Driving a RC Car with Java leJOS

Versión 0.3

Juan Antonio Breña Moral

27-ago-08

Index

Revision History

Name	Date	Reason For Changes	Version
Juan Antonio Breña Moral	21/06/2008	Initial version	0.1
Juan Antonio Breña Moral	08/07/2008	I have added new sections	0.2

1.- Introduction

1.1.- Goals

Many developers around the world choose leJOS, Java for Lego Mindstorm, as the main platform to develop robots with NXT Lego Mindstorm. I consider that this eBook will help leJOS community, Lego Mindstorm community, Robot's developers and Java fans to develop better software.

Robotics will be very important for the humanity in the next 10 years and this eBook is an effort to help in this way.

Many people spend several hours in their robotics projects with problems with wires & electronics, protocols and problems with programming languages, Lego Mindstorm is easy and Java/leJOS is an excellent platform to demonstrate your software engineering skills to develop better robots. NXT Brick is the easiest way to enter in the robotics world and leJOS, the best platform in the moment to use software engineering ideas.

Enjoy, Learn, Contact with me to improve the eBook and share your ideas.
Download latest eBook release here: http://juanantonio.info/jab_cms.php?id=206

Juan Antonio Breña Moral.
www.juanantonio.info

1.1.1.- About this document

This document has been written to explain how to control a RC Car with leJOS and the new Device for leJOS, NXTe Kit. In this article I will explain how to use a traditional RC Car with NXT using as glue NXTe.

The solution described in this document use the following leJOS features:

1. Multithreading
2. Bluetooth API with topology Master-Slave
3. Lattebox NXTe API

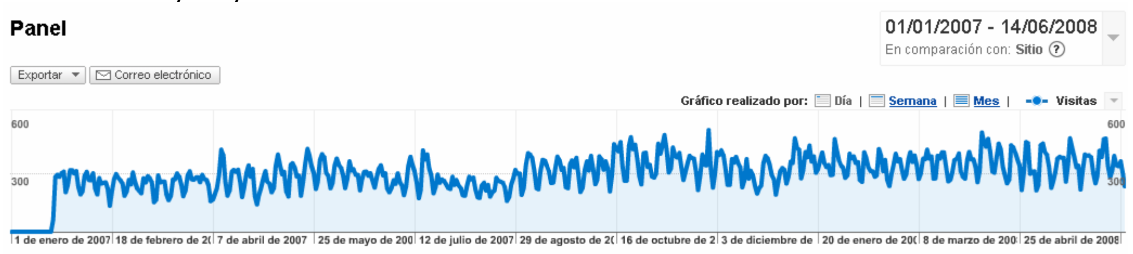
1.2.- LeJOS Project

LeJOS is Sourceforge project created to develop a technological infrastructure to develop software into Lego Mindstorm Products using Java technology.

Currently leJOS has opened the following research lines:

1. NXT Technology
    a. NXJ
    b. LeJOS PC API
    c. iCommand
2. RCX Technology
    a. leJOS for RCX

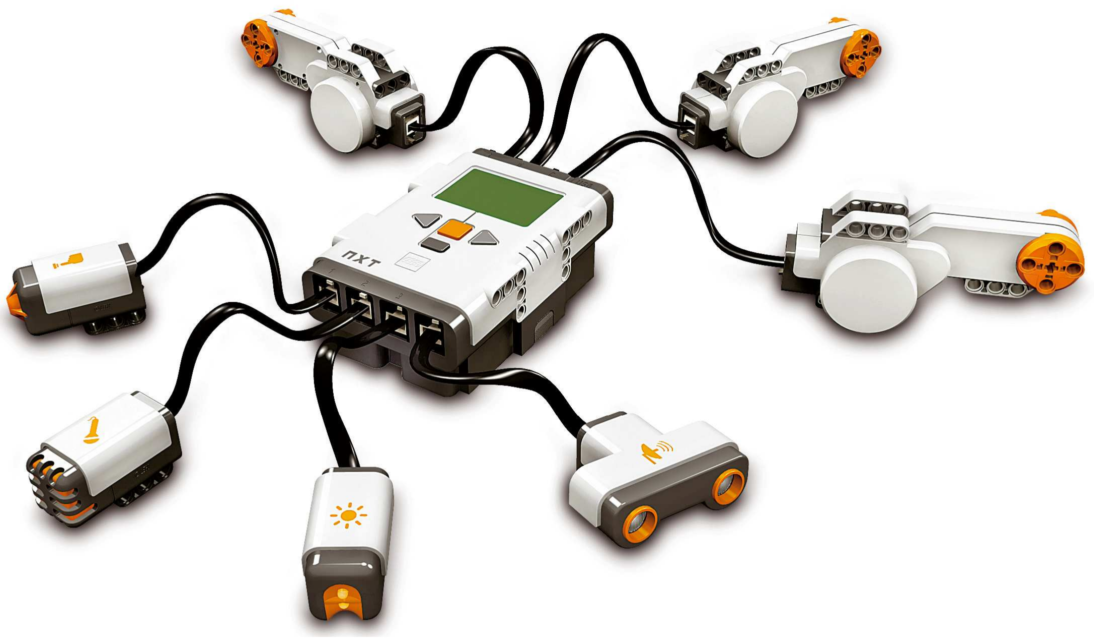
LeJOS project's audience has increased. Currently more than 500 people visit the website every day.



This eBook will focus in NXT technology with NXJ using a Windows Environment to develop software.

1.3.- NXT Brick

The NXT is the brain of a MINDSTORMS robot. It's an intelligent, computer-controlled LEGO brick that lets a MINDSTORMS robot come alive and perform different operations.



Motor ports
The NXT has three output ports for attaching motors - Ports A, B and C

Sensor ports
The NXT has four input ports for attaching sensors - Ports 1, 2, 3 and 4.

USB port
Connect a USB cable to the USB port and download programs from your computer to the NXT (or upload data from the robot to your computer). You can also use the wireless Bluetooth connection for uploading and downloading.

Loudspeaker
Make a program with real sounds and listen to them when you run the program

NXT Buttons

Orange button: On/Enter /Run
Light grey arrows: Used for moving left and right in the NXT menu
Dark grey button: Clear/Go back

NXT Display
Your NXT comes with many display features - see the MINDSTORMS NXT Users Guide that comes with your NXT kit for specific information on display icons and options

Technical specifications

- 32-bit ARM7 microcontroller
- 256 Kbytes FLASH, 64 Kbytes RAM
- 8-bit AVR microcontroller
- 4 Kbytes FLASH, 512 Byte RAM
- Bluetooth wireless communication (Bluetooth Class II V2.0 compliant)
- USB full speed port
- 4 input ports, 6-wire cable digital platform (One port includes a IEC 61158 Type 4/EN 50 170 compliant expansion port for future use)
- 3 output ports, 6-wire cable digital platform
- 100 x 64 pixel LCD graphical display
- Loudspeaker - 8 kHz sound quality. Sound channel with 8-bit resolution and 2-16 KHz sample rate.
- Power source: 6 AA batteries

1.3.1.- NXT Sensors used in the eBook

NXT Sensors used in the document are the following:

- NXT Motor
- Ultrasonic Sensor
- Compass Sensor
- NXTCam
- Tilt Sensor
- NXTCam
- NXTe

NXT Motor

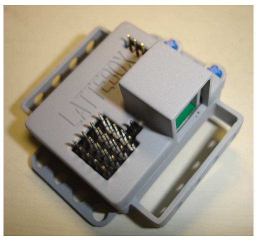
Ultrasonic Sensor

Compass Sensor

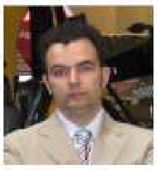
Tilt Sensor

NXTCam

NXTe



1.4.- About the author



Juan Antonio Breña Moral collaborates in leJOS Research team since 2006. He works in Europe leading Marketing, Engineering and IT projects for middle and large customers in several markets as Defence, Telecommunications, Pharmaceutics, Energy, Automobile, Construction, Insurance and Internet.

Further information:
www.juanantonio.info
www.esmeta.es

2.- Project Status

2.1.- Report

Last update: 2008/08/26

Currently I have developed the following products:

Product	Version	Notes
RC Controller	20080709	**RC Controller v1.1** RC Controller is a solution to manage TLB using BT. Currently, RC Controller sends data but it doesn't receive any data by TLB. GUI is friendly and data received by NXT motor is processed with TLB Protocol.
TLB Protocol	20080825	**TLB Protocol v1.2** TLB Protocol is stable protocol designed to manage Tamiya Lunch Box internal parameters, BT communications and Communication protocol.
TLB	20080826	**TLB v3** TLB runs stable but I have to increase the security to avoid crashing scenes. BT Listeners runs well. It incorporates features to reset internal values when BT communications fails. TLB uses a Multithreading solution. I have to research to incorporate backward functionality. Besides I will incorporate defensive measures in case of low Bluetooth signal. **Tamiya Lunch Box v1.2** Tamiya Lunch Box is the class which model the physical layer. **Racing TLB v1.0** Racing Car is a class which inherits from Tamiya Lunch Box. This class incorporates new features as US Grid Control and This class the method boost()
Calibration tools	20080825	**CalibrateServo v1.0** I have released new utilities to calibrate steering. The utility connects with system to observe the right angle to establish the physical component which manages the steering. **CalibrateDCMotor v1.0** I have released new utilities to calibrate DC Motor. Now it is possible to manage Backward method **RCDiagnostics v3** RCDiagnostics is a utility which connect with TLB to test the mechanics. This tool is local and it uses TLB NXT Brick.
UML Drawings	20080809	**NXTe UML Drawings v1.1** NXTe UML Drawings incorporate latest features: setPulse and getPulse

Conclusions:
The project has reached the status of is stable project. TLB listen a Bluetooth connection from RC Controller. When RC Controller and TLB establish a BT connection then RC Controller send commands using TLB Protocol. RC Controller encodes data. TLB has a BT Subsystem which manages BT communications. Data from this system is processed with TLB Protocol. Once TLB has the CMD then TLB executes them.

Currently main products: RC Controller and TLB use a Multithreading solution. I will research in TLB a solution using Behaviour with Threads.

2.2.- Roadmap

The project roadmap is above:

1. Improve TLB security
2. Add GPS Support
3. Add GPS Navigation subsystem
4. Add GPS logging
5. Redesign TLB system with Subsumption architecture
6. Add bidirectional communications
7. Add Event Listeners in GUI Subsystem

3.- The solution

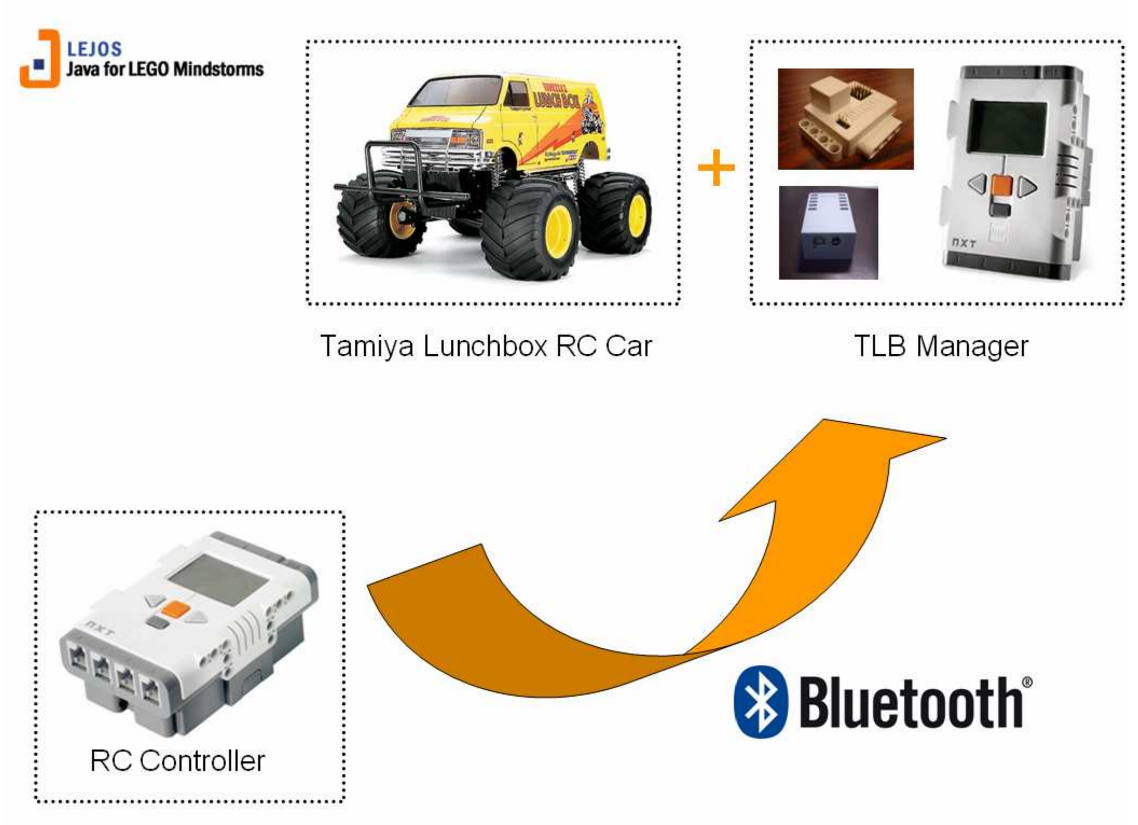
3.1.- Introduction

Traditionally, NXT Technology is able to manage NXT Servo, PF Motor and legacy RCX Motors. With the new product designed by the company Lattebox, now it is possible to manage RC Servos and DC Motors.

This paper will describe how to control a RC Car, in this example a Tamiya Lunch Box RC Car.

3.2.- Architecture

The component architecture about this project is the following:



The RC car which I have chosen for this project is the classical RC Model, Tamiya Lunchbox RC Car. This RC Car can be managed by the Lattebox solution NXTe Kit. NXTe kit is managed by a TLB Manager. TLB Manager is software developed with leJOS, Java for LEGO Mindstorm.

3.3.- Software components

Software components used in this solution are listed above:

1. RC Controller
2. TLB Manager

3. TLB Communication Protocol
4. System utilities
    a. RC Diagnostics
    b. Calibrate Servo

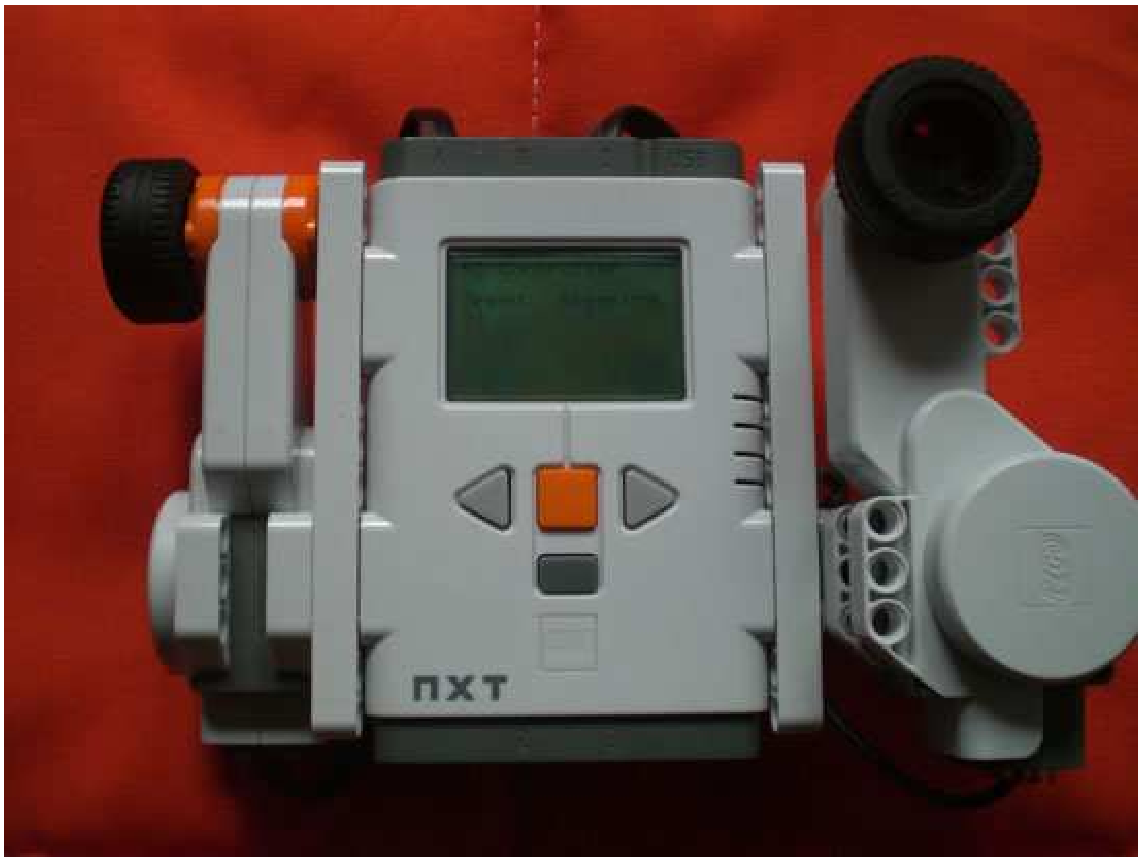
3.3.1.- RC Controller

RC Controller is a leJOS solution developed to communicate with LTB Manager using Bluetooth. This component use TLB Communication Protocol.

Physically, RC Controller is a NXT brick which is connected with 2 NXT Servos. In this case NXT Servos are used to control 3 Parameters:

1. Locomotion Power
2. Direction
3. Steering

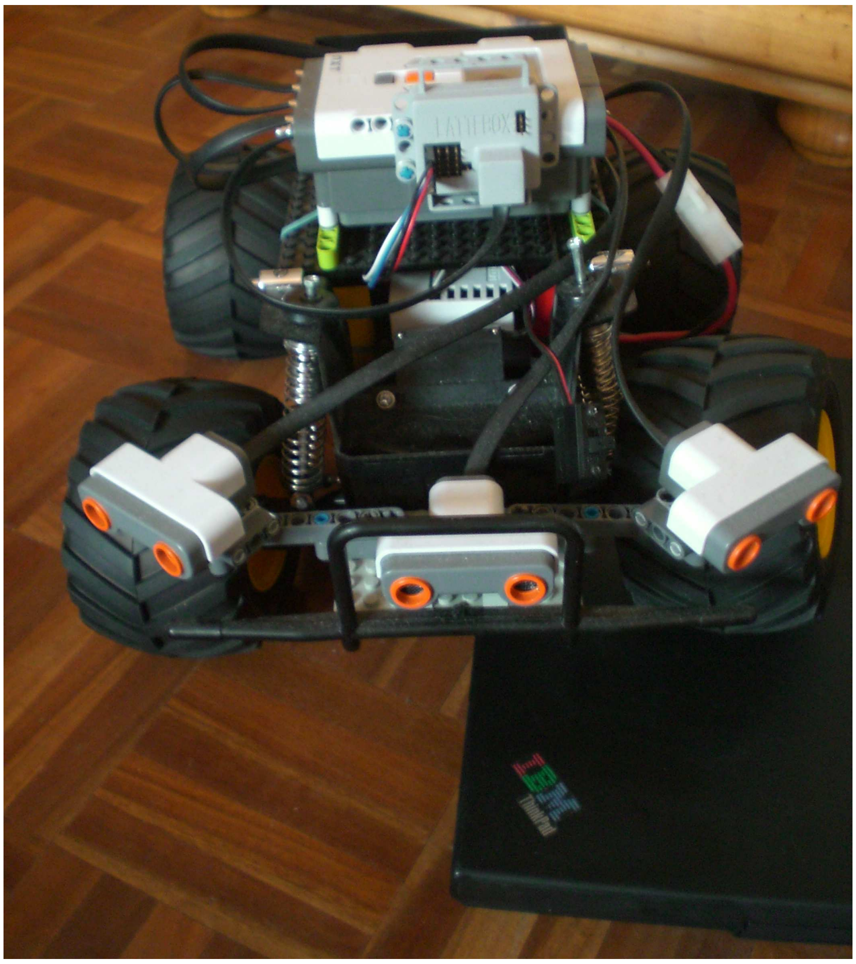
GUI is very simple, It shows the Speed with + or + if Car go forward or backward and the degree of steering and showing > if RC car turn right and < if RC turn left.



3.3.2.- TLB Manager

TLB Manager is leJOS software designed to achieve the following goals:

1. Manage Bluetooth communications
2. Manage TLB



In a future I will incorporate autonomous behaviour to avoid obstacles. Besides I have the idea to incorporate a GPS unit to generate a GPS Navigator.

3.3.2.1.- Problems detected

I have detected the following problems:

1. Bidirectional communications
2. leJOS events

When TLB sends data to RC Controller system crash. This failure is not critic but I would like to solve.

I have tried to use Java Events to manage better the GUI but currently I have problems to send an Object to a ButtonListener Object which implements the interface ButtonListener.

Stable code:

```
            while(!Button.ESCAPE.isPressed()){
                  try{

CODE

                  }catch(Exception e){
                        //Empty
                  }
            }
```

The code which I would like to implement:

```
MyCarButtonListener MCBL= new MyCarButtonListener(TLB);
Button.ESCAPE.addButtonListener(new MyButtonListener());
```

And the class MyCarButtonListener:

```
import lejos.nxt.*;

public class MyCarButtonListener implements ButtonListener{

      private TamiyaLunchBox TLB;

      public MyCarButtonListener(TamiyaLunchBox _TLB){
            TLB = _TLB;
      }

      public void buttonPressed(Button b) {
            LCD.clear();
      }

      public void buttonReleased(Button b) {
            this.TLB.turnOff();
            LCD.drawString("Program finished",0,7);
            System.exit(0);
      }
}
```

3.3.3.- TLB Communication Protocol

Communications between RC Controller and TLB Manager have to be defined in a formal protocol. It is necessary to establish a rule which every message send or receive have to finish with a default value.

If the communication uses integers to send and receive commands the protocol have this form:

ID	CODE	VALUE	DESCRIPTION
1	1010	SPEED 10	SET/GET SPEED 10 FORWARD
2	1020	SPEED 20	SET/GET SPEED 20 FORWARD
3	1030	SPEED 30	SET/GET SPEED 30 FORWARD
4	1040	SPEED 40	SET/GET SPEED 40 FORWARD
5	1050	SPEED 50	SET/GET SPEED 50 FORWARD
6	1060	SPEED 60	SET/GET SPEED 60 FORWARD
7	1070	SPEED 70	SET/GET SPEED 70 FORWARD
8	1080	SPEED 80	SET/GET SPEED 80 FORWARD
9	1090	SPEED 90	SET/GET SPEED 90 FORWARD
10	1100	SPEED 100	SET/GET SPEED 100 FORWARD
11	2010	SPEED -10	SET/GET SPEED 10 BACKWARD
12	2020	SPEED -20	SET/GET SPEED 20 BACKWARD
13	2030	SPEED -30	SET/GET SPEED 30 BACKWARD
14	2040	SPEED -40	SET/GET SPEED 40 BACKWARD
15	2050	SPEED -50	SET/GET SPEED 50 BACKWARD
16	2060	SPEED -60	SET/GET SPEED 60 BACKWARD
17	2070	SPEED -70	SET/GET SPEED 70 BACKWARD

18	2080	SPEED -80	SET/GET SPEED 80 BACKWARD
19	2090	SPEED -90	SET/GET SPEED 90 BACKWARD
20	2100	SPEED -100	SET/GET SPEED 100 BACKWARD
21	3010	TURN LEFT 10	SET/GET TURN LEFT 10
22	3020	TURN LEFT 20	SET/GET TURN LEFT 20
23	3030	TURN LEFT 30	SET/GET TURN LEFT 30
24	3040	TURN LEFT 40	SET/GET TURN LEFT 40
25	3050	TURN LEFT 50	SET/GET TURN LEFT 50
26	3060	TURN LEFT 60	SET/GET TURN LEFT 60
27	3060	TURN LEFT 70	SET/GET TURN LEFT 70
28	3070	TURN LEFT 80	SET/GET TURN LEFT 80
29	3080	TURN LEFT 90	SET/GET TURN LEFT 90
30	3100	TURN LEFT 100	SET/GET TURN LEFT 100
31	4010	TURN RIGHT 10	SET/GET TURN RIGHT 10
32	4020	TURN RIGHT 20	SET/GET TURN RIGHT 20
33	4030	TURN RIGHT 30	SET/GET TURN RIGHT 30
34	4040	TURN RIGHT 40	SET/GET TURN RIGHT 40
35	4050	TURN RIGHT 50	SET/GET TURN RIGHT 50
36	4060	TURN RIGHT 60	SET/GET TURN RIGHT 60
37	4070	TURN RIGHT 70	SET/GET TURN RIGHT 70
38	4080	TURN RIGHT 80	SET/GET TURN RIGHT 80
39	4090	TURN RIGHT 90	SET/GET TURN RIGHT 90
40	4100	TURN RIGHT 100	SET/GET TURN RIGHT 100
41	5001	GET SPEED	
42	5002	GET STEERING	
43	5003	GET BATTERY	

If the communications use strings the protocol is:

1. 3 characters to define the command: GSP | FOR | BAC | GST | TUL | TUR | BAT
2. 3 characters to define the value

Examples:

- Turn left 50 %: TUL050
- Turn right 80%: TUR080
- Forward 100%: FOR100
- Get battery: BAT000
- Get Speed: GSP000
- Get Steering: GST000

Note:
Current version uses Integers values to encode the protocol. RC Controller sends speed values to establish the power. RC Car only goes straight.

3.3.4.- System utilities

I have developed a set of utilities to make:

- Diagnostics
- Calibrate the steering

3.3.4.1.- RC Diagnostics

Every real machine needs to tune correctly to runs well. If you see any F1 Race you see several examples. Every F1 Racing Car has the same mechanic pieces but it is necessary to tune them to runs well. With a RC Car the concept is similar. In this case, there are 2 elements to tune:

- RC Servo
- DC Motor

The RC Servo handle the steering and it is necessary to know the limits about this one then you need to know the following answer about this questions:

1. What is the limit to turn left?
2. What is the limit to turn right
3. What is the neutral point

In case of DC Motor, you need to establish maximum level, minimum level and a security zone to run.

To know these parameters, it is necessary to develop tools to calibrate your RC Car. These values will be used when TamiyaLunchBox.java will be instanced because these parameters would be established.

3.3.4.2.- Utility to calibrate the RC Servo

When RC Car project begun to receive data from RC Controller I had to recalibrate the RC Servo which manages the steering. To make a right calibration was necessary to develop a utility to observe the steering to know middle angle, minimum angle and maximum angle

3.3.4.3.- Utility to calibrate the DC Motor

Currently, leJOS support for Lattebox NXTe/LSC allows to manages forward and backward methods. This utility allows to test the maximum and minimum speed when DC Motor goes forward and backward.

3.3.5.- UML Drawings

I am developing UML diagrams with the open source project: Argo UML:

The UML diagrams developed are:

- Lattebox NXTe Kit

4.- Technology

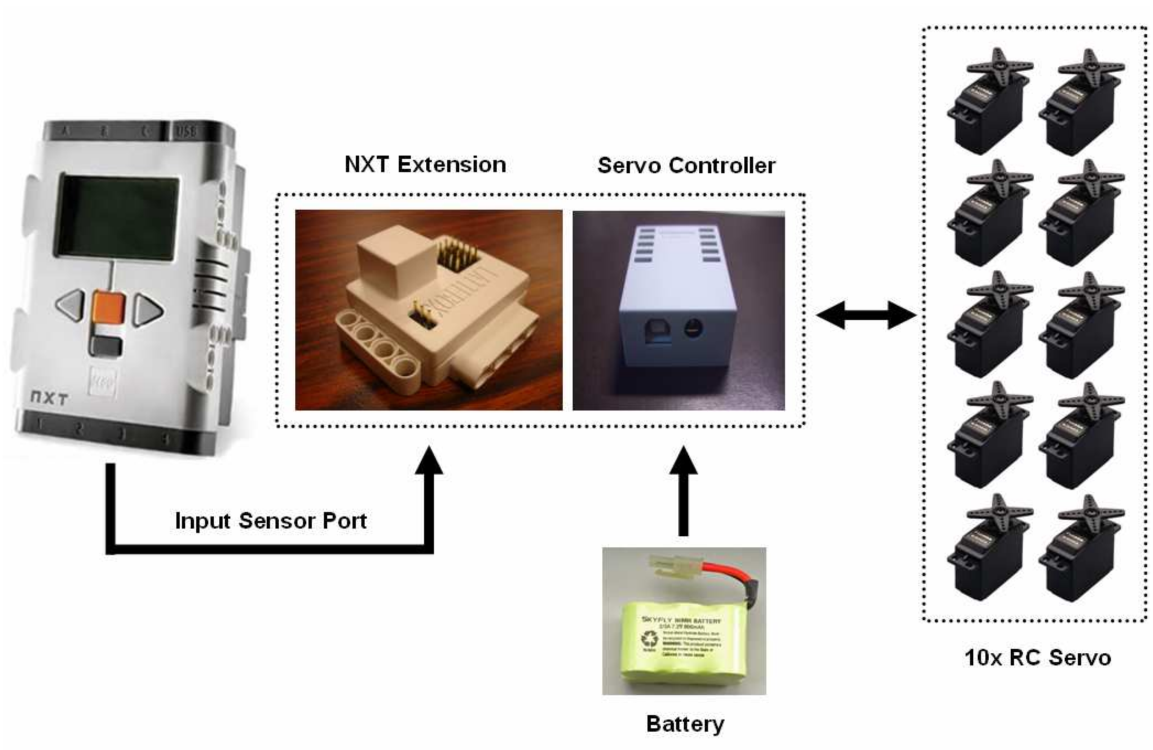
4.1.- Lattebox NXTe Kit

In 2008, Lattebox a hi-tech company located in Taiwan, released a new kind of NXT device, NXTe. NXTe allows controlling RC servos and DC Motors easily. NXT brick has 4 sensor ports to control NXT sensors as Ultrasonic Sensors, Compass Sensors, NXTCam Sensors, etc…

If you connect Lattebox NXTe in any free input sensor port, you could manage until 10 RC Servos with your NXT brick with an unique NXTe kit.

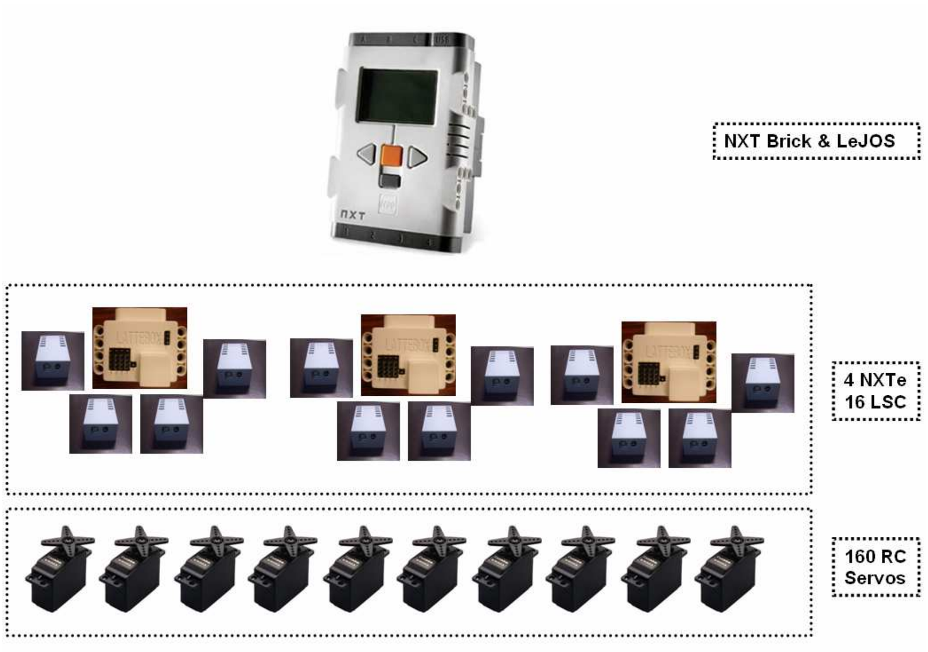
4.2.- Lattebox NXTe architecture

The NXTe architecture is the following:



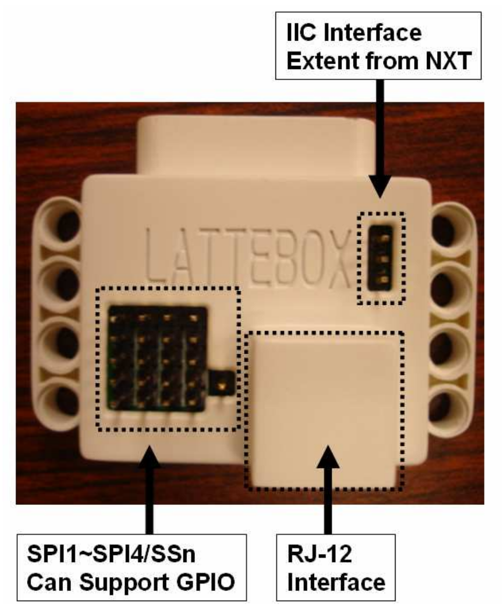
If you notice, with NXTe technology you can control:

4x NXTe * 4x LSC * 10x RC Servos = 160 Servos



4.2.1.- NXT Extension, NXTe

NXT Extension is new device developed by Lattebox to establish a bridge between the world of RC Servo and NXT Technology. NXTe uses the energy from NXT Brick. This device is able to manage until 4 Servo Controller.



4.2.2.- Servo Controller

Servo Controller Device, is connected with NXTe to manage until 10 RC Servos.

This device needs an external energy, 6.8V/4000mAh source to runs.

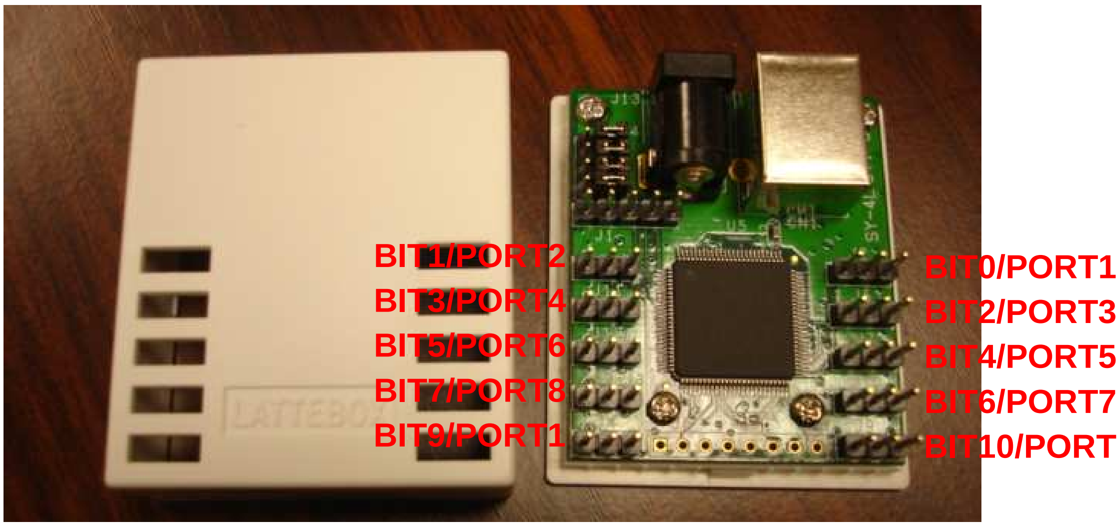
If you have 1 Servo Controller, connect this one in SPI 1 in NXTe:

When you connect any RC Servo or DC motor into LSC, you have to know pin details:

- Pin 1: Signal
- Pin 2: Positive
- Pin 3: Negative

4.2.2.1.- Positions in LSC

When you connect a RC Servo or a DC motor into your LSC, it is necessary to know the positions. With this graphics this task is easy.



4.3.- What kind of motors does NXTe Kit manages?

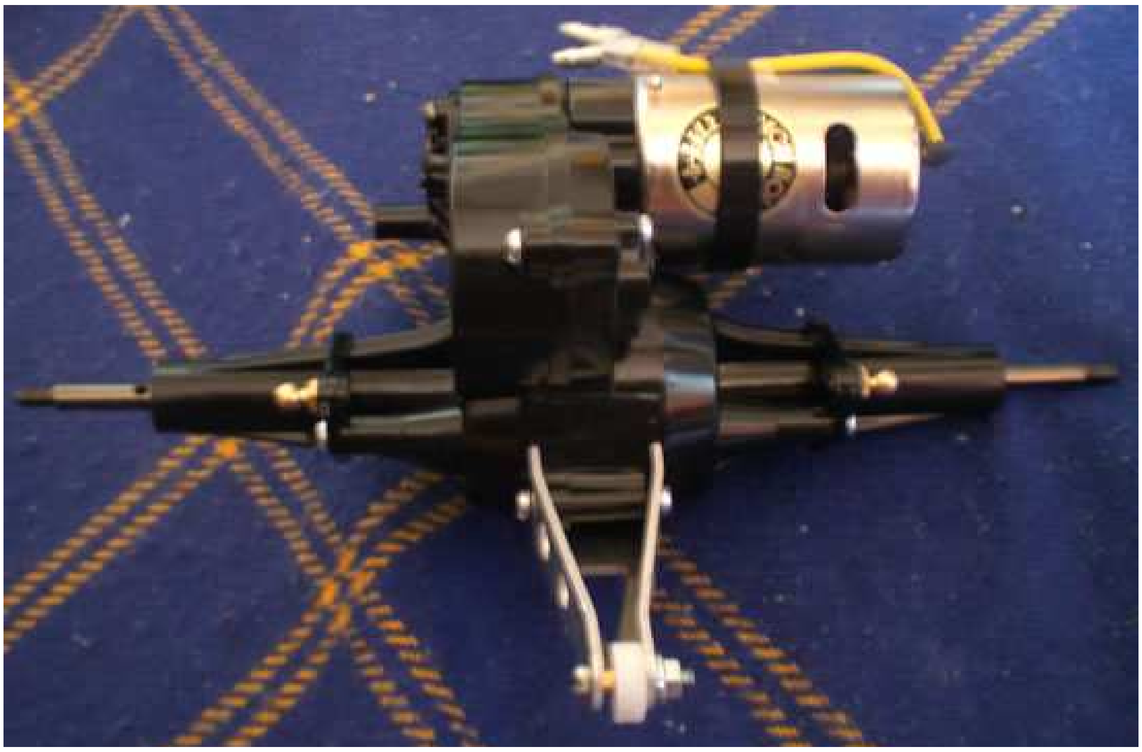
NXTe Kit manages RC Servos and DC motors. In this section I will explain with detail these motors.

4.3.1.- Servos

A Servo is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servo a coded signal. As long as the coded signal exists on the input line, the servo will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. Most servo motors can rotate about 90 to 180 degrees. Some rotate through a full 360 degrees.

4.3.2.- DC Servos

A DC Motor is a powerful motor for robotics purpose. It has so much power in relation to conventional NXT Servos.



4.3.3.- How does a servo work?

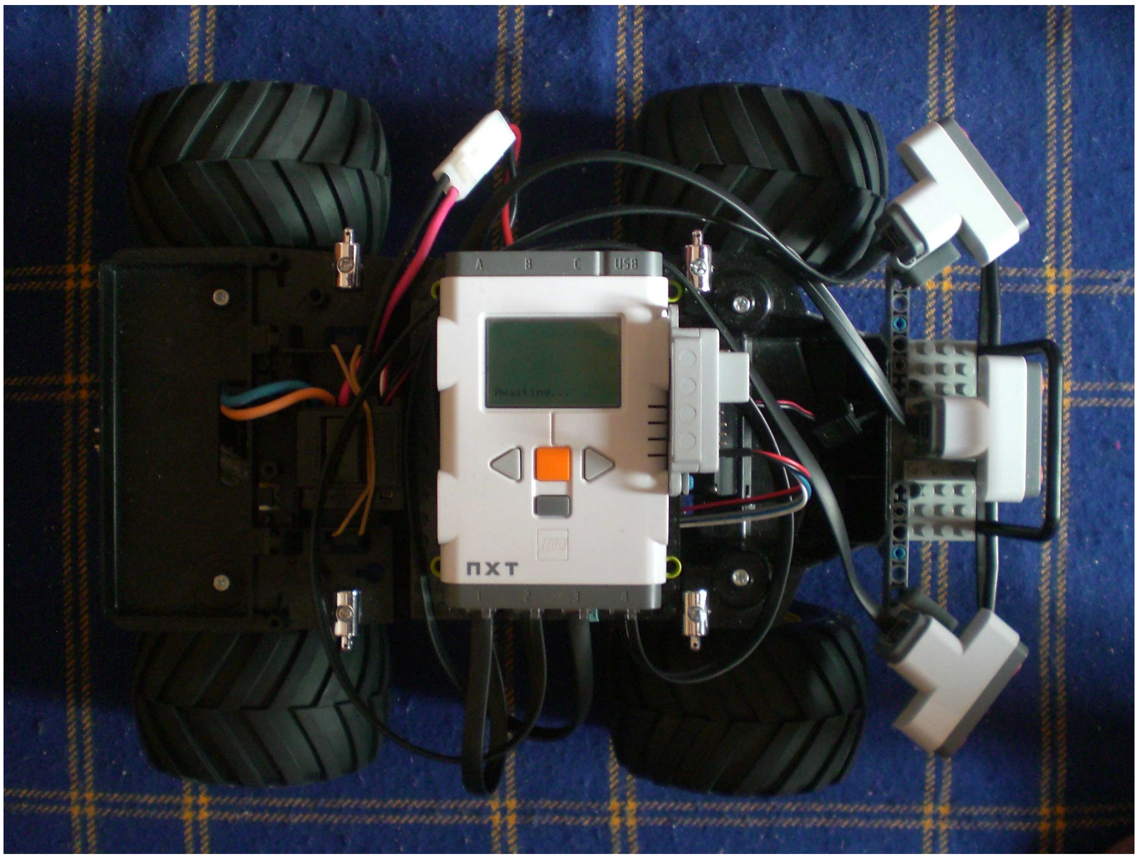
The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) that is connected to the output shaft. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of traveling somewhere around 180 degrees. Usually, its somewhere in the 210 degree range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear.

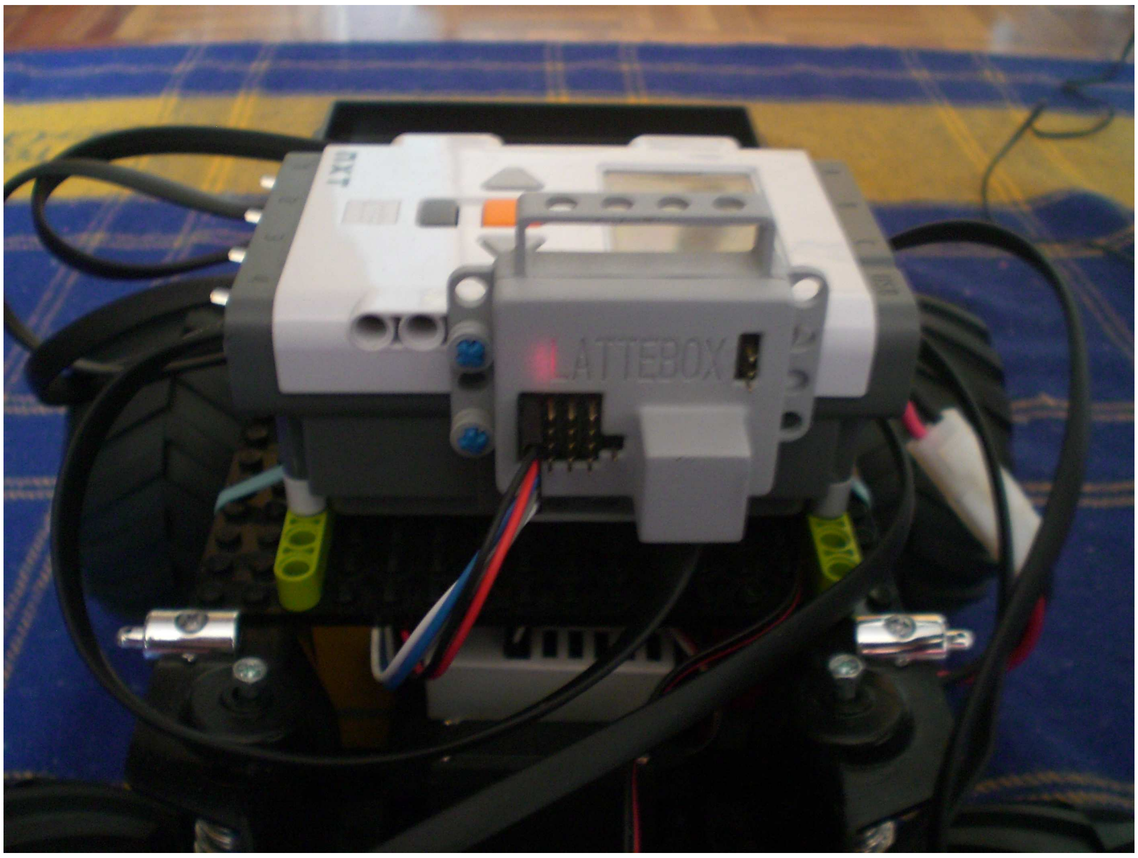
The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control.

4.4.- NXTe applications

4.4.1.- Manage a RC Car

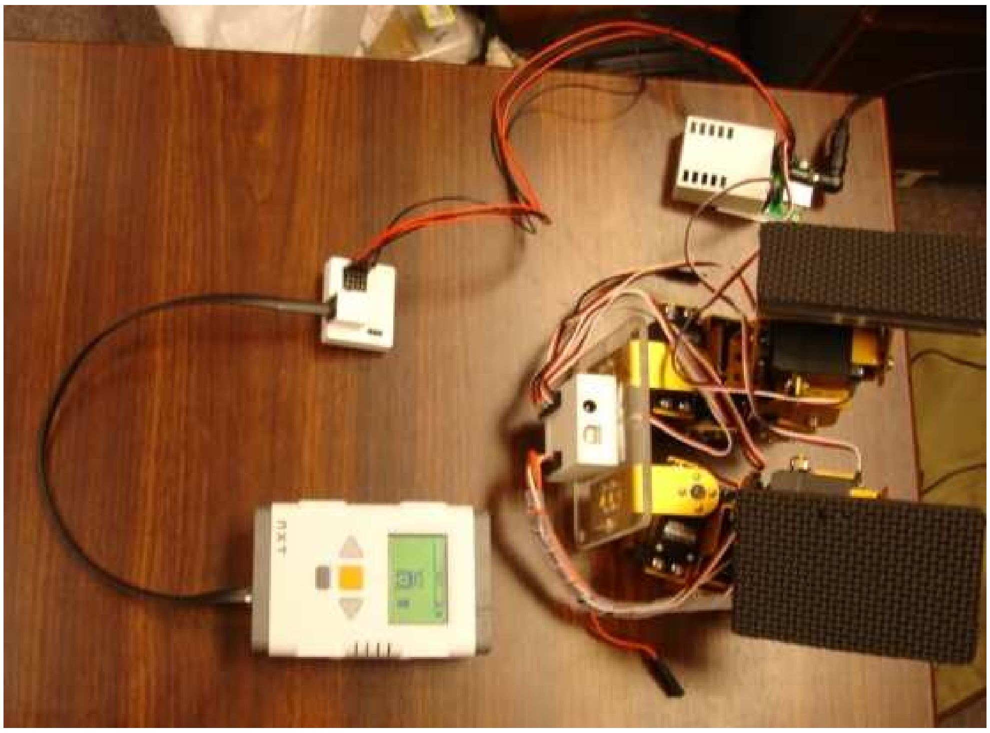
With a NXT Brick and a NXTe Kit, it is very easy to manage a RC Car.





4.4.2.- Manage a biped

With a NXT brick and a NXTe with a unique LSC it is possible to manage a biped.



4.5.- LeJOS and Lattebox

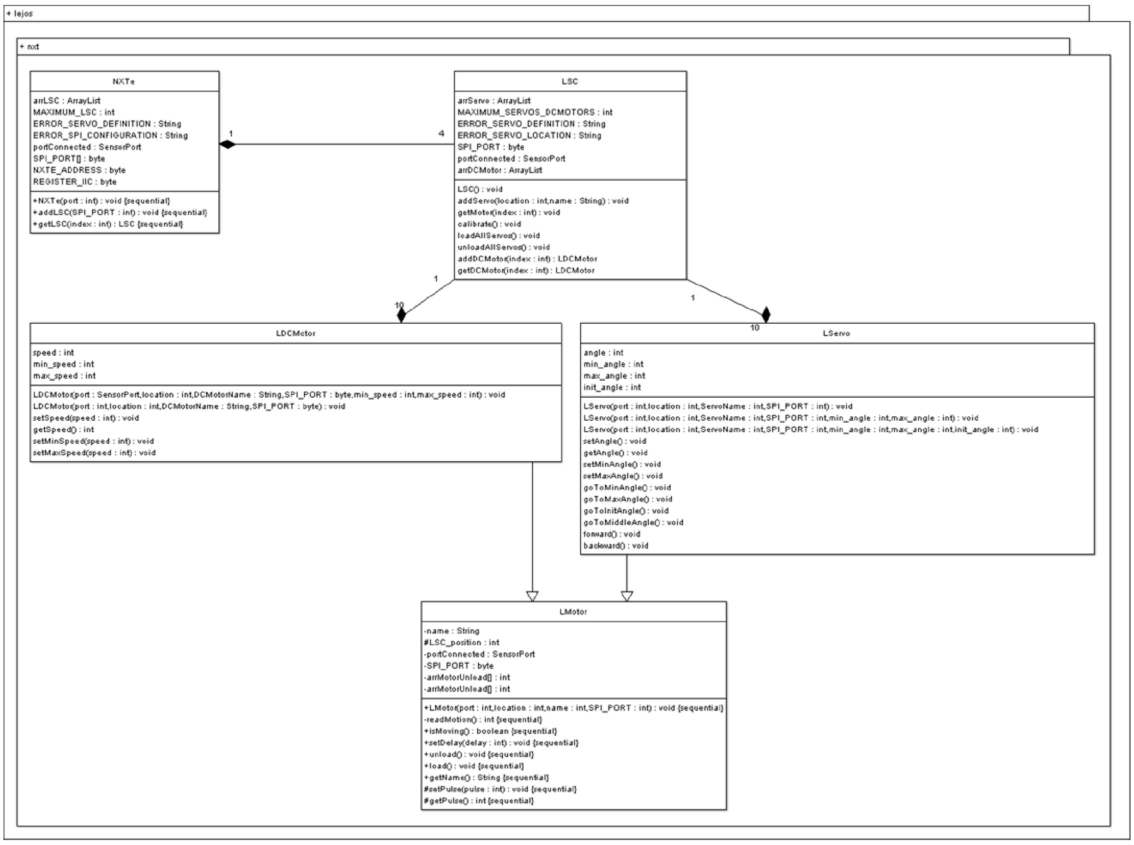
4.5.1.- Lattebox Support in LeJOS

Currently leJOS support NXTe Kit but it is necessary to improve:

- New tests for delay methods
- Test I2C Pin to manage others I2C Sensor plugged to NXTe

4.5.2.- UML

The UML Diagram to understand the classes which manages NXTe Kit is above:


+ lejos
+ nxt
NXTe
arrLSC : ArrayList
MAXIMUM_LSC : int
ERROR_SERVO_DEFINITION : String
ERROR_SPI_CONFIGURATION : String
portConnected : SensorPort
SPI_PORT[] : byte
NXTE_ADDRESS : byte
REGISTER_IIC : byte
+NXTe(port : int) : void {sequential}
+addLSC(SPI_PORT : int) : void {sequential}
+getLSC(index : int) : LSC {sequential}
1
4
LSC
arrServo : ArrayList
MAXIMUM_SERVOS_DCMOTORS : int
ERROR_SERVO_DEFINITION : String
ERROR_SERVO_LOCATION : String
SPI_PORT : byte
portConnected : SensorPort
arrDCMotor : ArrayList
LSC() : void
addServo(location : int,name : String) : void
getMotor(index : int) : void
calibrate() : void
loadAllServos() : void
unloadAllServos() : void
addDCMotor(index : int) : LDCMotor
getDCMotor(index : int) : LDCMotor
1
10
1
10
LDCMotor
speed : int
min_speed : int
max_speed : int
LDCMotor(port : SensorPort,location : int,DCMotorName : String,SPI_PORT : byte,min_speed : int,max_speed : int) : void
LDCMotor(port : int,location : int,DCMotorName : String,SPI_PORT : byte) : void
setSpeed(speed : int) : void
getSpeed() : int
setMinSpeed(speed : int) : void
setMaxSpeed(speed : int) : void
LServo
angle : int
min_angle : int
max_angle : int
init_angle : int
LServo(port : int,location : int,ServoName : int,SPI_PORT : int) : void
LServo(port : int,location : int,ServoName : int,SPI_PORT : int,min_angle : int,max_angle : int) : void
LServo(port : int,location : int,ServoName : int,SPI_PORT : int,min_angle : int,max_angle : int,init_angle : int) : void
setAngle() : void
getAngle() : void
setMinAngle() : void
setMaxAngle() : void
goToMinAngle() : void
goToMaxAngle() : void
goToInitAngle() : void
goToMiddleAngle() : void
forward() : void
backward() : void
LMotor
-name : String
#LSC_position : int
-portConnected : SensorPort
-SPI_PORT : byte
-arrMotorUnlead[] : int
-arrMotorUnlead[] : int
+LMotor(port : int,location : int,name : int,SPI_PORT : int) : void {sequential}
-readMotion() : int {sequential}
+isMoving() : boolean {sequential}
+setDelay(delay : int) : void {sequential}
+unload() : void {sequential}
+load() : void {sequential}
+getName() : String {sequential}
#setPulse(pulse : int) : void {sequential}
#getPulse() : int {sequential}

5.- Multimedia

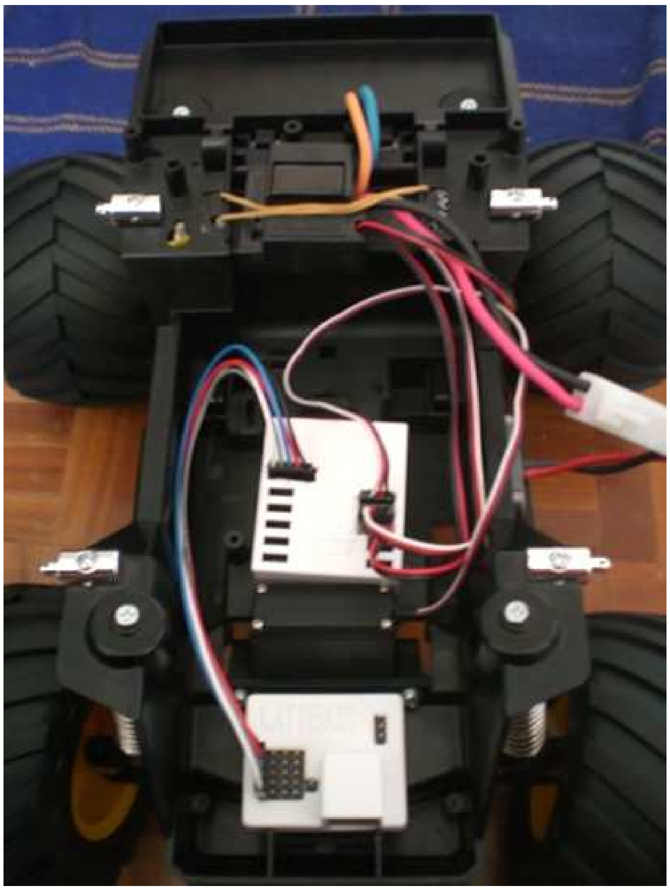
5.1.- Introduction

I use Youtube.com as the platform to upload videos about the progress of the project. Besides I am uploading photos in my personal website in the document 228, the URL is: http://www.juanantonio.info/jab_cms.php?id=228

5.2.- Images published

5.2.1.- Connections

This is the way to connect a RC Car with NXTe Kit. Normally a RC Car has a speed controller. Connect Control wire which manages DC Motor with LSC in position 3. Connect RC Servo into LSC in position 1, connect a red wire into position 5 to give energy from RC battery to LSC.



5.3.- Videos published

5.3.1.- Video with a communication tests

URL: http://www.youtube.com/watch?v=XsE3gZE9QQE

5.3.2.- Video about forward and backward methods with a DC Motor.

URL: http://www.youtube.com/watch?v=ZNzFxLcv2Hs

URL: http://www.youtube.com/watch?v=PTEps_1Bu5c

6.- Source Code

6.1.- Introduction

I am releasing source code, when I consider that the code is "stable"

The code released:

1. RCController
    a. Nothing. It is an experimental code for the moment.
2. TLB Protocol
    a. Nothing. It is an experimental code for the moment.
3. LTB
    a. RacingTLB.java
4. System utilities
    a. RCDiagnostics3.java
    b. CalibrateServo.java
    c. CalibrateDCMotor.java

6.2.- TLBProtocol

6.3.- TLB

RacingTLB.java

```
import lejos.nxt.*;

/**
 * @author Juan Antonio Brenha Moral
 *
 */
public class RacingTLB extends TamiyaLunchBox{

      private UltrasonicSensor USL;
      private UltrasonicSensor USC;
      private UltrasonicSensor USR;

      public RacingTLB(int NXTePort,int USPort1, int USPort2, int
USPort3){
            super(NXTePort);

            //Ultrasonic panel designed to detect frontal obstacles
            USL = new UltrasonicSensor(SensorPort.PORTS[USPort1]);
            USC = new UltrasonicSensor(SensorPort.PORTS[USPort2]);
            USR = new UltrasonicSensor(SensorPort.PORTS[USPort3]);
      }

      public int getLeftDistance(){
            return USL.getDistance();
      }

      public int getFrontalDistance(){
            return USC.getDistance();
      }

      public int getRightDistance(){
            return USC.getDistance();
      }
}
```

6.4.- System Utilities

RCDiagnostics2.java

```
import lejos.nxt.*;

/**
 * RCDiagnostics has been developed to calibrate your RC Car.
 *
 * @author Juan Antonio Brenha Moral
 *
 */
public class RCDiagnostics2 {

      //Input Devices
      private static Motor speedServo;
      private static Motor steeringServo;

      //Speed
      private static int currentSpeed;
      private static int previousSpeed;
      private static int diffSpeed;
      private static int speed;
      private static int realSpeedValue;
      private static boolean fb;

      //Steering
      private static int currentSteering;
      private static int previousSteering;
      private static int diffSteering;
      private static int steering;
      private static int realSteeringValue;
      private static boolean lr;

      //Limits
      private static final int maxSpeed = 100;
      private static final int minSpeed = 0;
      private static final int maxSteering = 100;
      private static final int minSteering = 0;

      //Others
      private static final String clearIntStr = "
";

      //Tamiya Lunch Box
      private static TamiyaLunchBox TLB;

      private static int motion;
      private static int DMminSpeed;
	private static int DMmaxSpeed;
	private static int DMspeed;
	private static int RSminAngle;
	private static int RSmaxAngle;
	private static int RSangle;

	public static void main(String[] args) throws Exception{

		speedServo = Motor.A;
		steeringServo = Motor.C;
		speedServo.resetTachoCount();
		steeringServo.resetTachoCount();

		LCD.clear();

		speed = 0;
		steering = 45;

		previousSpeed = speed;
		previousSteering = steering;

		TLB = new TamiyaLunchBox(0);
		DMminSpeed = TLB.getMinSpeed();
		DMmaxSpeed = TLB.getMaxSpeed();
		RSminAngle = TLB.getMinAngle();
		RSmaxAngle = TLB.getMaxAngle();

		boolean flagInit = true;

		LCD.clear();

		while(!Button.ESCAPE.isPressed()){

			realSpeedValue = speedServo.getTachoCount();
			currentSpeed = Math.abs(realSpeedValue);

			realSteeringValue = steeringServo.getTachoCount();
			currentSteering = Math.abs(realSteeringValue);
			diffSpeed = currentSpeed - previousSpeed;
			diffSteering = currentSteering - previousSteering;

			if(diffSpeed >= 0){
				if((speed + diffSpeed) <= maxSpeed){
					speed = speed + diffSpeed;
				}else{
					speed = maxSpeed;
				}
			}else{
				if((speed + diffSpeed) >= minSpeed){
					speed = speed + diffSpeed;
				}else{
					speed = minSpeed;
				}
			}

			//Corrections
			if(speed == 0){
				speedServo.resetTachoCount();
				currentSpeed = 0;
			}
```

```
//To know if RC has to drive forward or backward
if(realSpeedValue <= 0){
      fb = true;
}else{
      fb = false;
}

previousSpeed = currentSpeed;

if(diffSteering >= 0){
      if((steering + diffSteering) <= maxSteering){
            steering = steering + diffSteering;
      }else{
            steering = maxSpeed;
      }
}else{
      if((steering + diffSteering) >= minSteering){
            steering = steering + diffSteering;
      }else{
            steering = minSteering;
      }
}

if(steering == 0){
      steeringServo.resetTachoCount();
      currentSteering = 0;
}

//To know if RC has to turn left or right
if(realSteeringValue <= 0){
      lr = false;
}else{
      lr = true;
}

previousSteering = currentSteering;

LCD.drawString("RC Diagnostics",0,0);

LCD.drawString("Speed",0,2);
LCD.drawString(clearIntStr,0,3);
LCD.drawInt(speed,0,3);
if(fb){
      LCD.drawString("+",4,3);

      //Forward Method
      DMspeed = getDCMotorSpeed(speed);
      TLB.setSpeed(DMspeed);
}else{
      LCD.drawString("-",4,3);
}

LCD.drawString("Steering",8,2);
LCD.drawInt(steering,10,3);
if(lr){
      LCD.drawString("<",14,3);

}else{
      LCD.drawString(">",14,3);
}
```

```
			//Steering
			RSangle = getRCServoAngle(steering);
			TLB.setSteeringAngle(RSangle);
			LCD.refresh();
		}

		LCD.drawString("Program finished.",0,7);
		LCD.refresh();

		//At the end, unload all Servos
		TLB.turnOff();

	}//End Main

	private static int getDCMotorSpeed(int speed){
		float DCspeed = 
getLinealInterpolation(speed,minSpeed,maxSpeed,DMminSpeed,DMmaxSpeed);

		return Math.round(DCspeed);
	}

	private static int getRCServoAngle(int speed){
		float RSspeed = 
getLinealInterpolation(speed,minSteering,maxSteering,RSminAngle,RSmaxA
ngle);
		return Math.round(RSspeed);
	}

	/*
	From the HP Calculator idea:
	http://h10025.www1.hp.com/ewfrf/wc/fastFaqLiteDocument?lc=es&cc=
mx&docname=bsia5214&dlc=es&product=20037
	*/
	private static float getLinealInterpolation(int x,int x1, int 
x2, int y1, int y2){
		float y;
		y = ((y2-y1)/(x2-x1))*(x-x1) + y1;

		return y;
	}
}
```

CalibrateServo.java

```
import lejos.nxt.*;

public class CalibrateServo{
	private static NXTe NXTeObj;
	private static DebugMessages dm;
	private static int angle;
	private static int angle2;
	private static int motion;
	private static boolean direction = false;

	//Main
	public static void main(String[] args) throws Exception{
		dm = new DebugMessages();
		dm.setLCDLines(6);
		dm.echo("Testing NXTe");

		try{
```

```
                  NXTeObj = new NXTe(SensorPort.S4);//NXTe Controller
plugged in Port1
                  NXTeObj.addLSC(0);
                  dm.echo("Calibrating LSC");
                  //Servo 1 connected in location 1
                  NXTeObj.getLSC(0).addServo(1,"SAVOX, Digital SC-
0352");
                  //NXTeObj.getLSC(0).addDCMotor(3,"Tamiya DC Motor,
MABUCHI Motor RS 540SH");
                  NXTeObj.getLSC(0).calibrate();
                  NXTeObj.getLSC(0).getServo(0).load();
                  NXTeObj.getLSC(0).getServo(0).setMinAngle(0);
                  NXTeObj.getLSC(0).getServo(0).setMaxAngle(2000);


                  while(!Button.ESCAPE.isPressed()){

                        if (Button.LEFT.isPressed()){

      NXTeObj.getLSC(0).getServo(0).goToMinAngle();

      while(NXTeObj.getLSC(0).getServo(0).isMoving() == true){}
                              angle =
NXTeObj.getLSC(0).getServo(0).getAngle();
                              dm.echo("Goto Min");
                              dm.echo(angle);
                        }

                        if (Button.ENTER.isPressed()){

      NXTeObj.getLSC(0).getServo(0).goToMiddleAngle();

      while(NXTeObj.getLSC(0).getServo(0).isMoving() == true){}
                              angle =
NXTeObj.getLSC(0).getServo(0).getAngle();
                              dm.echo("Goto Mid");
                              dm.echo(angle);
                        }

                        if (Button.RIGHT.isPressed()){

      NXTeObj.getLSC(0).getServo(0).goToMaxAngle();

      while(NXTeObj.getLSC(0).getServo(0).isMoving() == true){}
                              angle =
NXTeObj.getLSC(0).getServo(0).getAngle();
                              dm.echo("Goto Max");
                              dm.echo(angle);
                        }
                  }

            }catch(Exception e){
                  dm.echo(e.getMessage());
            }

            //At the end, unload all Servos
            NXTeObj.getLSC(0).unloadAllServos();
            dm.echo("Test finished");
      }
}
```

CalibrateDCMotor.java

```java
import lejos.nxt.*;

/**
 *
 * @author Juan Antonio Brena Moral
 *
 */
public class CalibrateDCMotor {
      private static TamiyaLunchBox TLB;
      private static Motor speedServo;

      private static int currentSpeed = 0;
      private static int previousSpeed = 0;
      private static int diffSpeed = 0;
      private static int speed = 0;
      private static int realSpeedValue = 0;
      private static boolean fb = true;

      //Limits
      private static final int maxSpeed = 100;
      private static final int minSpeed = 0;
      private static final int maxSteering = 100;
      private static final int minSteering = 0;

      private static int DMForwardMinSpeed =
TLBProtocol.getDMForwardMinSpeed();
      private static int DMForwardMaxSpeed =
TLBProtocol.getDMForwardMaxSpeed();
      private static int DMBackwardMinSpeed =
TLBProtocol.getDMBackwardMinSpeed();
      private static int DMBackwardMaxSpeed =
TLBProtocol.getDMBackwardMaxSpeed();
      private static int RSminAngle = TLBProtocol.getMinAngle();//0;
      private static int RSmaxAngle =
TLBProtocol.getMaxAngle();//2000;

      /**
       * @param args
       */
      public static void main(String[] args) {
            try{
                  //touch = new TouchSensor(SensorPort.S1);
                  //NXTe connected in Port 4
                  TLB = new TamiyaLunchBox(3);
                  speedServo = Motor.A;

                  while(!Button.ESCAPE.isPressed()){

                        if (Button.ENTER.isPressed()){
                              TLB.backwardStop();
                              //Reset Internal values
                              speedServo.resetTachoCount();
                              realSpeedValue =
speedServo.getTachoCount();
                              currentSpeed = Math.abs(realSpeedValue);
                              currentSpeed = 0;
                              previousSpeed = 0;
                              diffSpeed = currentSpeed - previousSpeed;
```

```
                                speed = 0;
                                fb = false;
                        }

                        speed = getSpeed();
                        drawSpeed();

                        int _speed = 0;;
                        //Speed
                        LCD.drawString("    ",0,4);
                        if(fb){
                                //Forward Method
                                _speed = 
TLBProtocol.getDCMotorSpeedForward(speed);//getDCMotorSpeed(speed);
                        }else{
                                _speed = 
TLBProtocol.getDCMotorSpeedBackward(speed);//getDCMotorSpeed2(speed);
                        }
                        LCD.drawInt(_speed,0,4);
                        LCD.refresh();
                        TLB.setSpeed(_speed);
                }

                LCD.clear();
                LCD.drawString("Program finished", 0, 7);
                LCD.refresh();
                Thread.sleep(1000);
                TLB.turnOff();
                System.exit(0);
        }catch(Exception e){

        }
    }

    private static int getSpeed(){
        realSpeedValue = speedServo.getTachoCount();
        currentSpeed = Math.abs(realSpeedValue);

        //realSteeringValue = steeringServo.getTachoCount();
        //currentSteering = Math.abs(realSteeringValue);
        diffSpeed = currentSpeed - previousSpeed;
        //diffSteering = currentSteering - previousSteering;

        if(diffSpeed >= 0){
                if((speed + diffSpeed) <= maxSpeed){
                        speed = speed + diffSpeed;
                }else{
                        speed = maxSpeed;
                }
        }else{
                if((speed + diffSpeed) >= minSpeed){
                        speed = speed + diffSpeed;
                }else{
                        speed = minSpeed;
                }
        }

        //Corrections
        if(speed == 0){
                speedServo.resetTachoCount();
                currentSpeed = 0;
        }

        //To know if RC has to drive forward or backward
        if(realSpeedValue <= 0){
                fb = true;
        }else{
                fb = false;
        }

        previousSpeed = currentSpeed;

        return speed;
    }

    private static void drawSpeed(){
        LCD.drawString("Speed",0,2);
        LCD.drawString("   ",0,3);
        LCD.drawInt(speed,0,3);
        if(fb){
                LCD.drawString("+",4,3);
        }else{
                LCD.drawString("-",4,3);
        }
        LCD.refresh();
    }
}
```

7.- Links

7.1.- Java

http://www.cica.es/formacion/JavaTut/Cap7/flujo.html
http://www.cica.es/formacion/JavaTut/Fuentes/Cap7/TubTest.java
http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/threads/subprocesos.htm

7.2.- LeJOS NXJ

http://lejos.sourceforge.net/forum/
http://forums.nxtasy.org/

7.3.- Software Engineering

http://argouml.tigris.org/

7.4.- Autonomous Mobile Robots

http://www.mobilerobots.ethz.ch/
http://www.idsia.ch/%7Ejuergen/robotcars.html
http://www.cybercars.org/
http://www.isr.uc.pt/%7Eurbano/WorkICRA07/
http://teamster.usc.edu/%7Emoradi/iros07/iv_iros07.htm
http://www.asl.ethz.ch/robots/Smartter
http://emotion.inrialpes.fr/
http://ais.informatik.uni-freiburg.de/research/index_en.php
http://www.isr.uc.pt/%7Eurbano/publications/index.html
http://wwwlasmea.univ-bpclermont.fr/Control/index_fichiers/agv/uvc/uvc.html
http://rlai.cs.ualberta.ca/openpages2/CMPUT412+2008
http://www-scf.usc.edu/%7Ecsci445/