj_Trimis1, 1, mailbox1); //
if (dist > 0 ) //
{ //
    motor[Right_Motor]=70; //
    motor[Left_Motor]=70; //
} //
//-----

```

După ce s-a calculat media aritmetică a distanței dintre robot și obstacol, este trimisă prin Bluetooth de la NXT1 la NXT2. Această valoare a distanței este folosită și de unitatea secundară pentru controlul motoarelor. Dacă distanța calculată este mai mare decât zero, atunci motorul drept și cel stâng vor merge cu o putere de control de 70%. De la o putere de 50% sa trecut la o putere de 70% , deoarece o dată ce robotul a întâmpinat un obstacol trebuie să îl ridice și îi trebuie o putere mai mare pentru a urca mai ușor.

### 3.4.7.Ștergerea valorilor

Ștergerea valorilor de pe ecranul unității de control, are linia de cod de mai jos:

```

//---ștergerea valorilor--
eraseDisplay(); //
// -----

```

La începutul programului această funcție este folosită la ștergerea valorilor j și n ce au fost afișate pe ecran în timpul virajului la stânga sau la dreapta, pentru ca să putem afișa alte valori pe ecran. Însă la sfârșitul buclei principale, funcția va șterge toate calculele și valorile ce au fost făcute în această buclă, pentru a putea relua toate aceste calcule de la începutul acestei bucle.

Soft-ul unității a doua, adică slave sau NXT2 va începe la fel cu declararea de data aceasta doar a motoarelor, fiindcă la această unitate nu avem conectat nici un senzor.

În liniile de cod de mai jos sunt pecificate motoarele care sunt folosite și ce nume au.



```
#pragma config(Motor,  motorB,           Right_Motor,  tmotorNormal, PIDControl, encoder)
#pragma config(Motor,  motorC,           Left_Motor,   tmotorNormal, PIDControl, encoder)
/*!!Code automatically generated by 'ROBOTC' configuration wizard    !!*/
```

Pe prima linie este configurat motorul B cu numele “Right\_Motor”, adică motorul drept și “Left\_Motor”, adică motorul stâng.

Unitatea a doua fiind controlată de către cealaltă unitate nu va avea declarate multe variabile globale. Variabilele declarate în acest soft sunt:

```
int i;
int m;
```

Ambele variabile sunt folosite în două bucle diferite.

Acest soft va urmări următorii pași:

- ✚ Comanda de viraj pentru NXT2;
- ✚ Recepționarea mesajului de la NXT1;
- ✚ Controlul motoarelor în funcție de mesajul primit;

### 3.4.8.Comanda de viraj pentru NXT2

Comanda de viraj pentru NXT2 este aproximativ la fel ca cea a virajului pentru unitatea de control NXT1. Mai jos avem codul virajului la stânga.

```

//---comanda de viraj la stanga----
for(i=0; i<50; i++)           //
{                               //
    motor[Right_Motor]=-30;    //
    motor[Left_Motor]=-50;     //
    wait1Msec(100);           //
}                               //
//-----

```

Diferența acestor linii de cod este că avem de data aceasta alte variabile pentru aceste două bucle, acestea fiind i și m. Ca și la programul principal am creat două bucle cu comandă de viraj la stânga și la dreapta, la fel pentru 50 de incrementări ale variabilelor. Având 50 de incrementări pentru variabile i, j, m, n ale ambelor programe determină sincronizarea ambelor blocuri, adică a robotului în sine. Virajul la dreapta are următorul cod:

```

//---comanda de viraj la dreapta----
for(m=0; m<50; m++)           //
{                               //
    motor[Right_Motor]=-50;    //
    motor[Left_Motor]=-30;     //
    wait1Msec(100);           //
}                               //
//-----

```

Deosebirea virajului de la NXT2 față de cel de la NXT1 este puterea cu care este controlat fiecare motor. După cum se observa în figura 29, blocurile robotului sunt poziționate invers unul față de altul. Cât timp blocul principal va merge cu o putere pozitivă de control a motoarelor celălalt bloc secundar va avea o putere negativă de control, deoarece ambele blocuri trebuie să meargă în aceeași direcție. O dată ce am inversat blocul al doilea față de primul, vom avea schimbate și puterile între cele două motoare, adică motorul drept cu cel stâng a unității de control NXT2.

Motorul drept (Right\_Motor) în acest soft va avea o putere de -30% și cel stâng (Left\_Motor) o putere de -50% pentru un viraj la stânga, iar pentru un viraj la dreapta motorul drept va avea de data aceasta o putere de -50% și cel stâng -30%.

### 3.4.9.Recepționarea mesajului de la NXT1

După finisarea virajului a ambelor unități, acestea încep să ruleze bucla principală. Prima unitate care are conectați ambii senzori preia și controlul unității a doua cu ajutorul distanței măsurate de către senzorul ultrasonic. Valoarea citită de la senzor este trimisă de către NXT1 prin Bluetooth și recepționată de către NXT2, care aceasta la rândul său este folosită pentru controlul motoarelor. După ce mesajul a fost recepționat, el este afișat pe ecranul unității de control. Codul ce este fișat în continuare face acest lucru.

```
//---recepționarea mesajului de la NXT1-----  
ubyte mesaj_Primit1[1];                                //  
cCmdMessageRead(mesaj_Primit1, 1, mailbox1);           //  
nxtDisplayTextLine(1, "mesaj_Primit: %i", mesaj_Primit1); //  
wait1Msec(100);                                         //  
//-----
```

Pentru o vizibilitate mai bună a mesajului de pe ecranul unității de control este pus un timp de întârziere a recepționării următorului mesaj. În prima linie de cod este menționat câți octeți are mesajul recepționat. Cea de-a doua linie are o funcție cCmdMessageRead, care citește mesajul primit (mesaj\_Primit1) din prima căsuță (mailbox1).

### 3.4.10.Controlul motoarelor în funcție de mesajul recepționat

În funcție de mesajul primit vom controla motoarele în continuare. Acest control ne va ajuta în special la întâlnirea obstacolelor, deoarece o dată ce am depistat un obstacol va fi nevoie ca blocul al doilea unde avem a doua unitate de control să meargă un pic mai încet față de cum mergea mai înainte, pentru a nu grăbi primul bloc unde avem prima unitate de control când urcă obstacolul în sine.

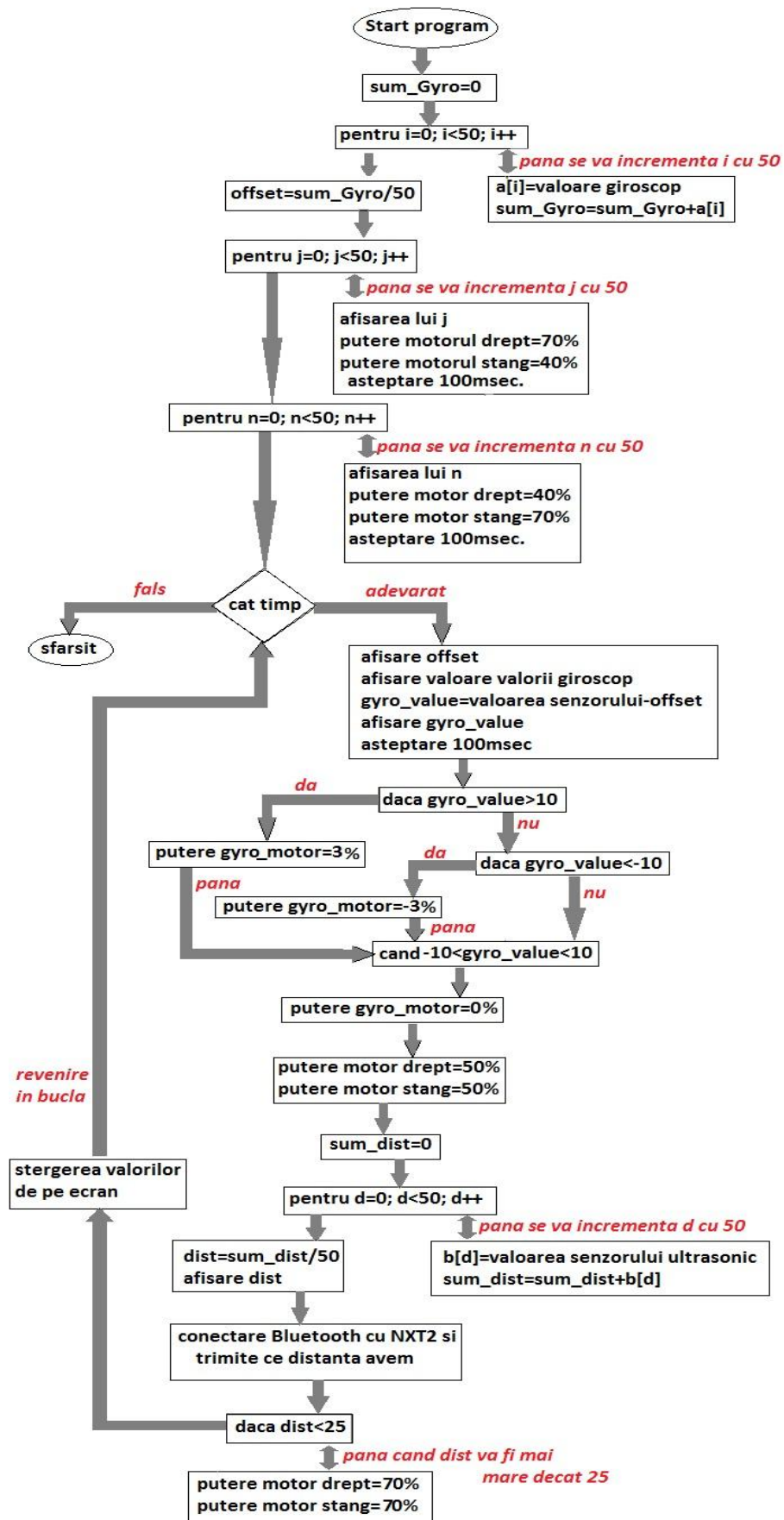
```

//---controlul motoarelor in functie de mesajul primit---
if (mesaj_Primit1 < 15 && mesaj_Primit1 > 5)      //
{                                                  //
    motor[Right_Motor]=-55;                      //
    motor[Left_Motor]=-55;                      //
    wait1Msec(100);                             //
    eraseDisplay();                             //
}                                                  //
else                                              //
{                                                  //
    motor[Right_Motor]=-50;                      //
    motor[Left_Motor]=-50;                      //
}                                                  //
//-----

```

### 3.5.Pseudocodul programului

Primul pseudocod este pentru programul principal al acestu robot care este afișat mai jos. Principiul de funcționare al acestui pseudocod începe cu startul programului. După ce s-a dat start acestui program, s-a inițializat suma valorilor giroscopului cu zero pentru a putea calibra senzorul giroscop.



Calibrarea se face printr-o buclă care citește senzorul giroscop de 50 de ori, ca să facem mai departe o medie a acestor valori citite. După ce s-au citit senzorul de 50 de ori și s-a făcut suma acestora se iese din buclă, unde se calculează offset-ul, împărțind acea suma a valorilor la numărul de citiri.

Aflând acest offset vom trece mai departe în altă buclă. Această buclă este un viraj la stânga și execută această comandă atîta timp cît variabila `j` este incrementată și ajunge la numărul 50. Bineînțeles că această valoare poate fi modificată, dacă vom pune o valoare mai mare vom sta în această buclă mai mult timp, adică va dura mai mult comanda de viraj, însă dacă vom pune o valoare mai mică vom avea un timp mai scurt pentru viraj.

Când s-a terminat comanda pentru viraj la stânga, intrăm în altă buclă care de data aceasta vom avea comandă de viraj la dreapta.

Bucula principală a programului începe abia acum care poate dura la nesfârșit teoretic, dar practic nu. La începutul acestei bucle avem comanda de afișare a offset-ului, a valorii citite de la senzorul giroscop pe ecranul unității de procesare. Se va calcula valoarea reală a giroscopului (`gyro_value`), prin scăderea offset-ului din valoarea citită de la senzor și se va afișa pe ecran.

Dacă valoarea calculată a giroscopului va fi mai mică decât 10, atunci motorul giroscopului va porni cu o putere de 3% și va merge până când valoarea va scădea sub 10. Însă dacă valoarea giroscopului este mai mică decât 10, atunci se va verifica următoarea condiție dacă valoarea giroscopului este mai mică decât -10, atunci motorul iar va porni dar de data aceasta cu o putere de -3% până când valoarea va fi mai mare decât -10.

Altfel această valoare nu corespunde nici uneia din cele două condiții și va sări la condiția a treia care este un interval. Acest interval motorul giroscopului va fi nemișcat, deoarece giroscopul se află în poziția cerută.

În timp ce giroscopul își verifică starea în care se află, motoarele blocului principal vor merge înainte cu o putere de 50%.

Inițializând mai întâi suma valorilor distanței citite de la senzorul ultrasonic cu zero înainte de a intra iarași într-o buclă, unde vom citi senzorul de 50 de ori. Valorile citite de senzor se vor

aduna. După ce variabila  $d$  s-a incrementat de 50 de ori se va face o medie a acestor distanțe și se va afișa pe ecran. Am citit senzorul de 50 de ori pentru a fi siguri ca avem distanța corectă.

La conectarea prin Bluetooth a celor două unități, aceasta distanță va fi trimisă celui alt NXT adică 2. În cazul în care distanța măsurată este mai mică decât 25, atunci puterea acestor motoare va crește pentru a putea urca obstacolul. La finalul acestei bucle principale se va șterge toate valorile calculate și afișate în timpul aceste bucle și se va relua bucla de la început.

Cel de-al doilea pseudocod care este afișat mai jos are altă funcționare și care începe la fel cu startul acestui program. dacă primul bloc face un viraj la stânga atunci este necesar ca și cel de-al doilea bloc să facă la fel, doar că vom avea de data aceasta puteri negative deoarece blocul al doilea este inversat față de primul.

Deci pentru a face viraj la stânga, motorul drept va avea o putere de mers de 20% și cel stâng o putere de 50%. Aceasta va dura atâta timp cât variabila  $i$  se va incrementa până la 50, adică va număra până la 50.

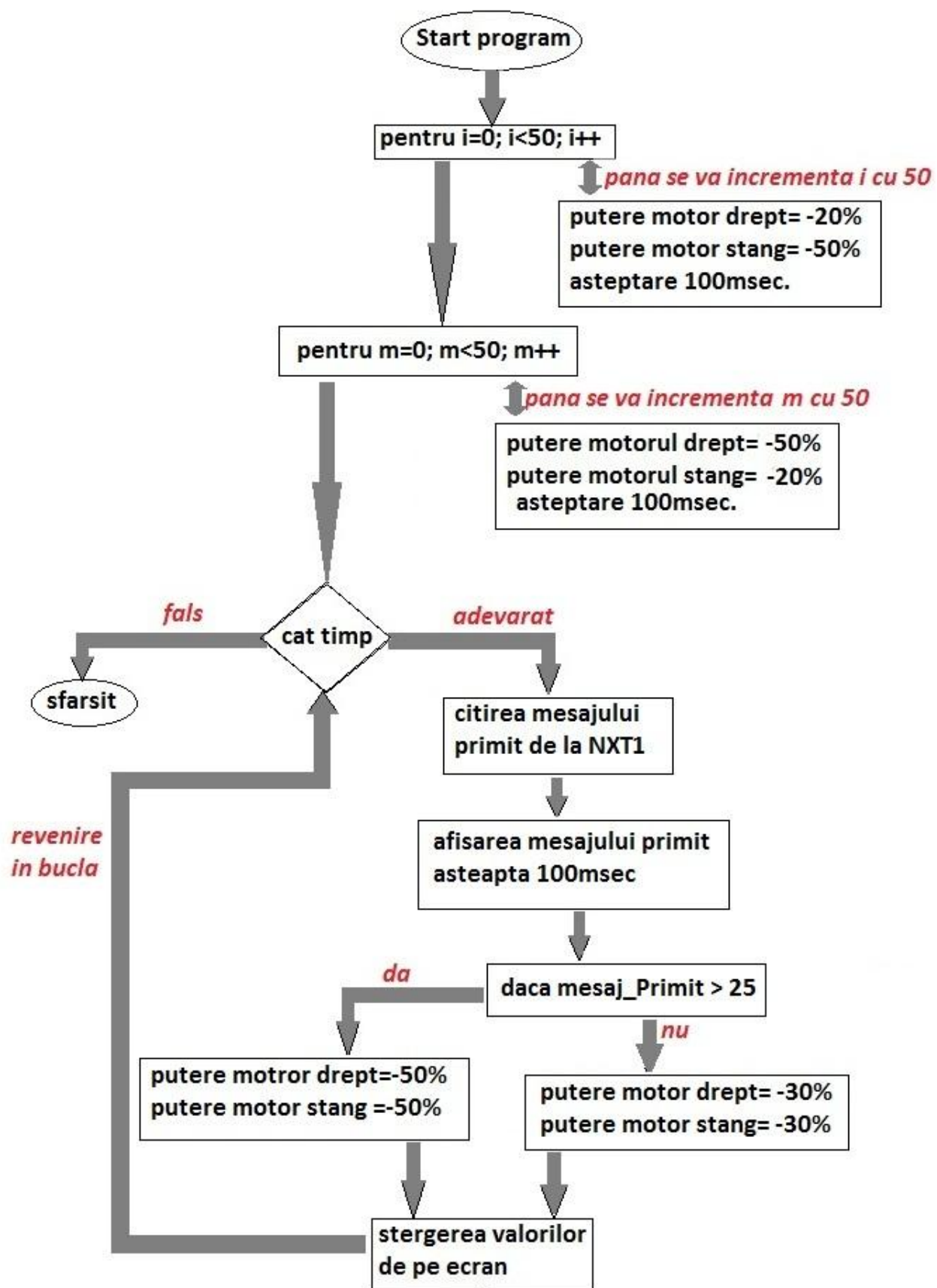
După finisarea acestei bucle, se va începe altă bucla care de data aceasta va face un viraj la dreapta cu o putere a motorului drept de -50% și a celui stâng de -20%.

De aici începe bucla principală a celui de-al doilea pseudocod, care la fel durează până la infinit. Primul lucru care se face în buclă este citirea mesajului ce a fost transmis de către NXT1. Mesajul ce este recepționat este o valoare ce reprezintă distanța față de obstacol.

Vizualizând acest mesaj pe ecranul unității, putem să-l urmărim în timp ce robotul se misca.

Dacă valoarea mesajului primit este mai mare decât 25, atunci ambele motoare ale acestui bloc vor merge cu aceeași putere de -50%, în cazul în care valoarea mesajului va fi mai mică decât 25, atunci ambele motoare vor merge la fel cu aceeași putere doar că mai mică, adică cu o putere de -30%.

La finalul acestei bucle toate valorile vor fi șterse de pe ecranul unității de control și se va relua citirea aceste bucle din nou.





## Concluzie

În urma realizării acestei lucrări am observat că concepțiile teoretice diferă de cele practice. Am dorit să fac robotul să treacă peste obstacole, măsurând distanța până la acel obstacol cu ajutorul senzorului ultrasonic și am obținut aceasta.

Am observat că senzorul giroscop pe care am dorit să-l țin într-o anumită nu este atât de ușor de controlat fiindcă este foarte sensibil chiar și la cele mai mici mișcări ale lui. Am încercat să-l readuc aproximativ la poziția necesară care mi-am propus-o și am realizat aceasta.

Am realizat o comunicare Bluetooth între cele două unități de control, caci doar astfel ele se pot sincroniza în mișcările care trebuie să le facă. Toate acestea le-am realizat cu Platforma Lego Mindstorms.

# Bibliografie

<http://www.philohome.com/motors/motorcomp.htm>

<http://www.robotc.net/>

<http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com>

<http://www.generationrobots.com/en/content/65-ultrasonic-sonar-sensors-for-robots>

<http://www.vexrobotics.com/wiki/ROBOTC>

<http://books.ifmo.ru/file/pdf/1081.pdf>

<http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044>

Michael\_Gasperi,\_Philippe\_Hurbain\_Extreme\_NXT\_Extending\_the\_LEGO\_MINDSTORMS\_NXT  
\_to\_the\_Next\_Level,\_Second\_Edition\_Technology\_in\_Action\_\_2009

Mario\_Ferrari,\_Guilio\_Ferrari\_Building\_Robots\_with\_LEGO\_Mindstorms\_NXT\_\_2007

Karsten\_Berns,\_Daniel\_Schmidt\_Programmierung\_mit\_LEGO\_Mindstorms\_NXT\_Robotersysteme  
,\_Entwurfsmethodik,\_Algorithmen\_eXamen.press\_\_\_\_0

Jr. Jerry Lee Ford Lego Mindstorms NXT 2.0 for Teens 2011

John\_Baichtal,\_Matthew\_Beckler,\_Adam\_Wolf\_Make\_LEGO\_and\_Arduino\_Projects\_Projects\_for  
\_extending\_MINDSTORMS\_NXT\_with\_open-source\_electronics\_\_2012

James\_Jeffrey\_Trobaugh\_Winning\_Design!\_LEGO\_MINDSTORMS\_NXT\_Design\_Patterns\_for\_F  
un\_and\_Competition\_\_2010

Mario Ferrari, Guilio Ferrari Building Robots with LEGO Mindstorms NXT 2007

Terry\_Griffin\_The\_Art\_of\_LEGO\_MINDSTORMS\_NXT\_G\_Programming\_\_2010

## Anexă

**Codul principal pentru master, adică NXT1:**

```
int Gyro_value;
int i;
int a[50];
int sum_Gyro=0;
int dist;
int d;
int j;
int n;
int b[50];
int offset;

//-----

//---programul principal pentru NXT1-----

task main()
{
    //--calibrarea giroscopului-----
    for (i=0; i<50; i++)
    {
        a[i]= SensorValue[Gyro_Sensor];
        sum_Gyro = sum_Gyro + a[i];
    }
    offset = sum_Gyro / 50;

    //-----
```

```

//---comanda de viraj la stanga-----
for(j=0; j<50; j++)
{
    nxtDisplayBigTextLine(1, "j=%i", j);
    motor[Right_Motor]=70;
    motor[Left_Motor]=40;
    wait1Msec(100);
}
eraseDisplay();

//-----

//---comanda de viraj la dreapta----
for(n=0; n<50; n++)
{
    nxtDisplayTextLine(2, "n=%i", n);
    motor[Right_Motor]=40;
    motor[Left_Motor]=70;
    wait1Msec(100);
}
eraseDisplay();

//-----

while (true)
{
    //---afisarea valorilor senzorului giroscop-----

    nxtDisplayTextLine(1, "offset: %i", offset);
    nxtDisplayTextLine(2, "SensorValue: %i", SensorValue[Gyro_Sensor]);
}

```

```

Gyro_value=SensorValue(Gyro_Sensor)-offset;
nxtDisplayTextLine(3, "Gyro_value: %i",Gyro_value);
grade=grade+Gyro_value*.001;
nxtDisplayTextLine(4, "grade: %.3f",grade);
wait1Msec(100);

//-----

//---controlul giroscopului-----
if (Gyro_value > 10)
    motor[ Gyro_Motor]=3;
    wait1Msec(50);
if (Gyro_value <-10)
    motor[ Gyro_Motor]=-3;
    wait1Msec(50);
if (Gyro_value<10 && Gyro_value>-10)
    motor[ Gyro_Motor]=0;
    wait1Msec(50);

//-----

//---pornirea motoarelor la NXT1----
    motor[Right_Motor]=50;
    motor[Left_Motor]=50;

//-----

//---distanța fata de obiect și afișarea----
int sum_dist=0;
for(d=0; d<50; d++)
{

```

```

        b[d]=SensorValue(Sonar_Sensor);
        sum_dist=sum_dist+b[d];
    }
    dist=sum_dist/50;
    nxtDisplayTextLine(5, "dist= %i", dist);
    wait1Msec(300);

    //-----

    //---comunicarea valorii distantei celui alt NXT-----
    ubyte mesaj_Trimis1[1]={ dist };
    btConnect(1, "NXT2");
    cCmdMessageWriteToBluetooth(mesaj_Trimis1, 1, mailbox1);
    if (dist < 25 )
    {
        motor[Right_Motor]=70;
        motor[Left_Motor]=70;
    }

    //-----

    //---stergereea valorilor--
    eraseDisplay();

    // -----
}

}

//-----

```

### Codul pentru cel de-al doilea NXT va fi:

```
#pragma config(Motor, motorB,      Right_Motor,  tmotorNormal, PIDControl, encoder)
#pragma config(Motor, motorC,      Left_Motor,   tmotorNormal, PIDControl, encoder)
/*!!Code automatically generated by 'ROBOTC' configuration wizard      !!*/

int i;
int m;

//---programul principal pentru NXT2-----
task main()
{

    //---comanda de viraj la stanga----
    for(i=0; i<50; i++)
    {
        motor[Right_Motor]=-20;
        motor[Left_Motor]=-50;
        wait1Msec(100);
    }
    //-----

    //---comanda de viraj la dreapta----
    for(m=0; m<50; m++)
    {
        motor[Right_Motor]=-50;
        motor[Left_Motor]=-20;
        wait1Msec(100);
    }
}
```

```

}

//-----

while (true)
{
    //---receptionarea mesajului de la NXT1-----
    ubyte mesaj_Primit1[1];
    cCmdMessageRead(mesaj_Primit1, 1, mailbox1);
    nxtDisplayTextLine(2, "mesaj_Primit1: %i", mesaj_Primit1);
    wait1Msec(100);
    //-----

    //---controlul motoarelor in functie de mesajul primit---
    if (mesaj_Primit1 > 25 )
    {
        motor[Right_Motor]=-50;
        motor[Left_Motor]=-50;
        wait1Msec(100);
    }
    else
    {
        motor[Right_Motor]=-30;
        motor[Left_Motor]=-30;
    }
    //-----

    //---stergerea ecranului---
    eraseDisplay();

```



```
//-----  
}  
}  
//-----
```

[http://www.robotics.ucv.ro/flexform/aplicatii\\_ser2/Mecatronica%20I/mirela%20berceanu-  
nu-  
APLICATII%20%20UTILIZAND%20SENZORII%20%20LEGO%20MINDSTORMS/Lucrare  
\\_Finala\\_mirela\\_berceanu\\_modul2.htm](http://www.robotics.ucv.ro/flexform/aplicatii_ser2/Mecatronica%20I/mirela%20berceanu-<br/>nu-<br/>APLICATII%20%20UTILIZAND%20SENZORII%20%20LEGO%20MINDSTORMS/Lucrare<br/>_Finala_mirela_berceanu_modul2.htm)

### Partea tehnică:







