Develop leJOS Programs Step by Step

Versión 0.4

Juan Antonio Breña Moral

7-jul-08

Index

Revision History

Name	Date	Reason For Changes	Version
Juan Antonio Breña Moral	12/01/2008		0.1
Juan Antonio Breña Moral	18/02/2008	Add FAQ section	0.2
Juan Antonio Breña Moral	09/03/2008	Add Tortoise SVN Section	0.3
Juan Antonio Breña Moral	23/06/2008	Add leJOS RC Car Project	0.4

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.

Juan Antonio Breña Moral.
www.juanantonio.info

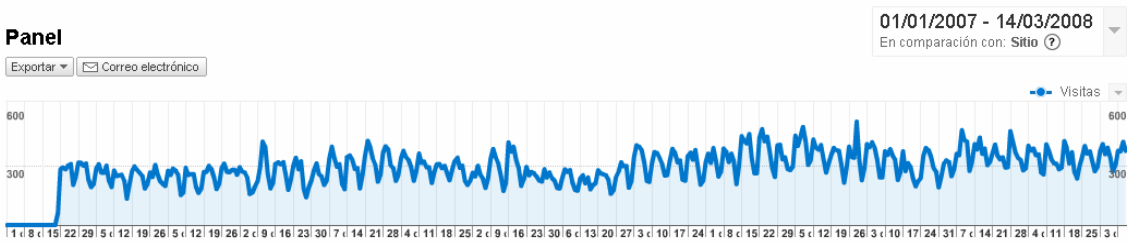
1.2.- LeJOS Project

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. 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
- RFID Sensor

NXT Motor

Ultrasonic Sensor

Compass Sensor

Tilt Sensor


NXTCam

RFID Sensor

Lattebox 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

1.5.- About the collaborators

Frank Zimmermann is a Doctor in Mathematics and Professor for CIS at the University of Applied Sciences Nordakademie since 1996. Frank teaches Java, Software Engineering and Information Systems at the university. He discovered leJOS and NXT Technology in 2007.

Further information:
http://fermat.nordakademie.de/
http://www.nordakademie.de/

Patrick Lismore a Napier University Edinburgh student finishing his Bsc (Hons) Software Technology. An aspiring entrepreneur and IT professional who have experience teaching programming and robotics at Carnegie Mellon University. Patrick first got involved with leJOS and NXT's while studying in his last year of University. Patrick research at University involved designing and developing concurrent robotics software using leJOS, JCSP re and Bluetooth.

Further information:
http://www.newirelandcomputing.net
http://www.patricklismore.com

2.- NXJ

2.1.- Introduction

NXJ is a project to develop a JVM for NXT Lego Mindstorm. NXJ is the evolution of leJOS RCX, the JVM for RCX Lego Mindstorm.

In this eBook, we learn how to use NXJ JVM to use the packages that leJOS community improves everyday.

If you want to develop robots with NXT Brick technology NXJ, Java for Lego Mindstorm, it is necessary to congurate your computer and your enviroment to develop.

2.2.- Configure your computer

2.2.1.- Prerequisites

If you have just purchased your new NXT Lego Mindstorm or your have a NXT Brick and you can experiment with NXJ, this Chapter is essential to test your first program *"HelloWorld.java"* into your NXT brick.

Read carefully this chapter to make the settings correctly.

NOTE: Some windows in these chapters have been captured using a Spanish Windows XP Professional Edition.

The software that you need to start programming into NXJ are:

1. Lego Mindstorm NXT Software CD
2. Latest Java Developer Kit
3. LibUSB Filter Driver for Microsoft Windows
4. Latest NXJ Release

2.2.2.- Lego Mindstorm NXT Software

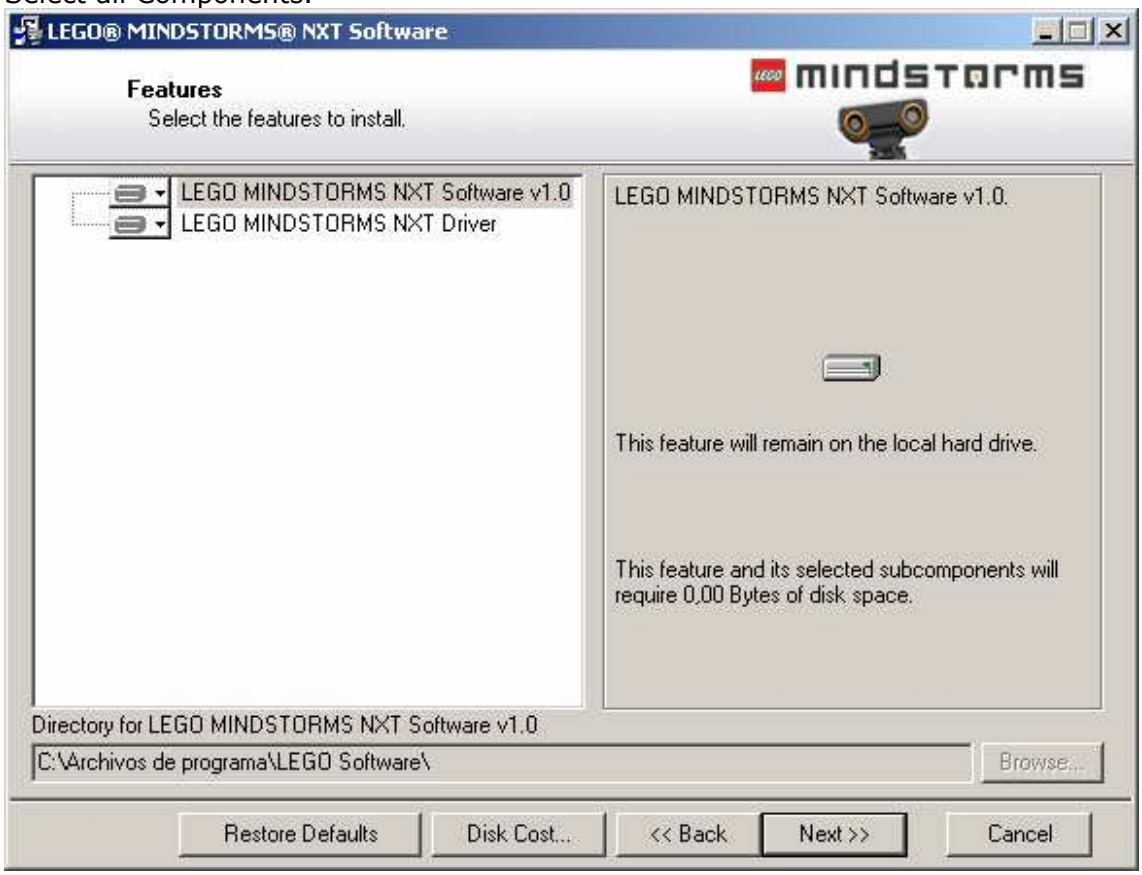
When you purchase your Lego Mindstorm NXT Kit, it includes a CD. If you run the CD you can install an IDE to program with NXT-G platform. Besides, this CD includes Windows Driver to recognize Lego Devices in Windows Operating Systems.

Follow the steps to install correctly the Lego Mindstorm NXT Driver in your computer:

First Step: Execute your CD in your Desktop PC or Laptop with Windows Operating System. The installer will make a question about the language.

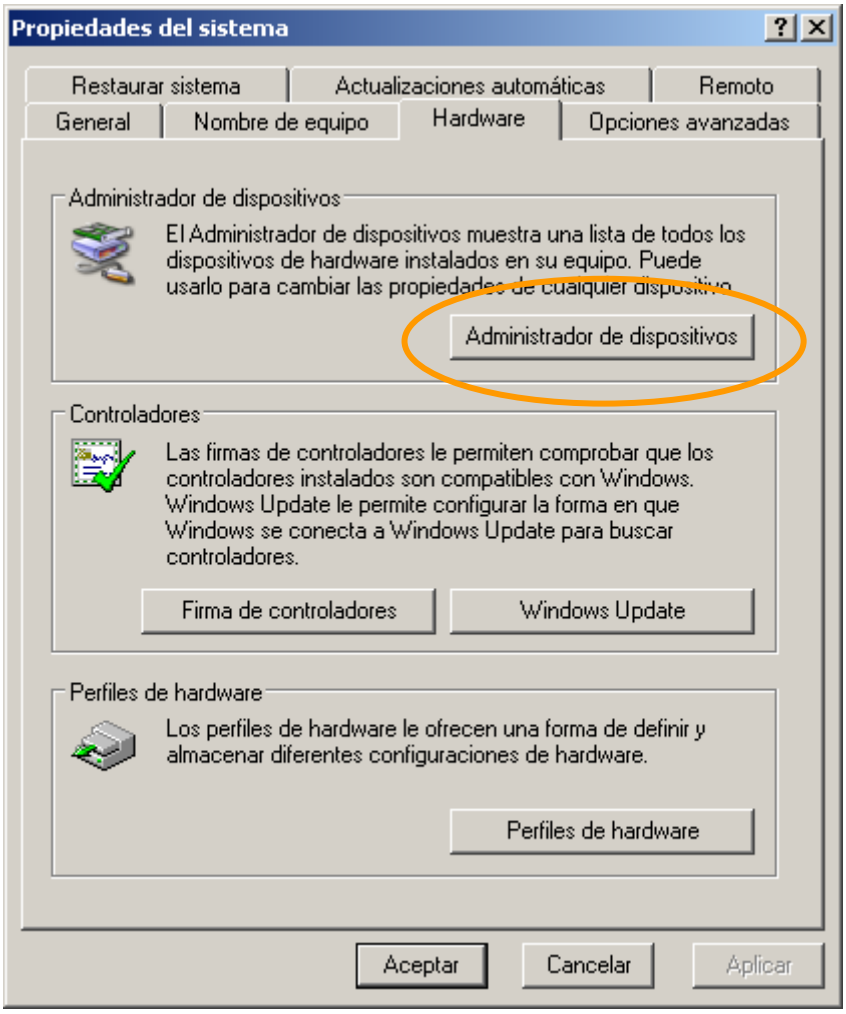
Step2: Select the components to install.

Select all Components.

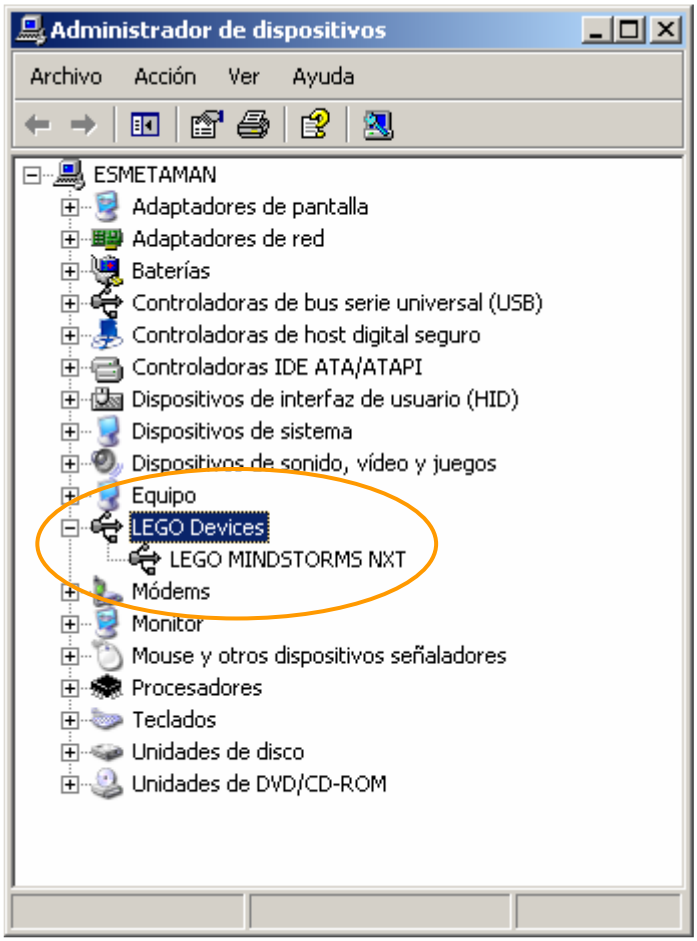


The first time when you connect your NXT Brick with your computer, Windows doesn't recognize your Brick, Windows will think that NXT brick is a USB device. In this moment update the driver. If you update the driver, Windows recognize correctly that you have connected a NXT Brick.

If you want to check this installation process, right click with the mouse on PC Icon and enter in properties.



Click in the button Manage Devices then you will see all devices connected / installed in your computer:



Note: In this point of your installation, Windows Operating System recognizes NXT Bricks connected into your computer.

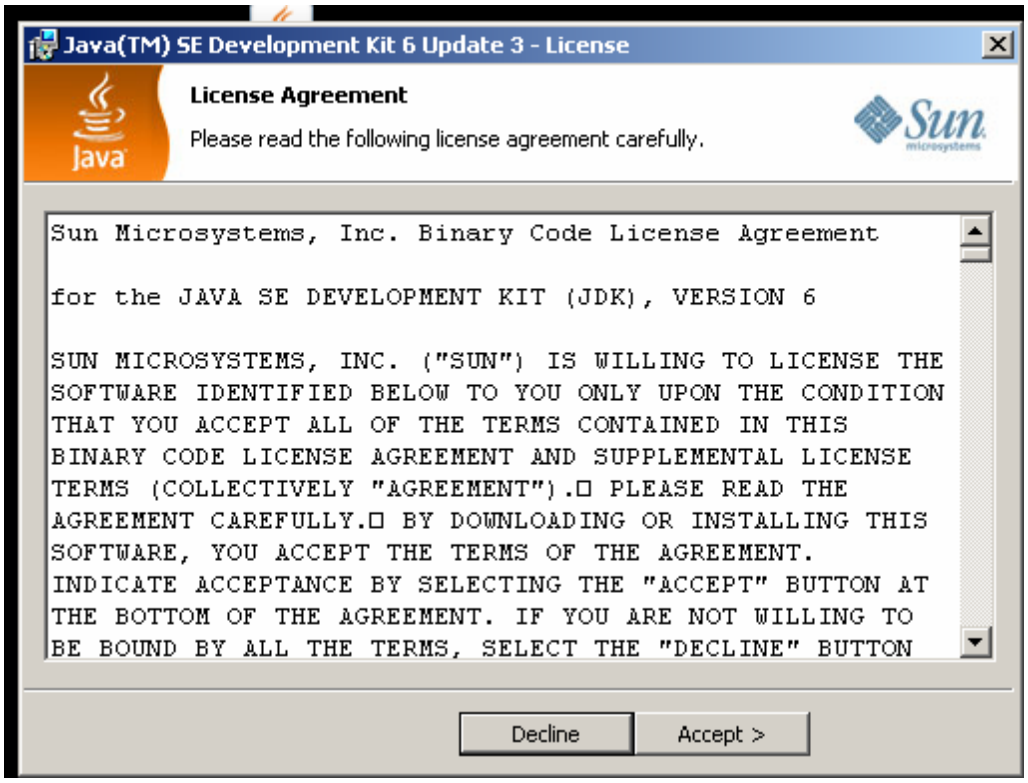
2.2.3.- Java Developer Kit

If you want to develop with NXJ, you must to install the Java SDK, because is the unique way to create programs with Java. The programming language used to develop NXJ programs is Java. Download the latest Java SDK from http://java.sun.com/javase/downloads/index.jsp . When you have downloaded your latest Java SDK use the assistant to install Java SDK.

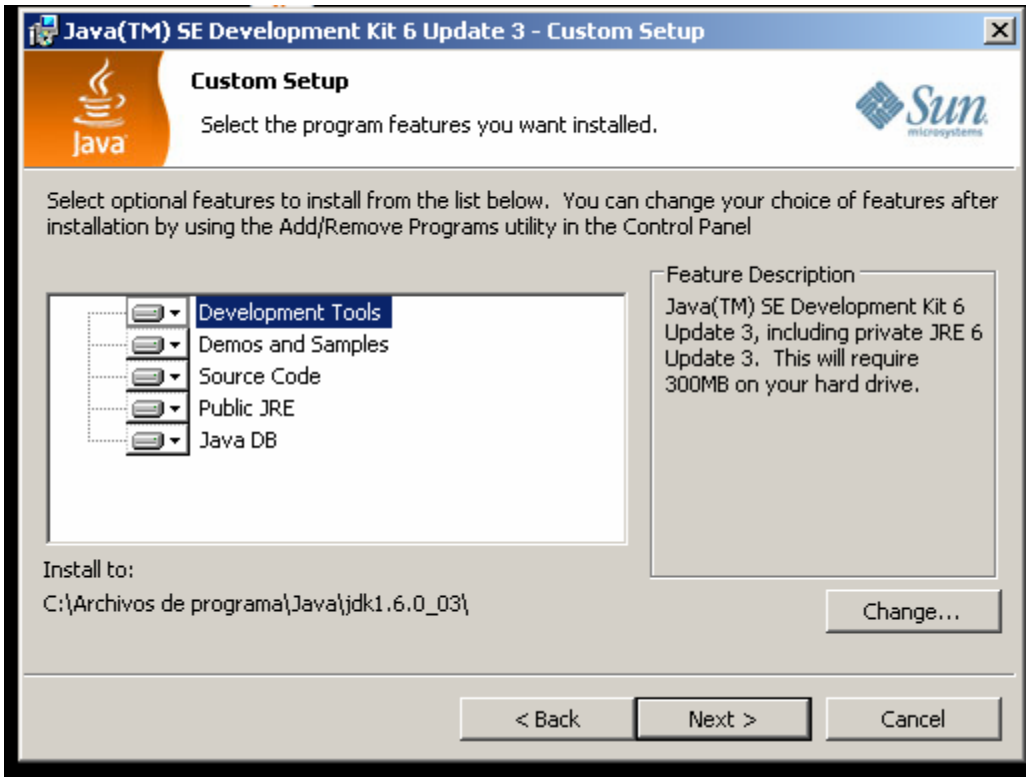
NOTE: To create this document, the installer used is: jdk-6u3-windows-i586-p.exe

The necessary steps to install the latest Java Developer Kit are:

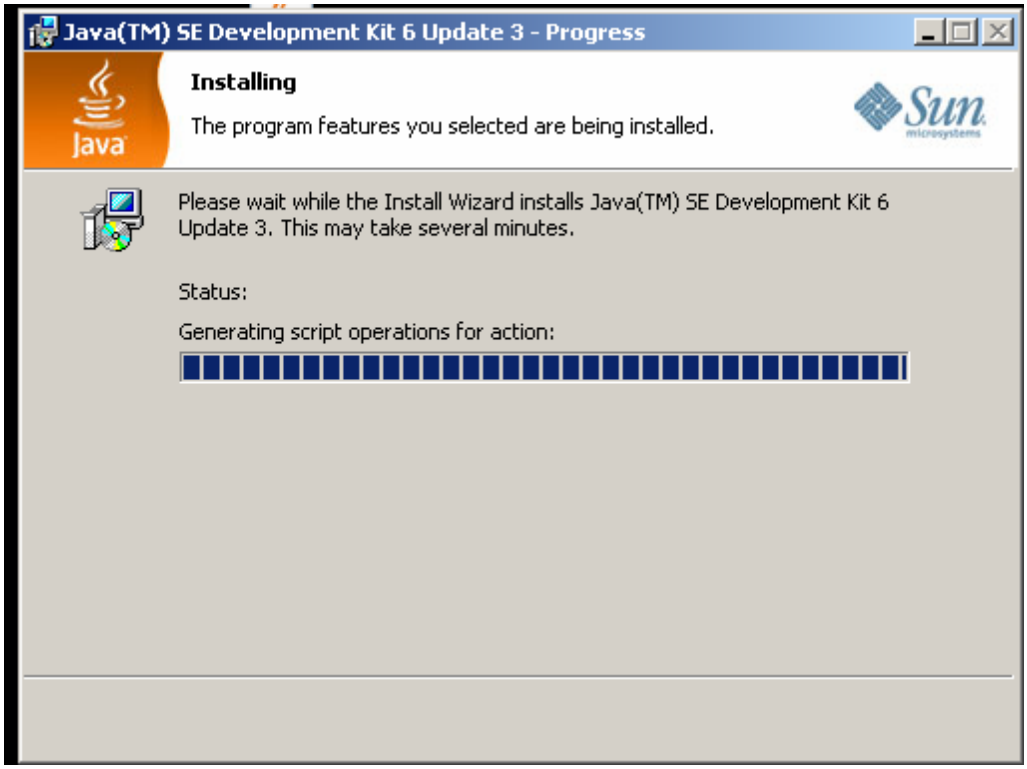
Step1: Accept the licence.



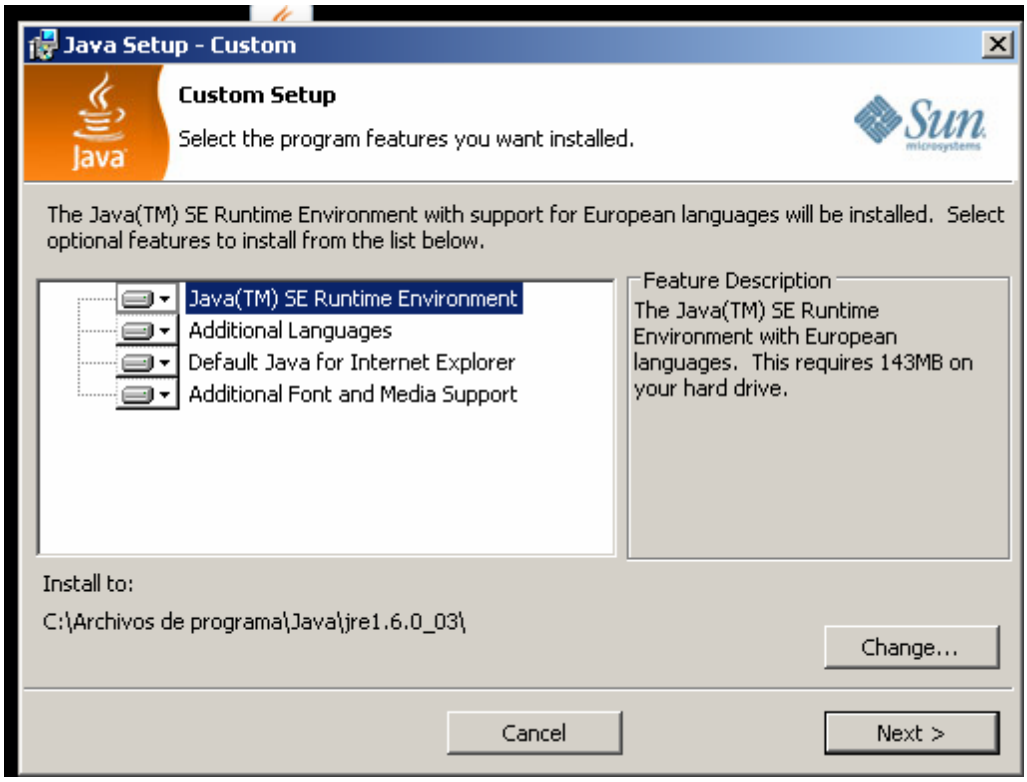
Step2: Select the components to install.



Once you have chosen the components, the assistant will install them into your computer.



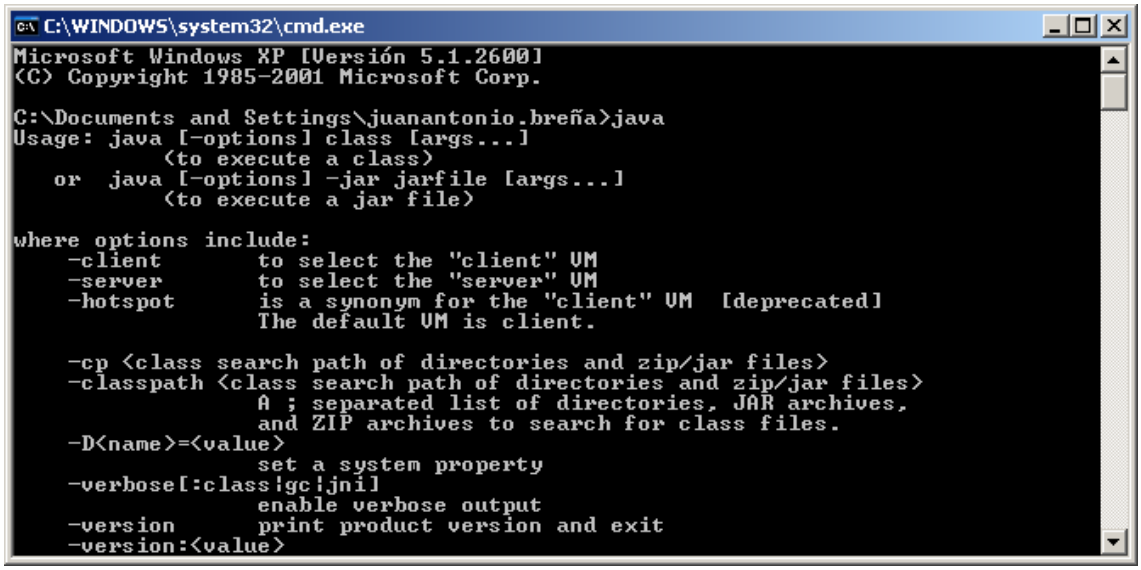
When the installer has finished with the component's installations, this one makes another question about optional features. Select all features.



Once you have selected all features, they will be installed.

Note: In this point of the installation, your computer is able to create any Java Program, test it.

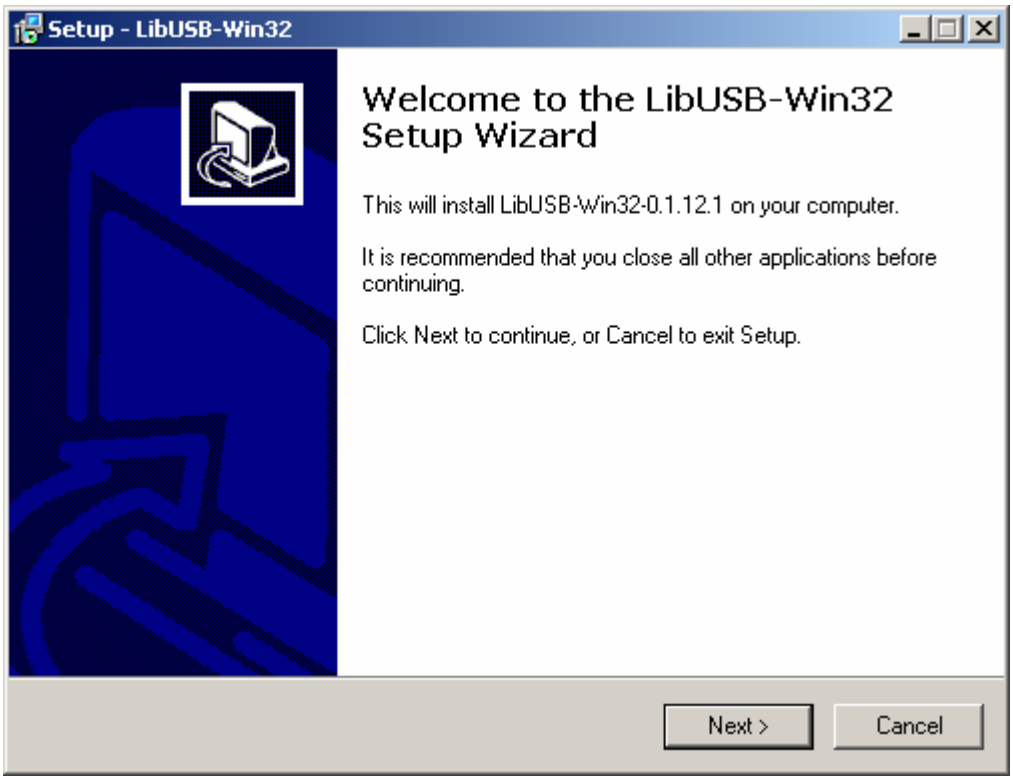
Open console window and type java:

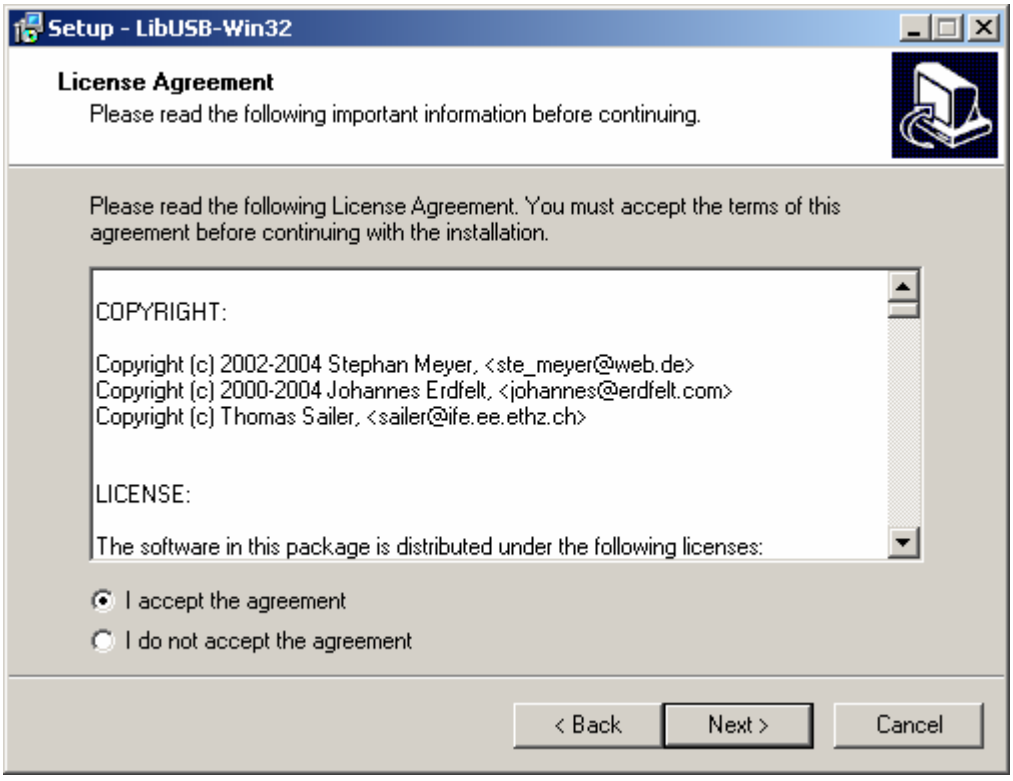


2.2.4.- LibUSB Filter Driver for Microsoft Windows

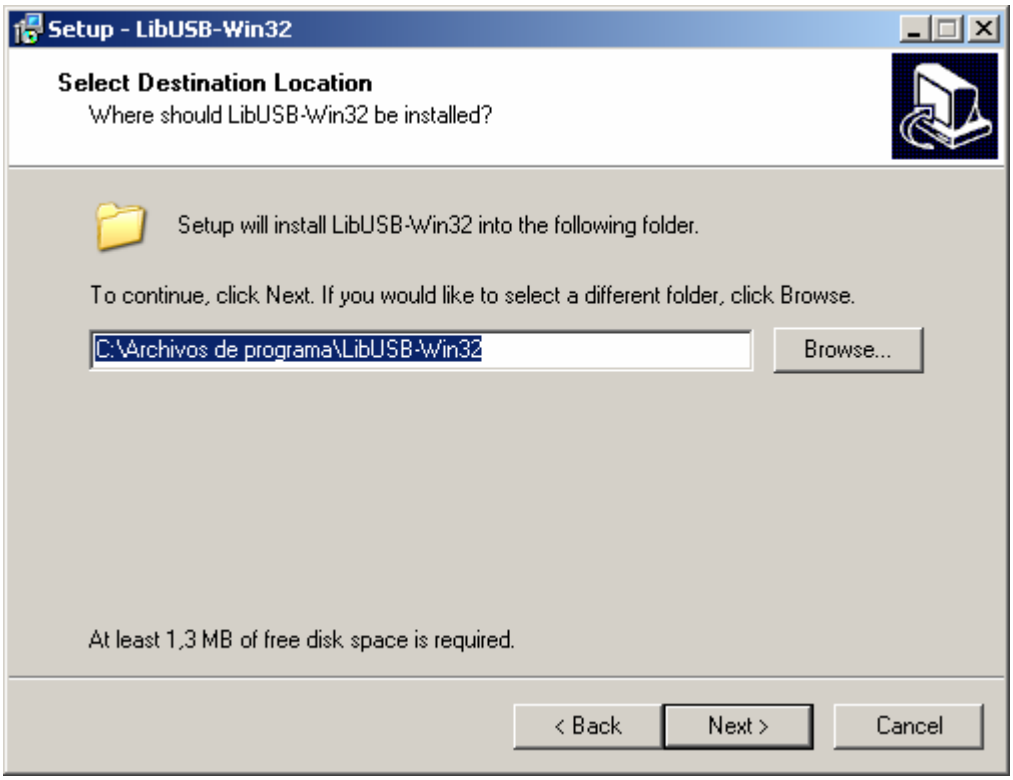
LibUSB is a library which allows accessing any USB device on Windows in a generic way without writing any line of kernel driver code. LeJOS for NXT Lego Mindstorm uses LibUSB to make several actions. To download LibUSB for Windows http://libusb-win32.sourceforge.net/. When you have the installer, execute it.

Step1: Accept the licence.

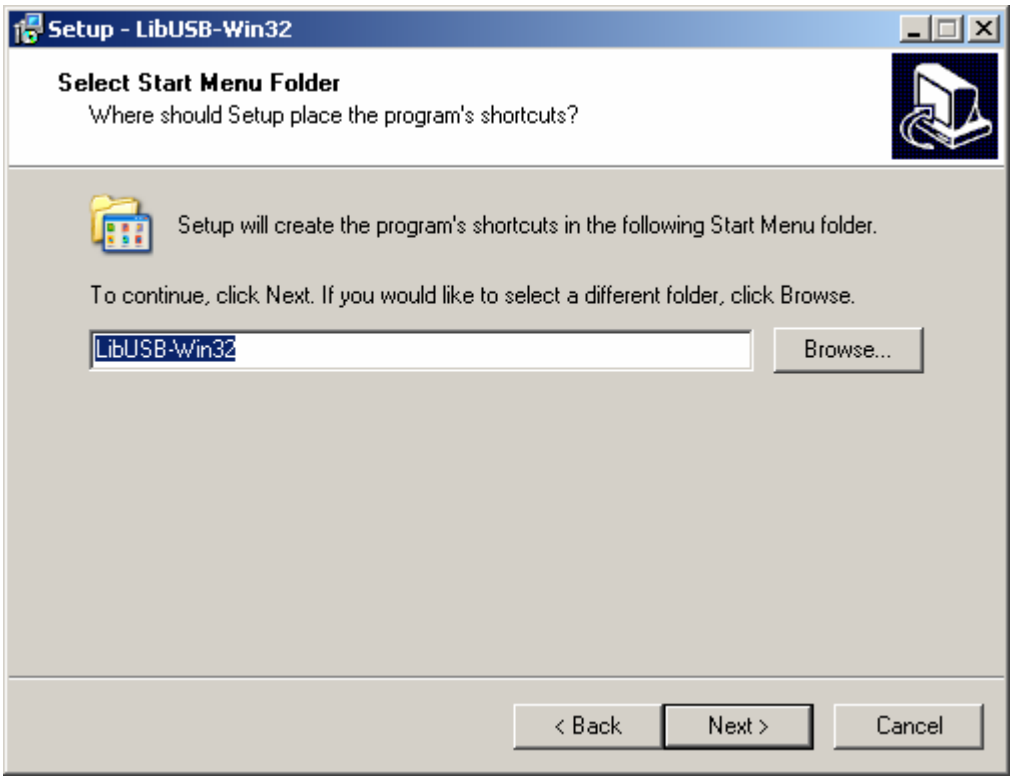




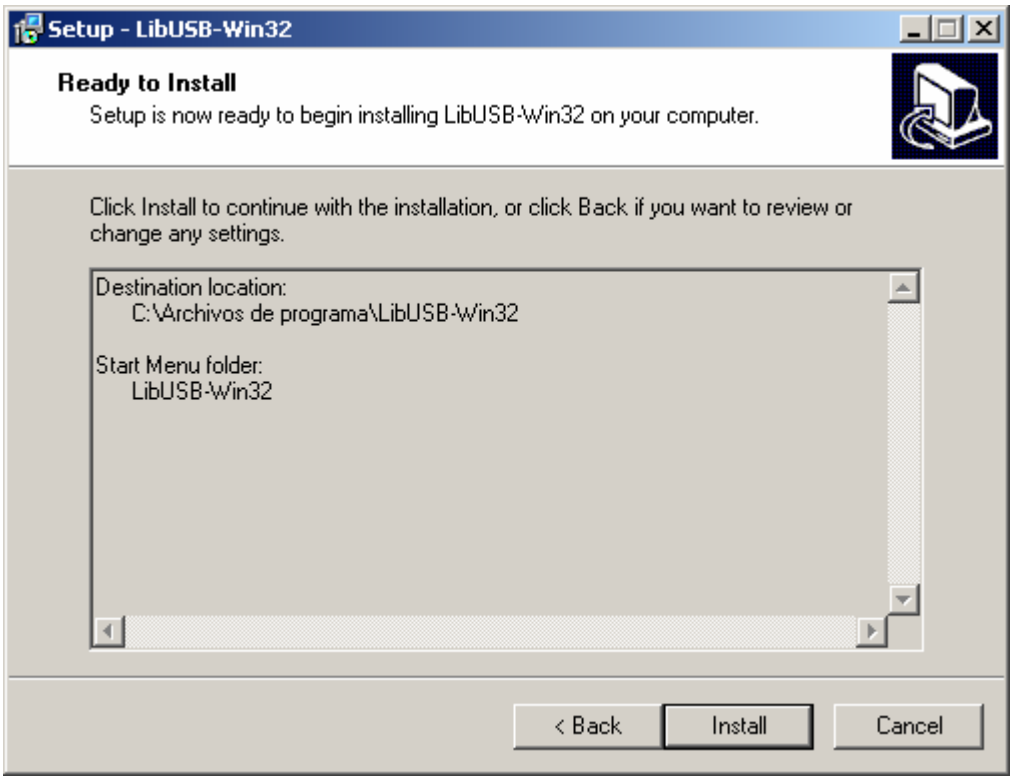
Step2: Select where you want to install the software

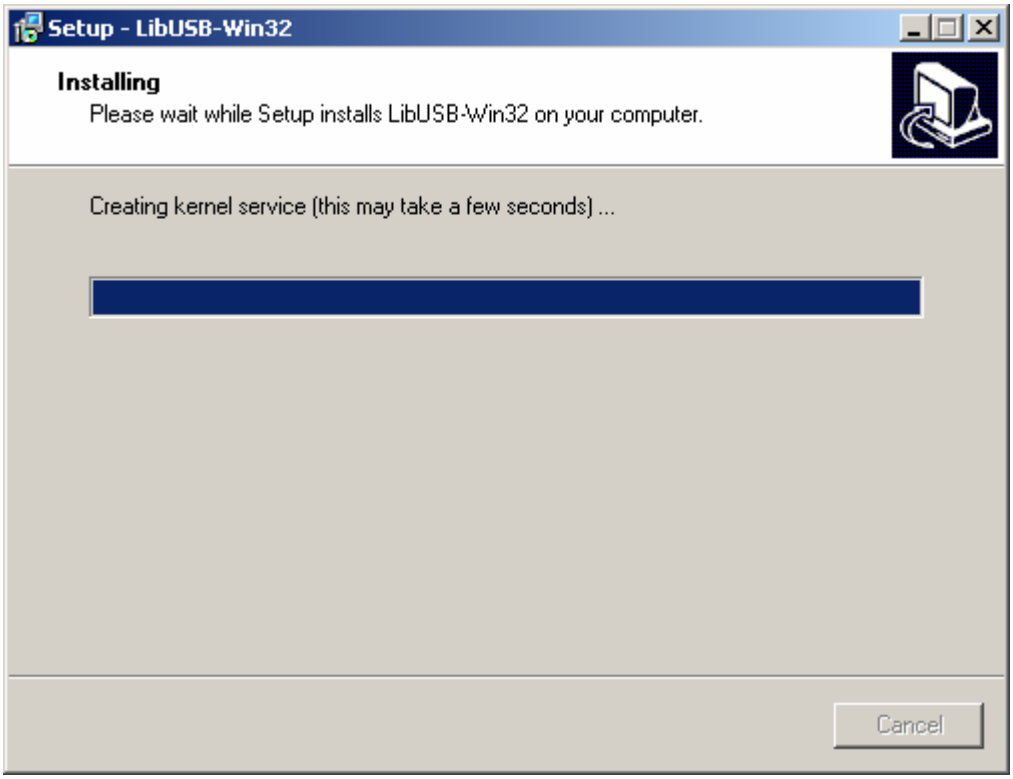


Step3: Choose the place when you will see in Windows Start menu

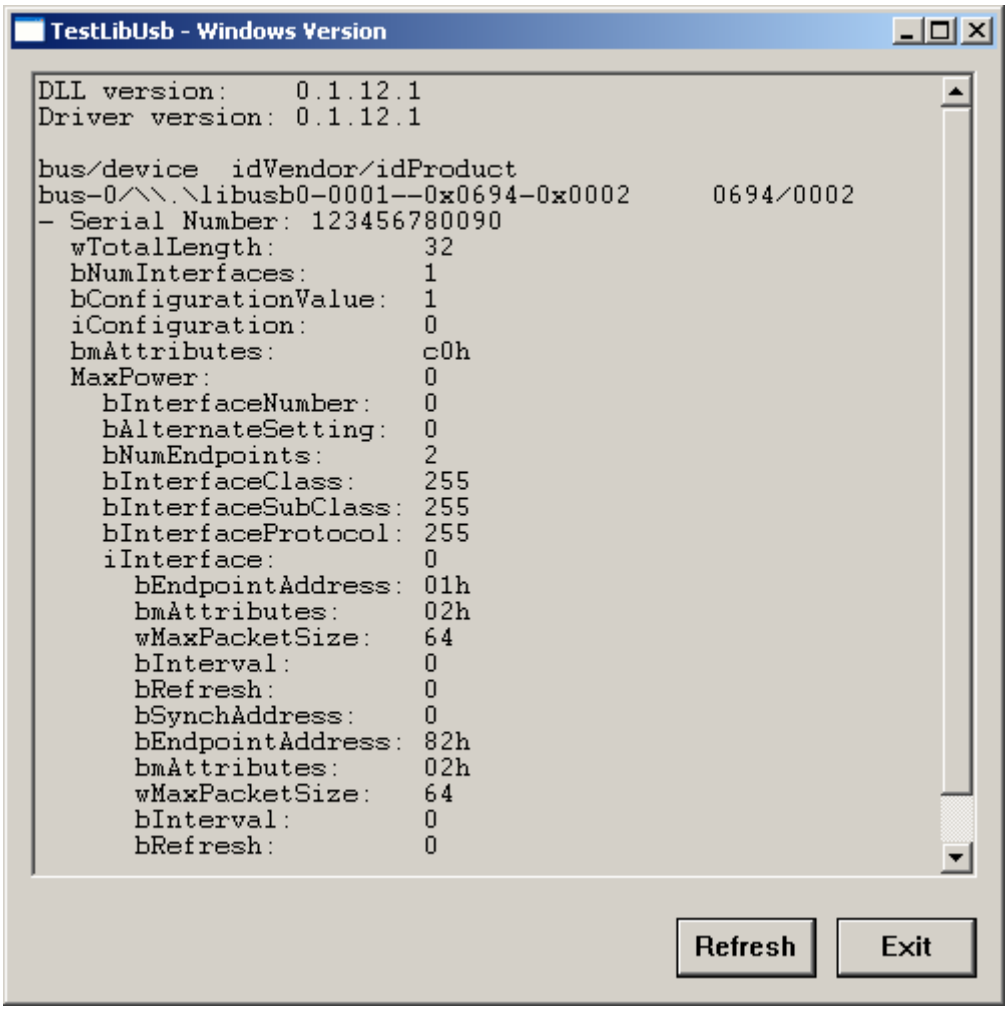


When you have indicated these options, the assistant will Install LibUSB. Once LibUSB has been installed, the assistant will start LibUSB Services in Windows.





If you want to see LibUSB in action, test the program included with the installation. Connect NXT Brick with the computer with USB Wire and turn on the brick. Execute the LibUSB test program, you should see:

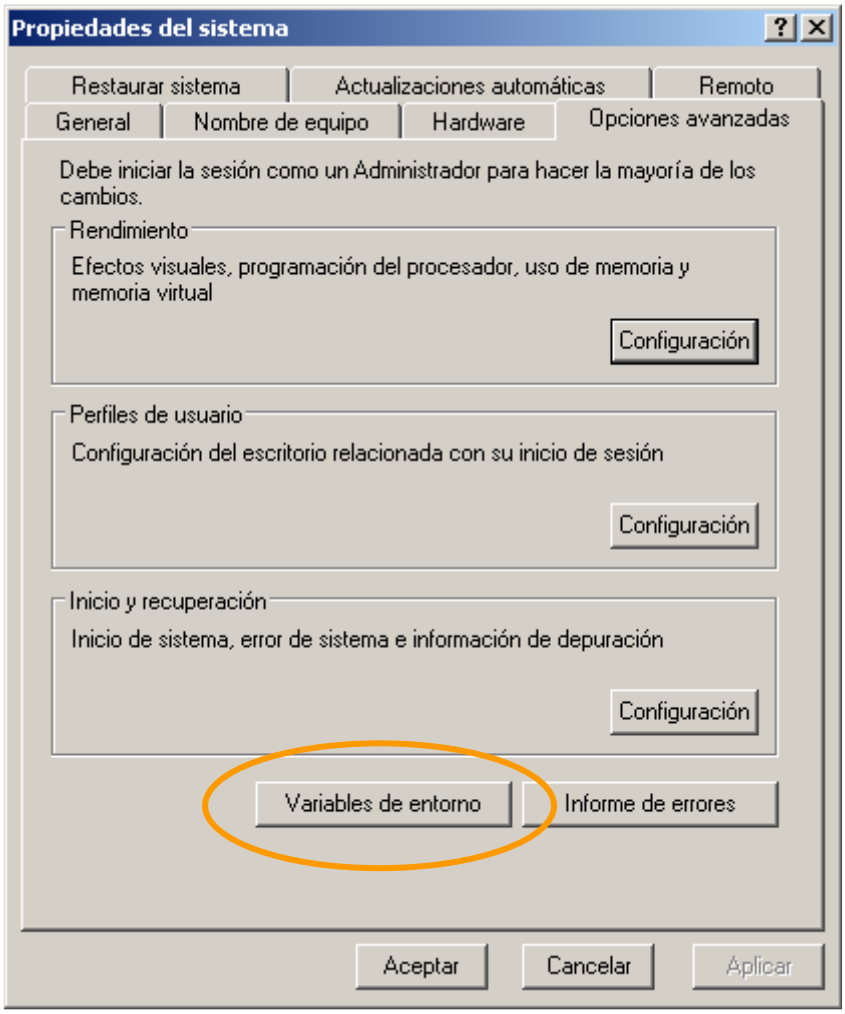


2.2.5.- LeJOS NXJ

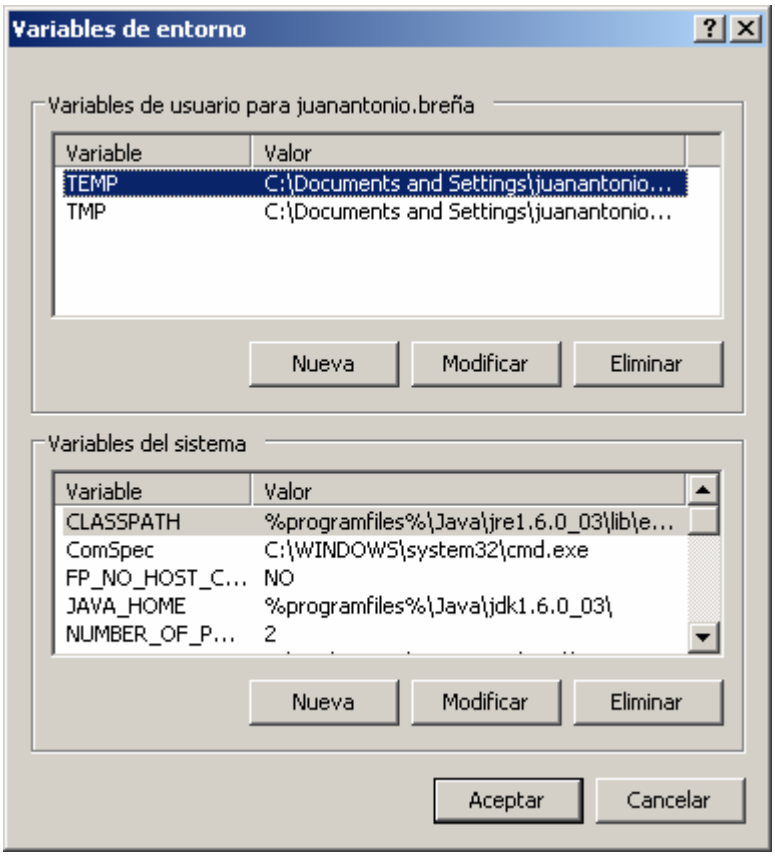
NXJ is a LeJOS project which is evolving along the time with new features. Currently LeJOS Research Team has launched the release 0.5. To download: http://sourceforge.net/project/showfiles.php?group_id=9339&package_id=217619

Once you have downloaded zip file, unzip it and place into your favourite place into your computer for example in the path: C:/DATA/ROBOTICS/NXJ/

In this moment, you have to access to MyPC properties in the environment variables section.



In this window you can see the environment variables that your windows Operating system uses:



It is necessary to create 2 variables and update a variable to finish the installation's process.

Step1: Create the environment variable JAVA_HOME
NXJ needs to know where your JDK is installed.

For example if you have installed jdk1.6.0_03, search in your computer the installation path. It is very important to put the final **"\"**

JAVA_HOME
C:\Archivos de programa\Java\jdk1.6.0_03\

Step2: Create the environment variable NXJ_HOME
It is necessary to establish where your NXJ Installation is.

NXJ_HOME
C:\DATA\ROBOTICS\NXJ\leJOS _NXJ_win32_0_5_0beta\leJOS _nxj

NOTE: without final "\" in the path

Step3: Update the environment variable PATH
Environment variable PATH is used by several applications and systems. In your case you had to update these variable indicating 2 things:

1. Where is located bin folder in your JDK installation
2. Where is located bin folder in your NXJ installation

This is an example about the content of a right PATH:

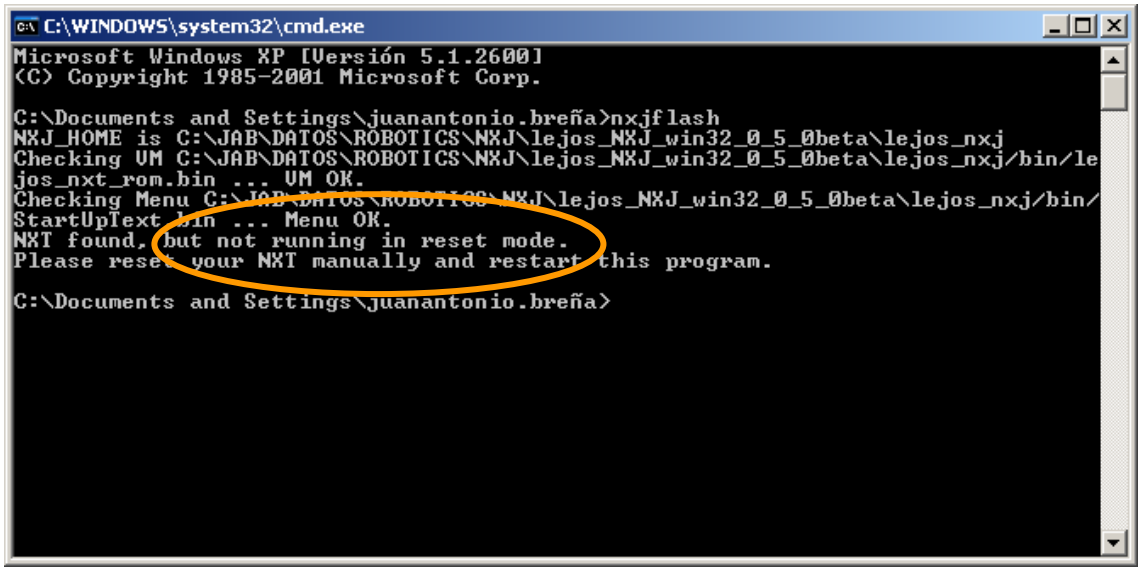
PATH

%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;C:\Archivos de programa\ATI Technologies\ATI.ACE\Core-Static;c:\Archivos de programa\Microsoft SQL Server\90\Tools\binn\;C:\Archivos de programa\QuickTime\QTSystem\;%programfiles%\Java\jdk1.6.0_03\bin\;C:\DATA\ROBOTICS\NXJ\leJOS_NXJ_win32_0_5_0beta\leJOS_nxj\bin;

Step4: Reboot your computer
Reboot your computer to make effect your changes in environment variables

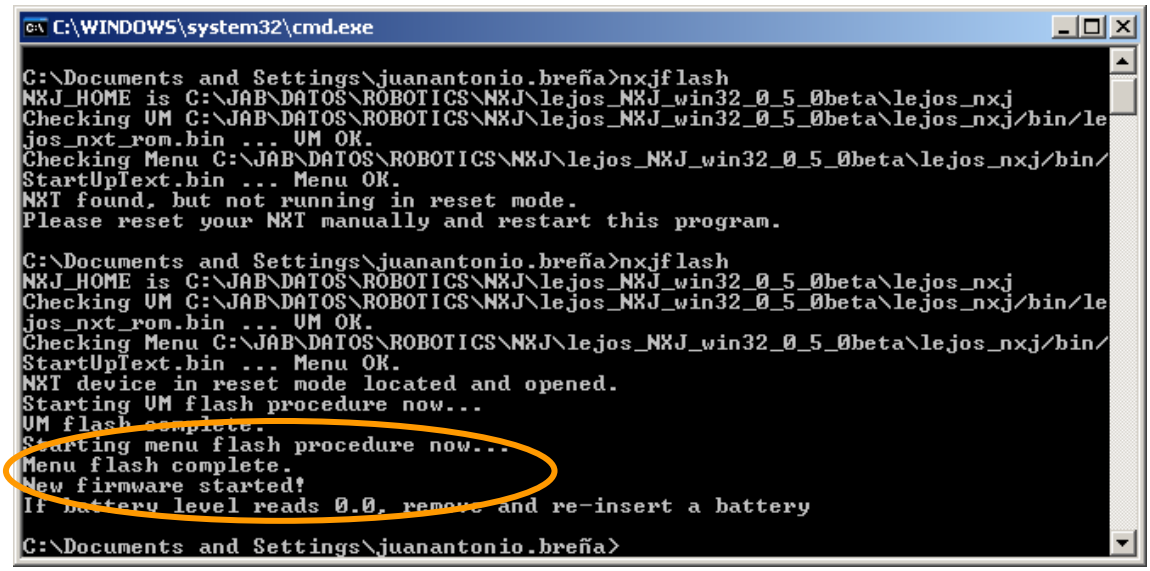
Step4: Install NXJ into your NXT Brick
Once you have installed your developer environment, it is necessary to install NXJ into your NXT Brick.

The command to this action is nxjflash. If you do this action before changing the mode in your NXT brick then you will see in your console window the following message:



To update the mode in your NXT brick then you have to push reset button. To find that button see at the back of the NXT, upper left corner and push it for more than 5 seconds then you will hear an audibly sound. In this moment you can install NXJ.

Repeat the previous action in your windows console:



If you reach this step then your brick understand NXJ, congratulations!

This is the moment to test your first program, HelloWorld.java

2.3.- Configure your enviroment to develop

2.3.1.- Introduction

Develop any NXJ program is easy and you could develop with notepad.

2.3.2.- Develop with Eclipse IDE and NXJ Plugin

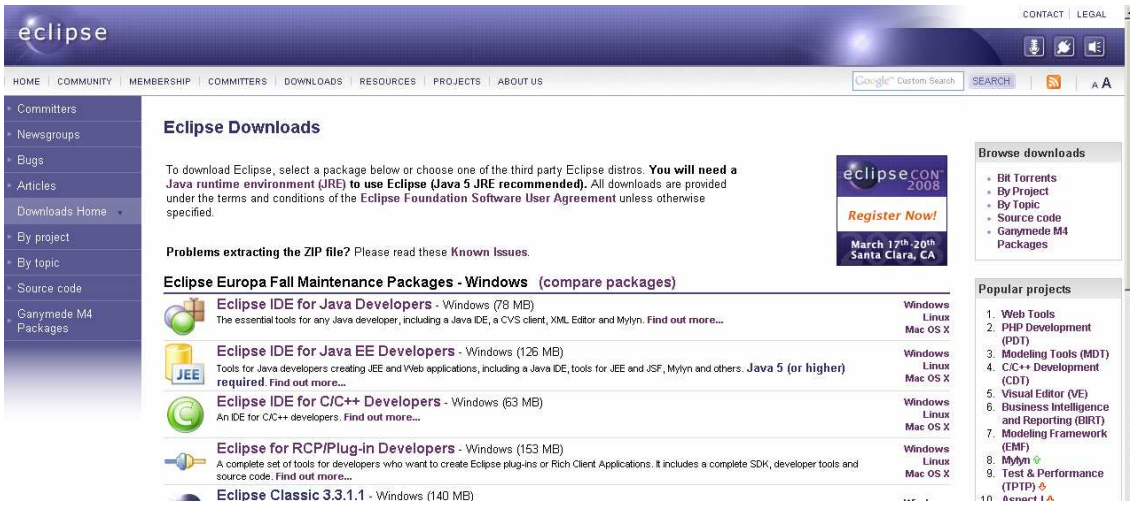
2.3.2.1.- Introduction

Eclipse is an extensible, open source IDE (integrated development environment). The project was originally launched in November 2001, when IBM donated $40 million worth of source code from Websphere Studio Workbench and formed the Eclipse Consortium to manage the continued development of the tool.

The stated goals of Eclipse are "to develop a robust, full-featured, commercial-quality industry platform for the development of highly integrated tools." To that end, the Eclipse Consortium has been focused on three major projects:

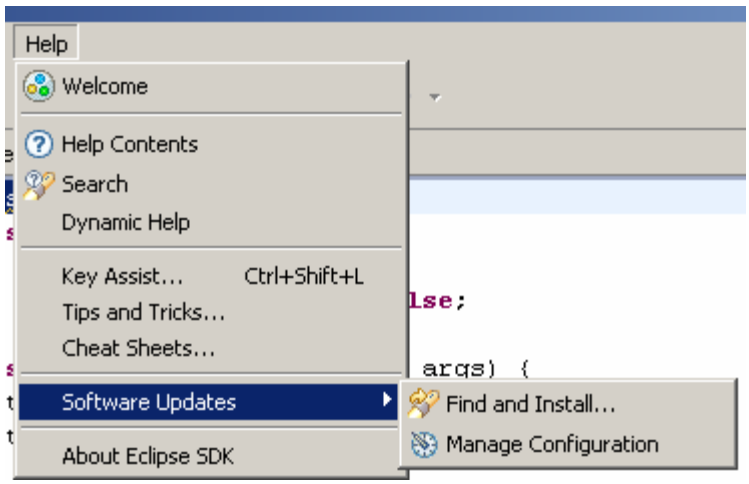
1. The Eclipse Project is responsible for developing the Eclipse IDE workbench (the "platform" for hosting Eclipse tools), the Java Development Tools (JDT), and the Plug-In Development Environment (PDE) used to extend the platform.
2. The Eclipse Tools Project is focused on creating best-of-breed tools for the Eclipse platform. Current subprojects include a Cobol IDE, a C/C++ IDE, and an EMF modeling tool.
3. The Eclipse Technology Project focuses on technology research, incubation, and education using the Eclipse platform.

Download Eclipse Europa 3.3 Classic from eclipse's website. Use the following URL: http://www.eclipse.org/downloads/

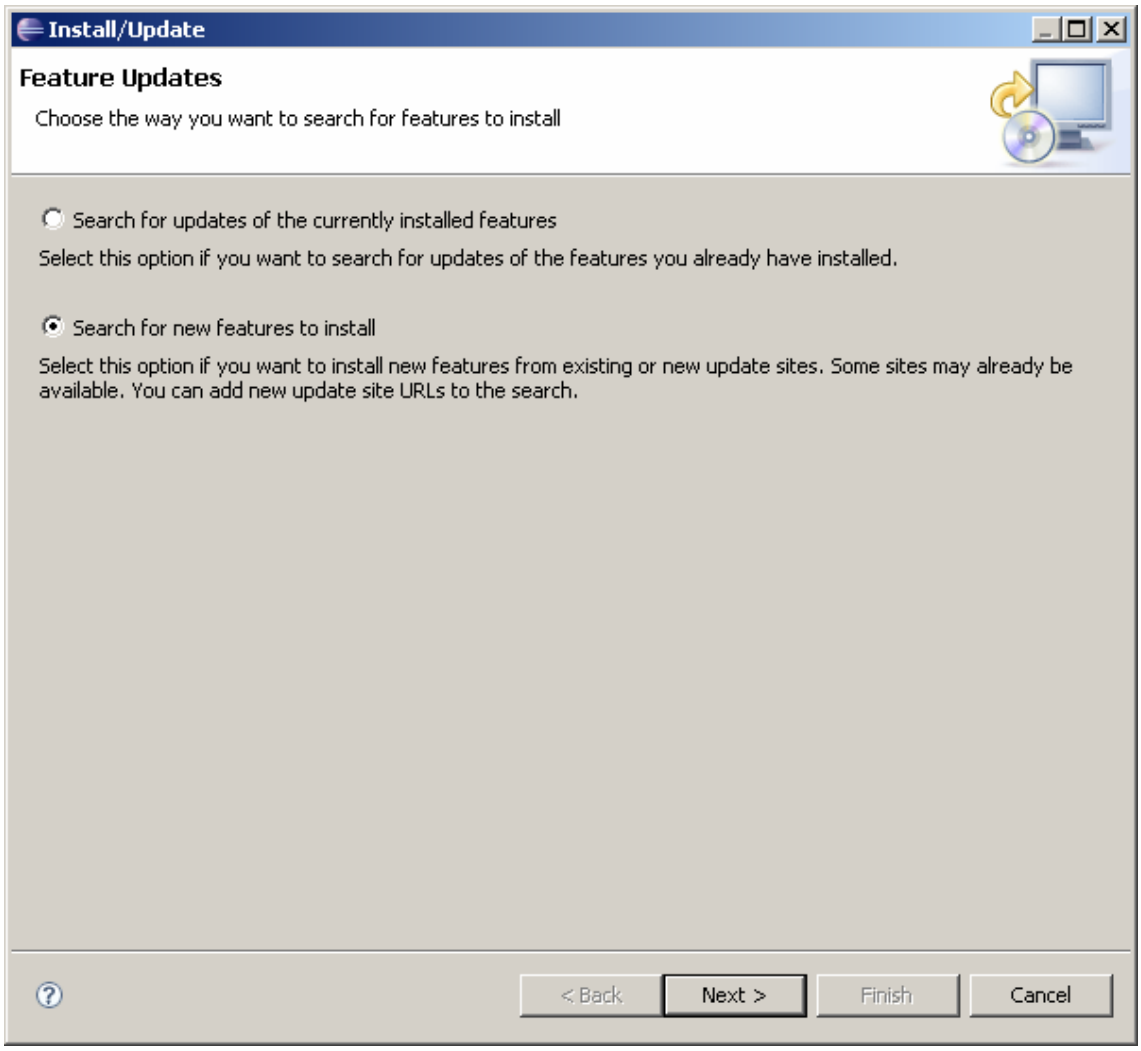


2.3.2.2.- Install NXJ Plug-in

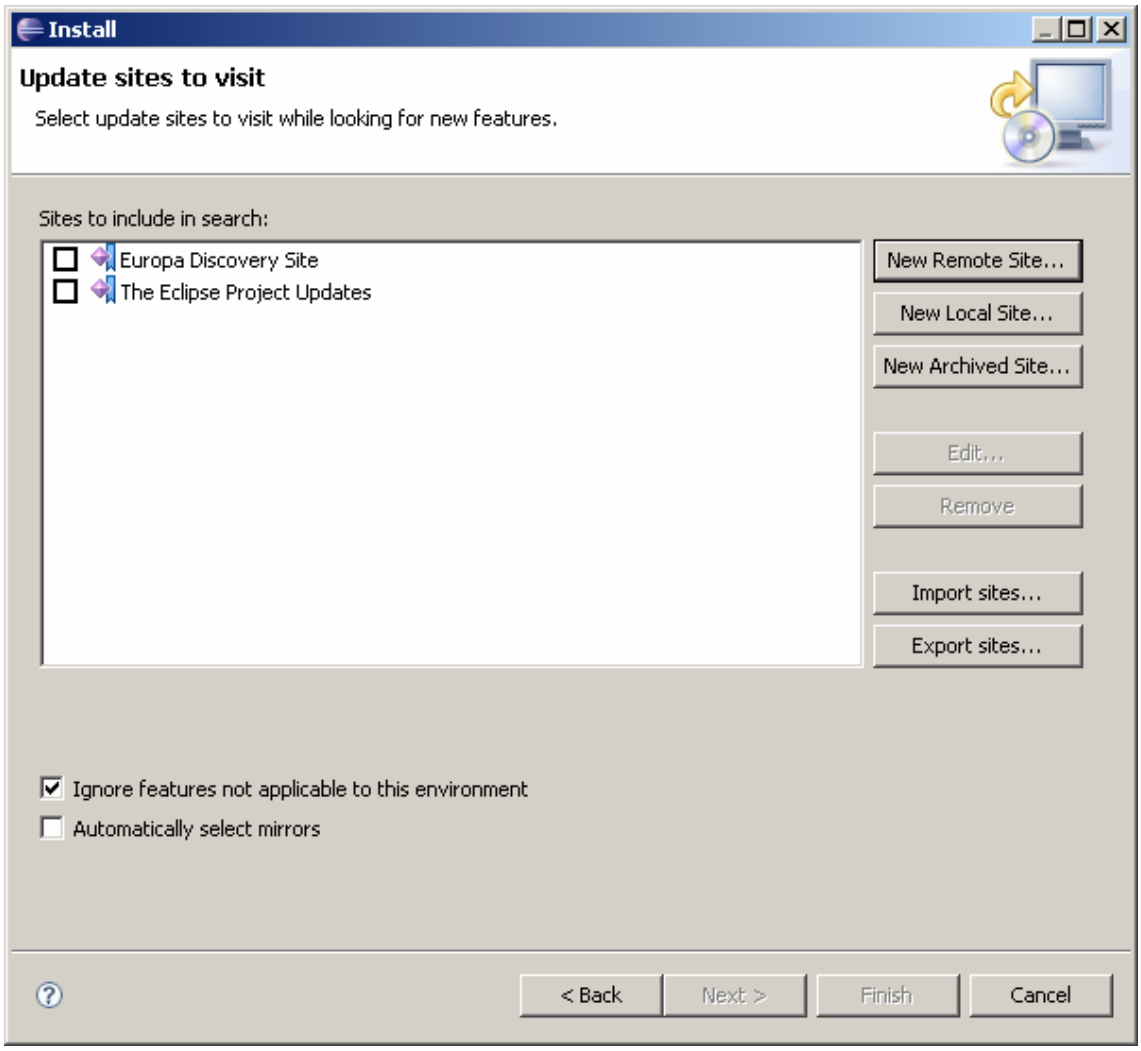
When you execute Eclipse and you want to install any Eclipse plug-in, in this case NXJ Plug-in, you have to go to *help > software updates > find and install:*



Then you will see the following assistant. Select the second option: "Search for new features to install"

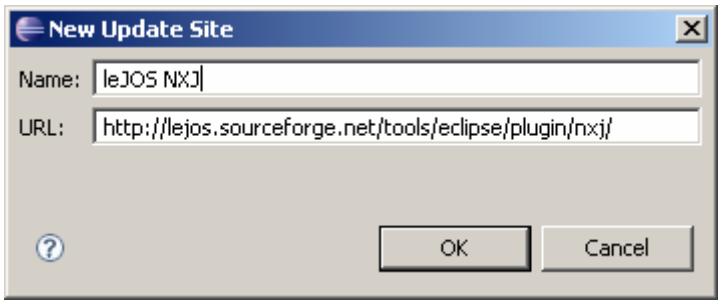


Click in the button next to indicate where is NXT Plug-in. Click in New remote Site:

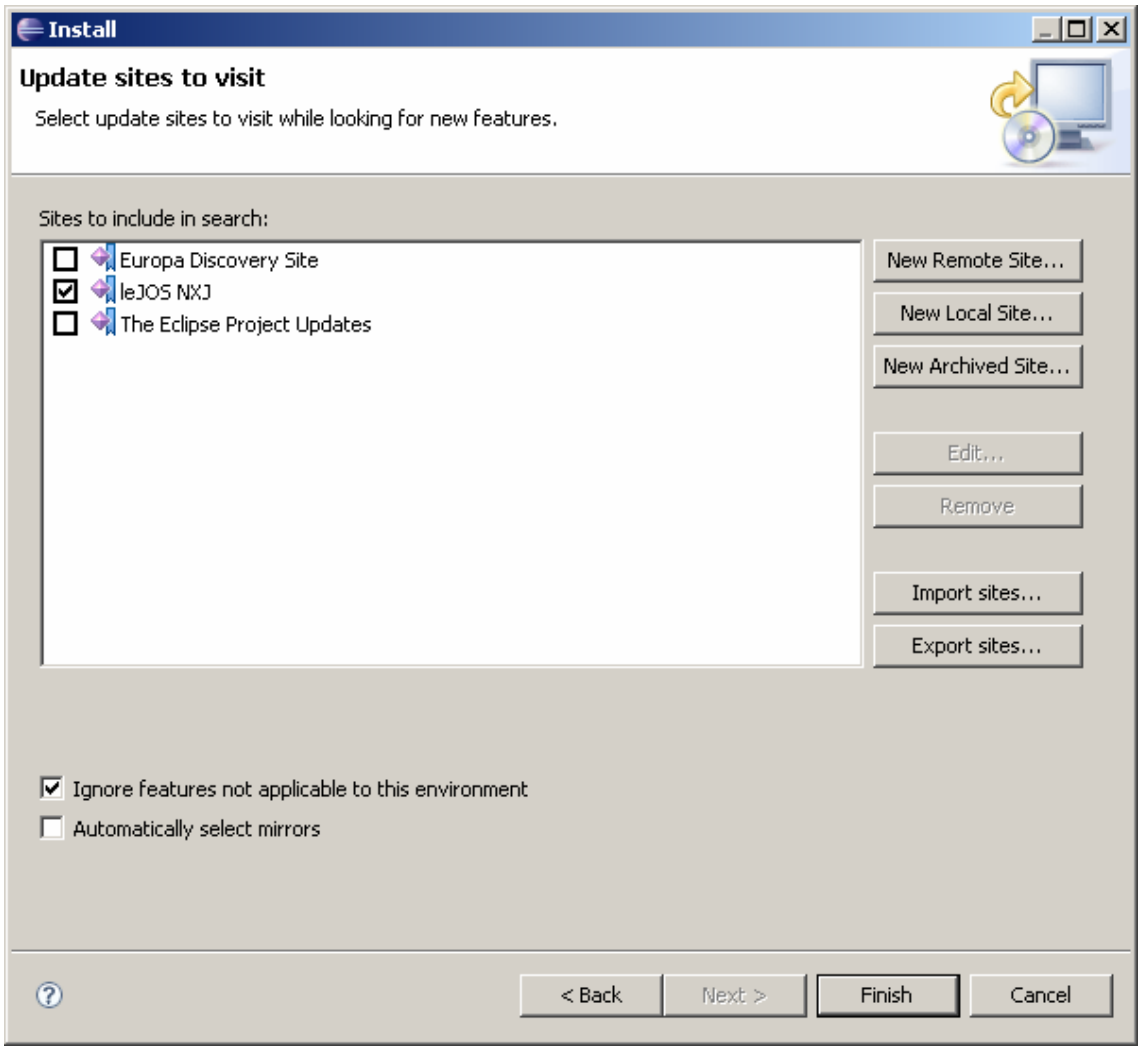


The parameters to write in the next window are:

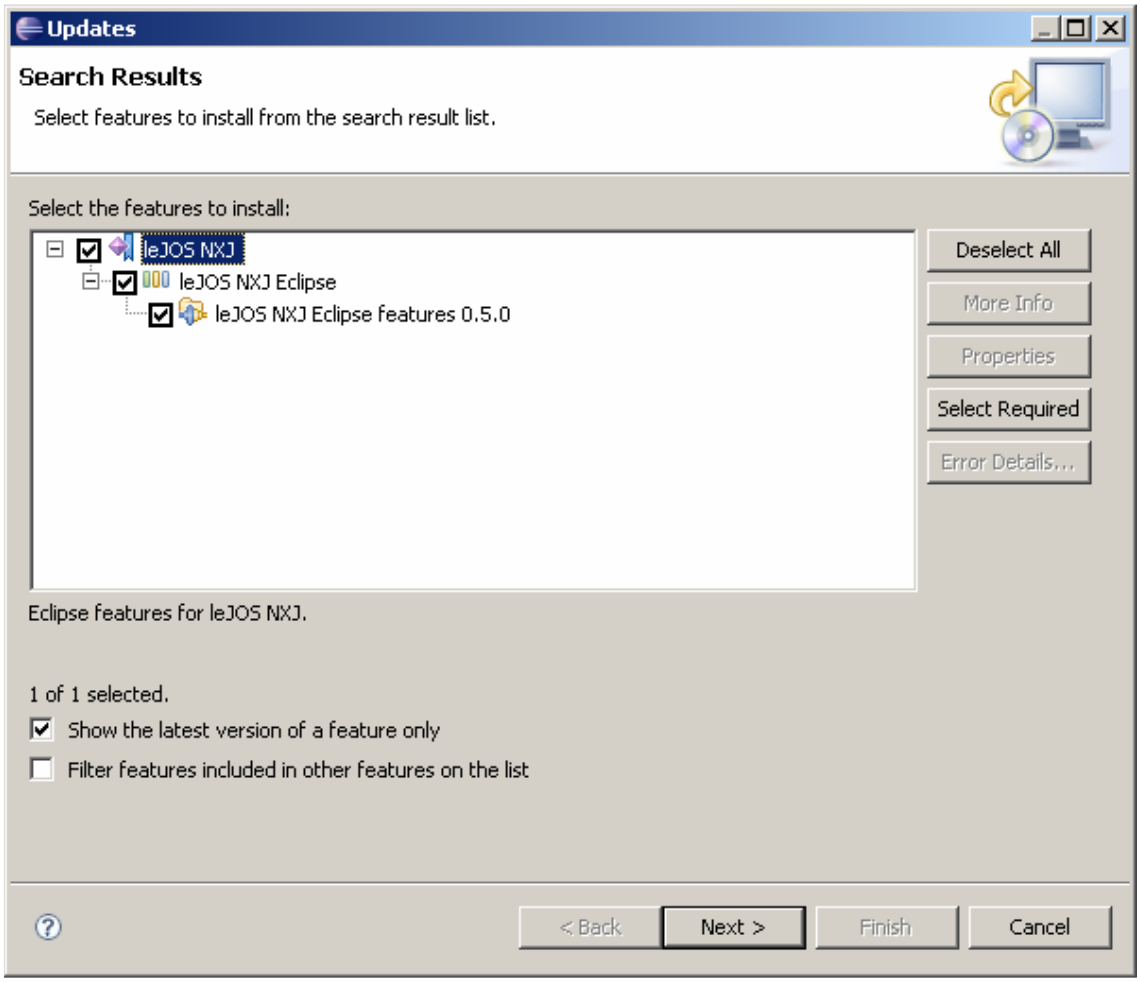
Name: leJOS NXJ
URL: http://lejos.sourceforge.net/tools/eclipse/plugin/nxj/

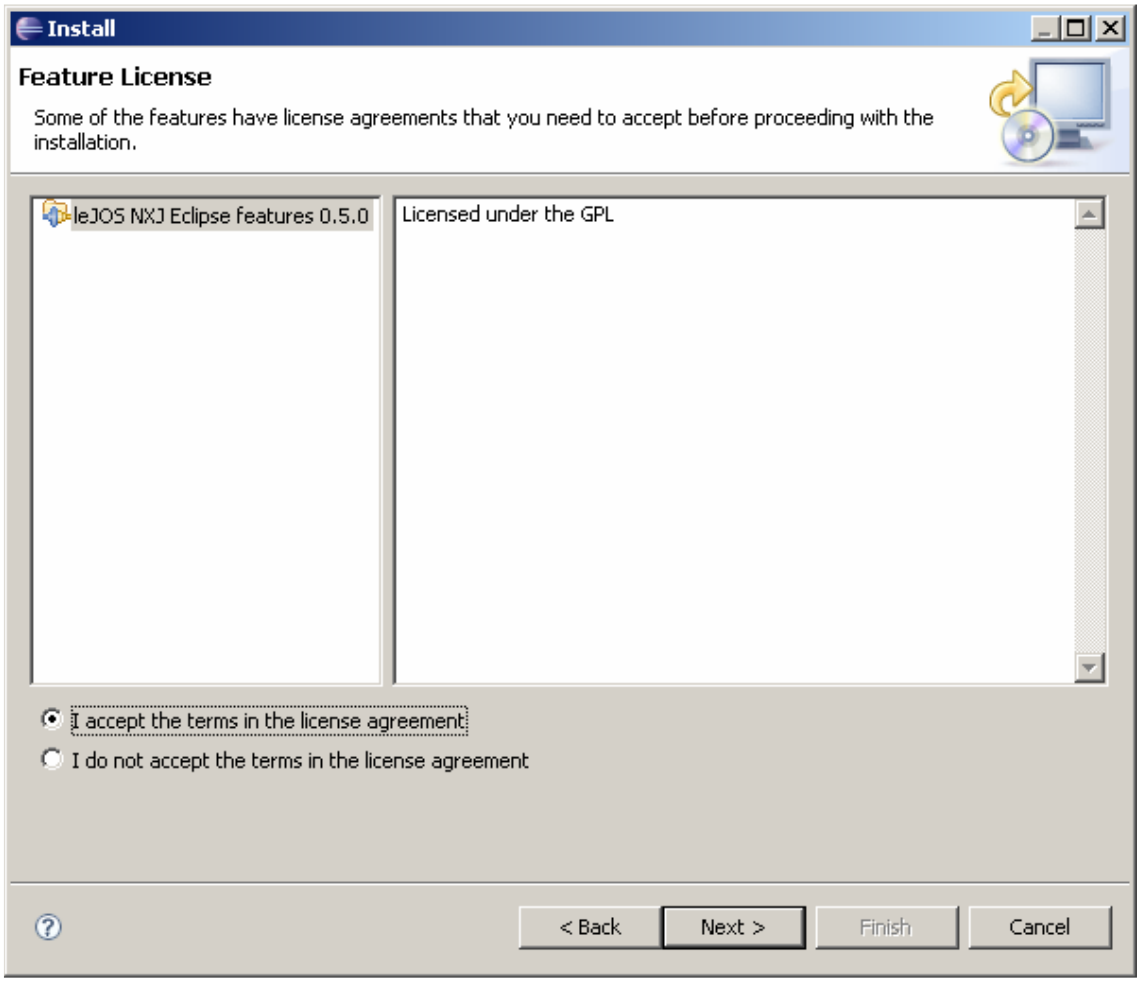


Then, select the site leJOS NXJ:

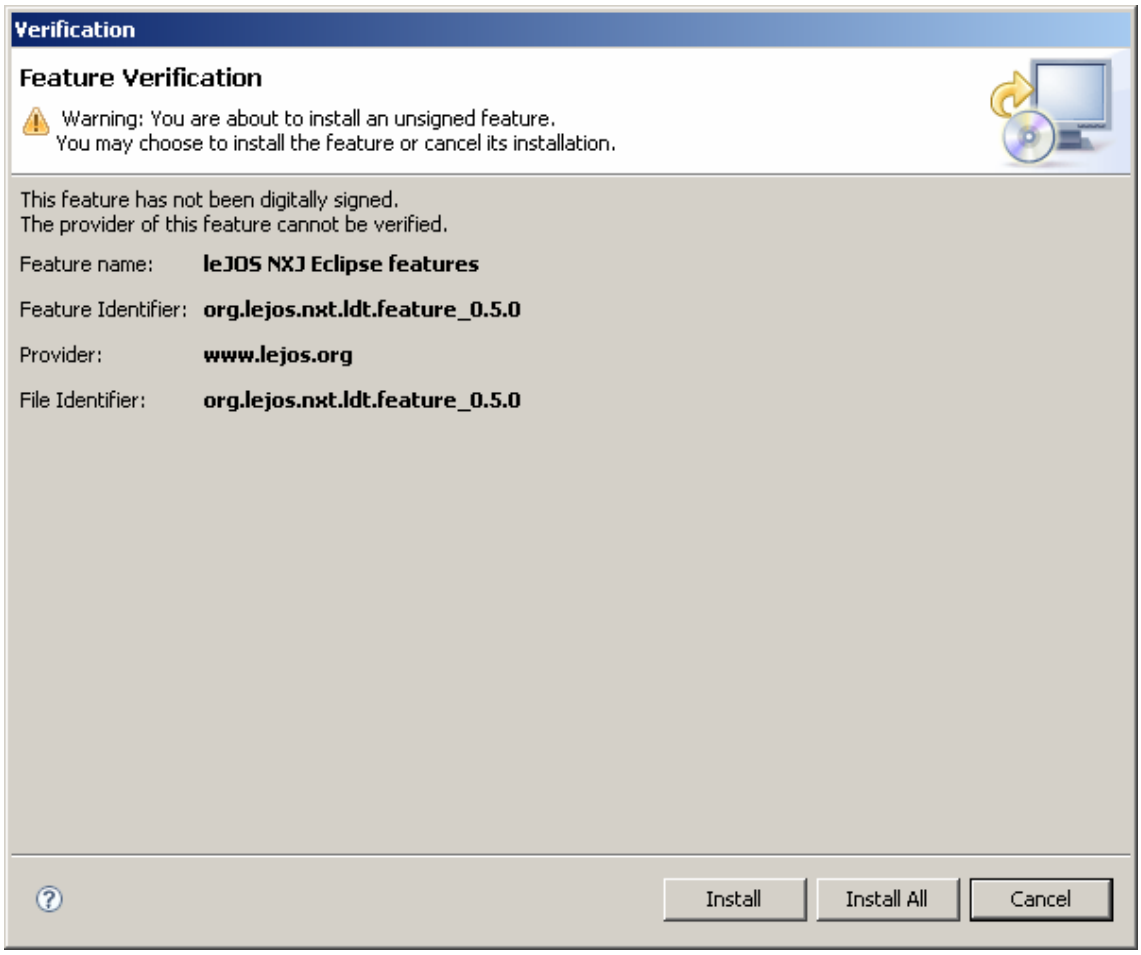


And click in Finish button. In next window, Check all options and click in next button.


Updates
Search Results
Select features to install from the search result list.
Select the features to install:
leJOS NXJ
leJOS NXJ Eclipse
leJOS NXJ Eclipse features 0.5.0
Deselect All
More Info
Properties
Select Required
Error Details...
Eclipse features for leJOS NXJ.
1 of 1 selected.
Show the latest version of a feature only
Filter features included in other features on the list
< Back
Next >
Finish
Cancel



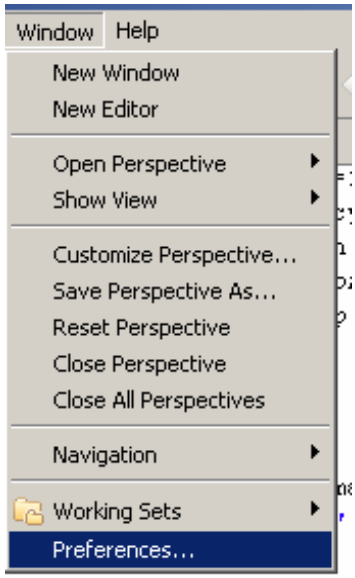
Then you have to accept the terms in the licence agreement and click in next button to install the NXJ plug-in.



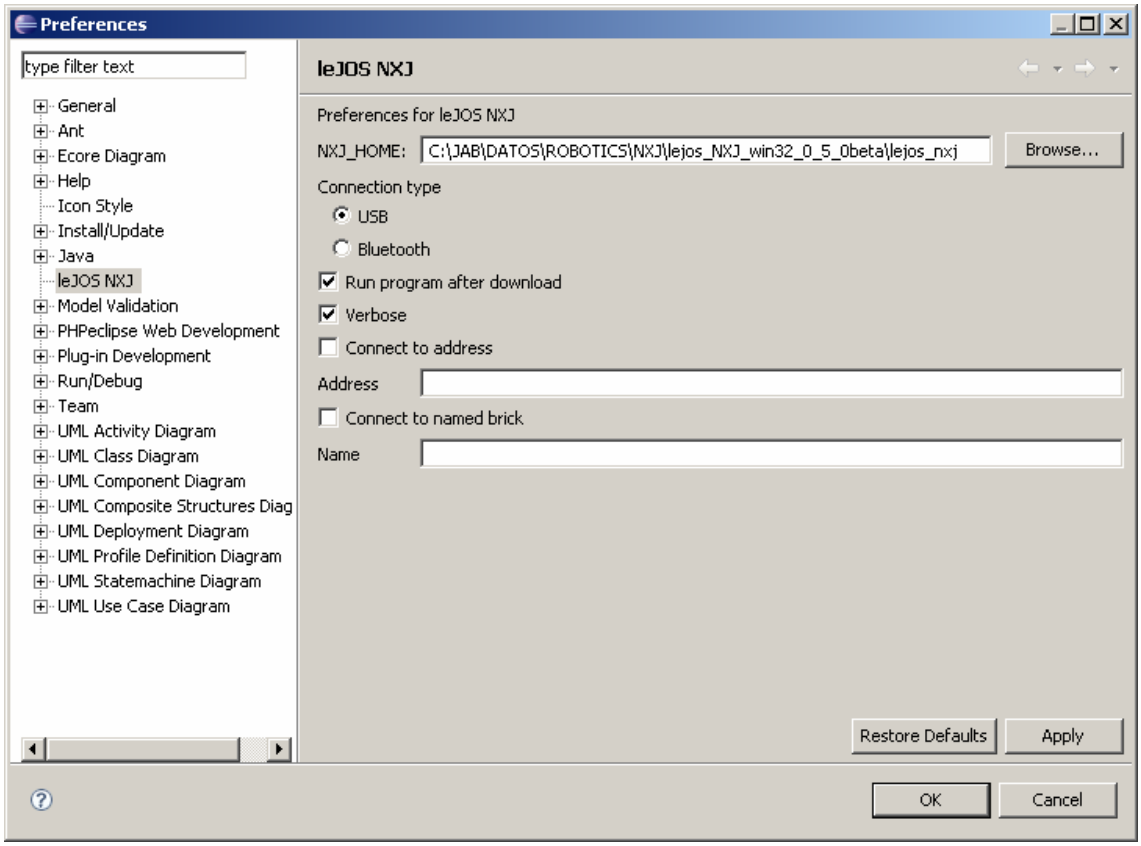
If you have reached to this window, you have finished installing the NXJ plug-in.

2.3.2.3.- Configure Eclipse IDE to use NXJ Plug-in

Once you have installed NXJ Plug-in, you have to set NXJ_HOME variable in preference area in Eclipse.



NXJ_HOME = C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj

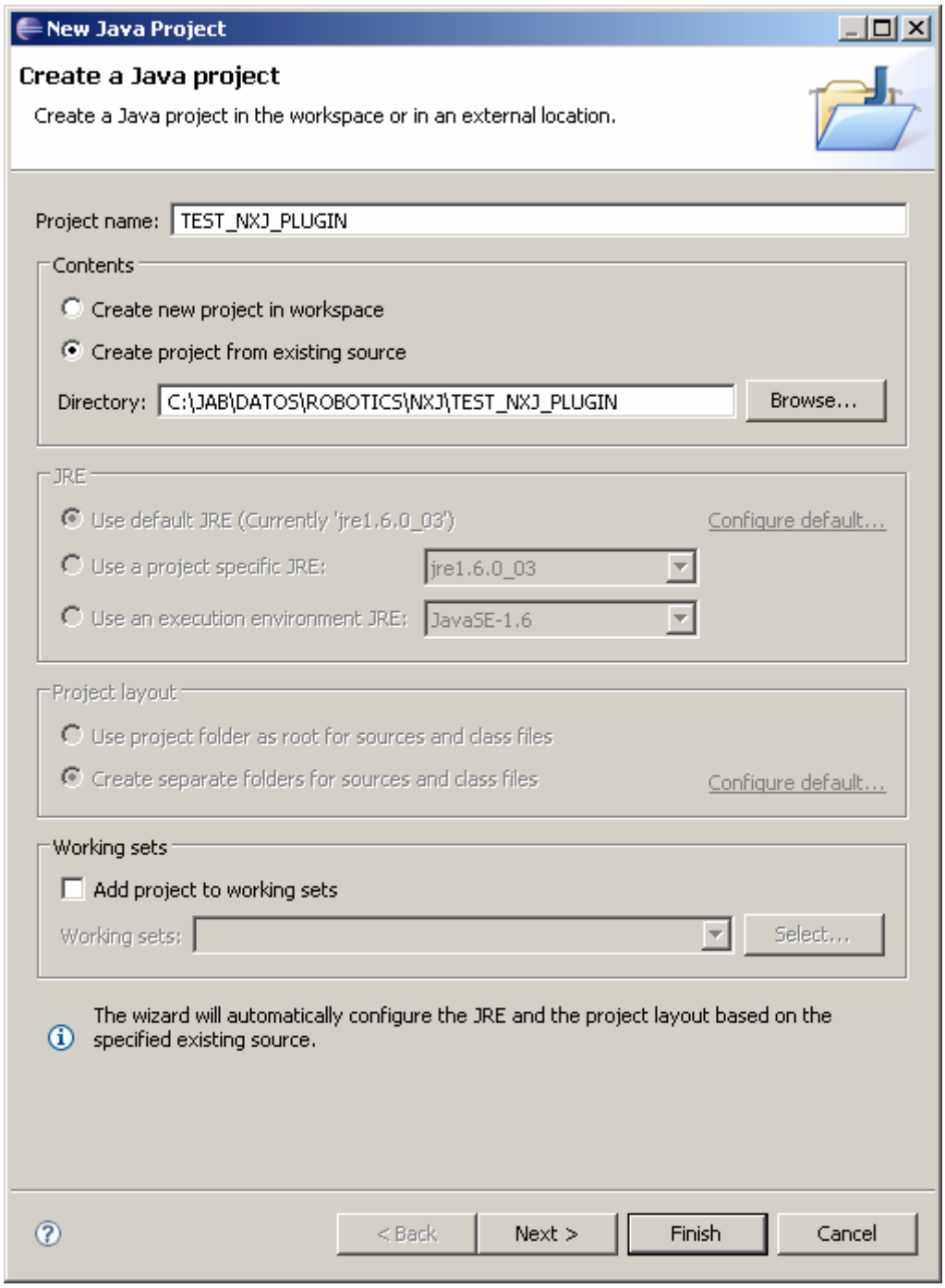


2.3.2.4.- Creating a NXJ Project in Eclipse IDE

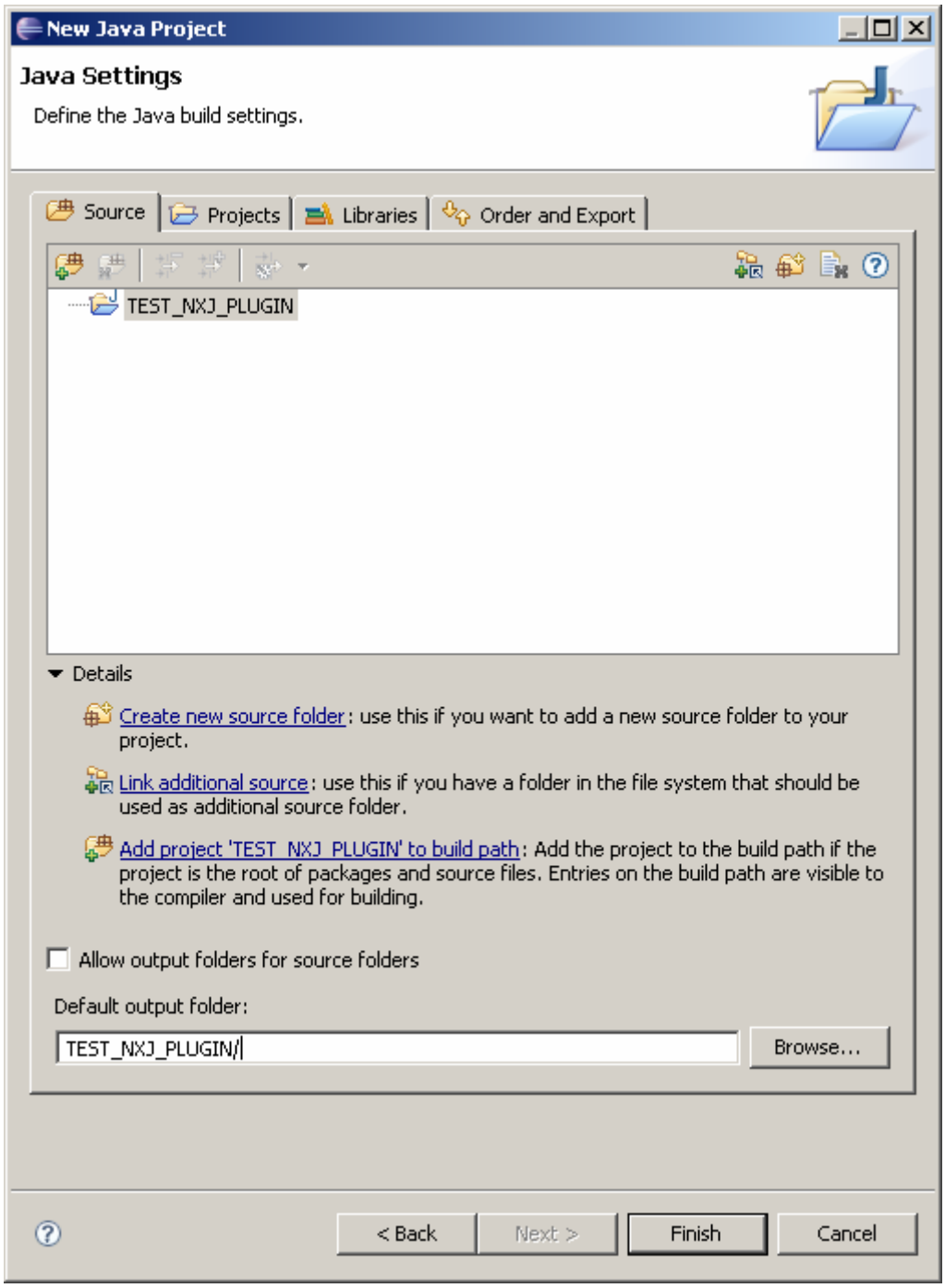
2.3.2.4.1.- Creating a Java Project

In Eclipse, you can create a Java Project. To create a Java project, go to *File* > *New* > *Java Project*:

Then you will see an assistant where you have to indicate the name of the Java Project and where you want to store your Java Classes and Byte codes:

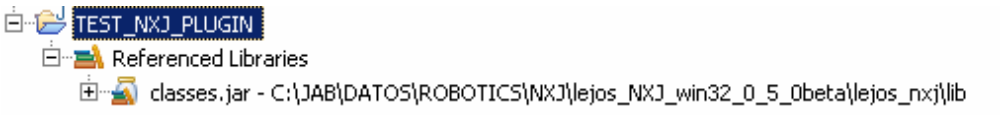


Click in next button. In next window, you have to indicate where you want to store .class files. You have to indicate that class files have to store in the same path where you have java files.



Then you will see in tree menu in Eclipse IDE a new Java Project named TEST_NXJ_PLUGIN. Select it and click in "Convert to leJOS NXJ project"

Then your project will be associated with lib path in your NXJ release.

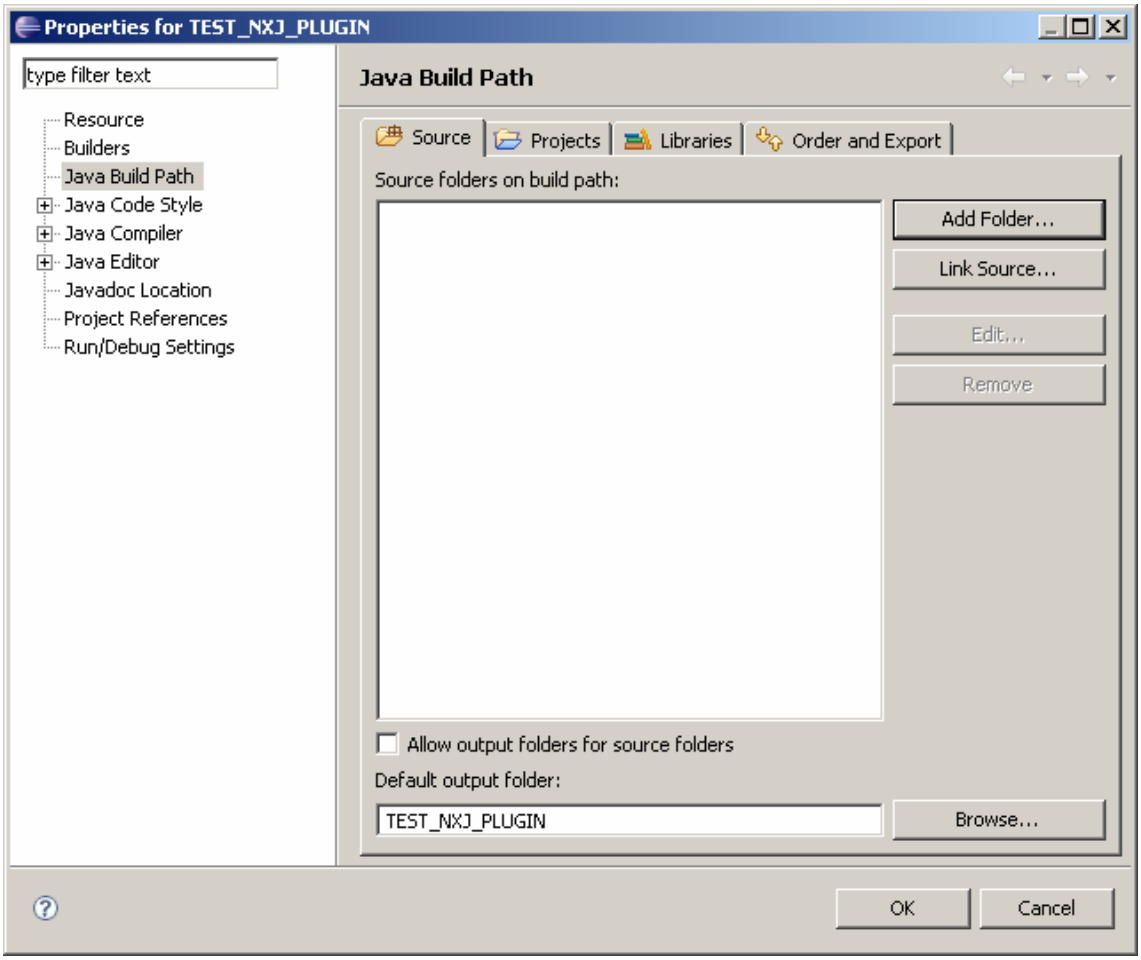


Besides, in Eclipse console window, you will see the following message:

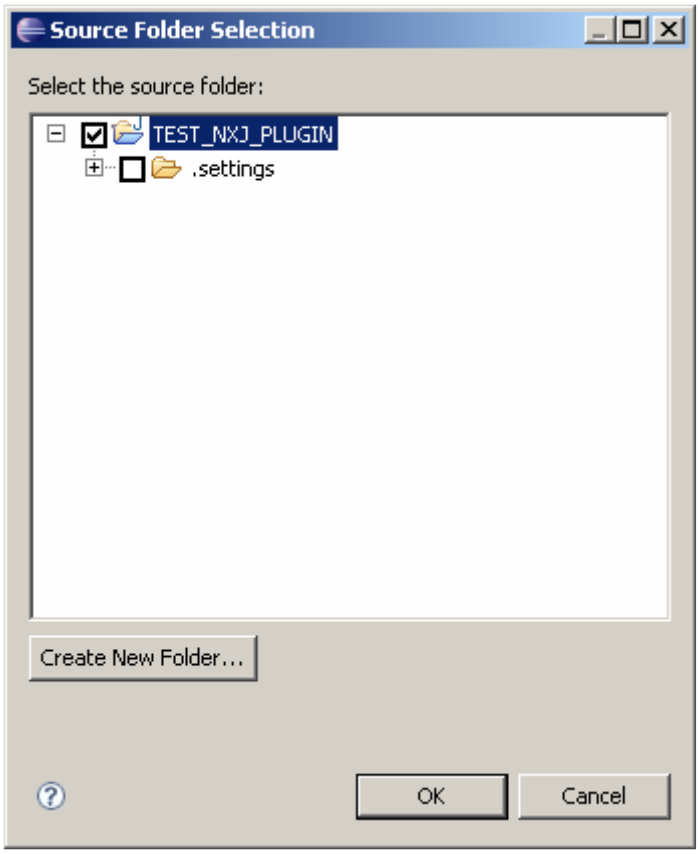
Project TESTING_NXJ_PLUGIN now is a leJOS NXJ project

Then, it is necessary to define Build path. Select your NXJ Project and click in Configure Build Path in context menu:

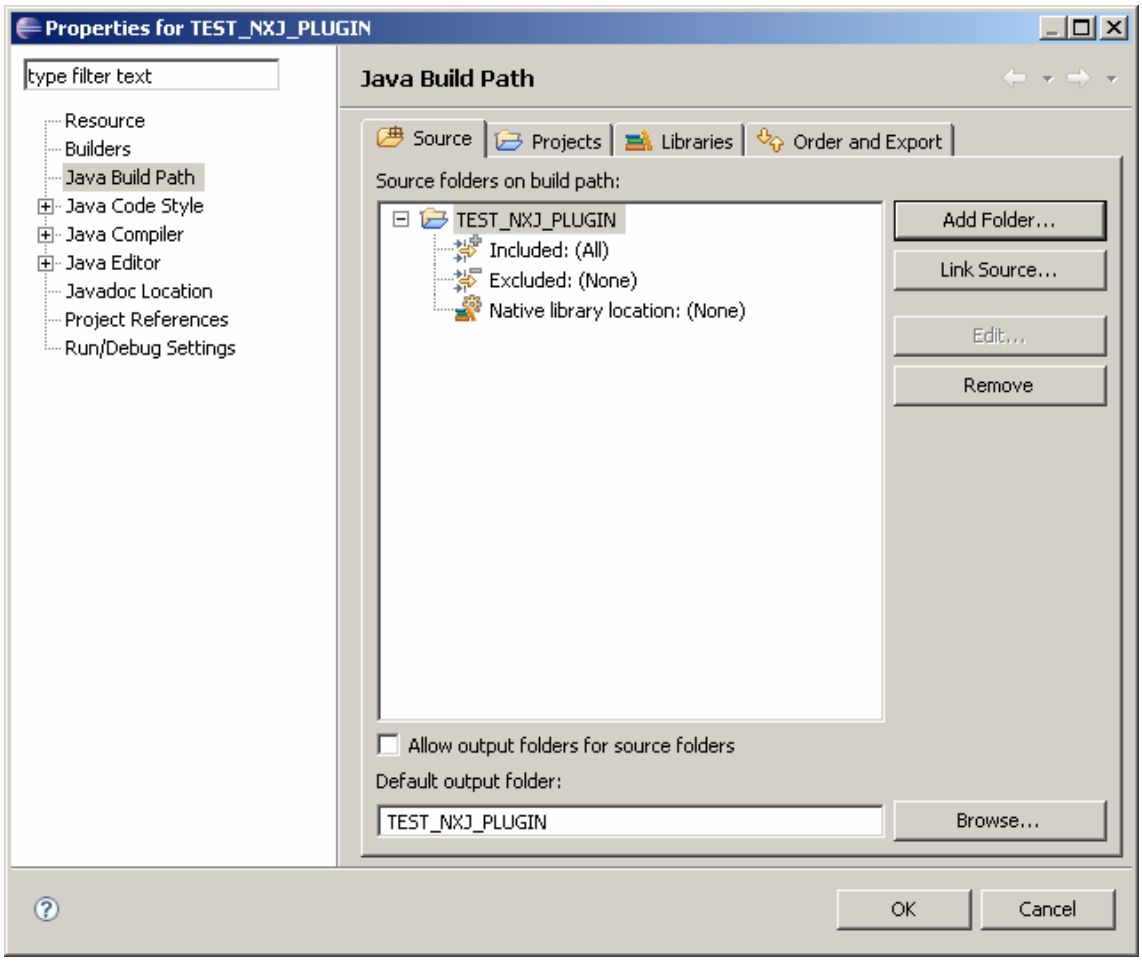
Add new folder:



And check the folder TEST_NXJ_PLUGIN:



Then, click in Ok button to see the changes in your project:



Then you have to click in Ok button to finish this step.

2.3.2.5.- NXJ Plug-in Features

2.3.2.5.1.- Introduction

NXJ Plug-in allow to NXJ developers to do the following actions:

1. Upload latest firmware into NXT Brick
2. Upload NXJ programs into your NXT Brick
3. Convert any Java project into NXJ project

In this section, we explain the first and second feature.

Besides, NXJ Plugin includes a excellent documentation.

2.3.2.5.2.- Upload firmware

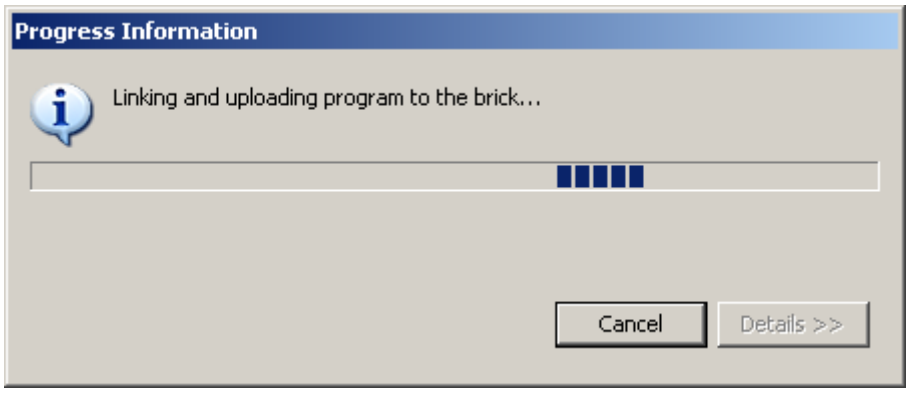
If you have to upload leJOS firmware to NXT brick, you can use this plug-in. Connect your NXT brick by USB wire to your computer and click in *Upload firmware*:

2.3.2.5.3.- Upload Program to the NXT brick

Normally you send programs to NXT brick using DOS console or using Eclipse manually or anoher IDE. With NXJ Plug-in the process to send NXJ programs is so easy.

When you finish developing your NXJ Program, simply selecting the class and with context menu, selecting the option *Upload Program to the NXT Brick*:

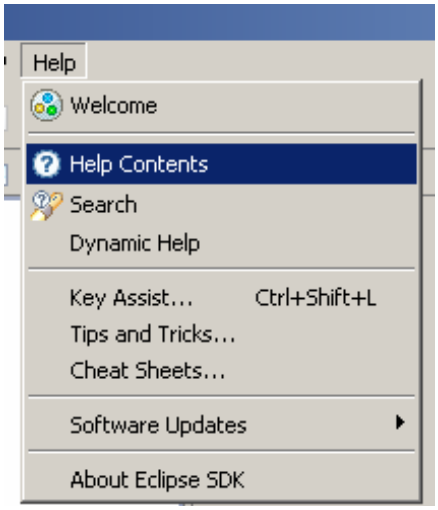
Then if your class doesn't have any syntactic error, connect your NXT brick and click in the option, then you will see the following window:



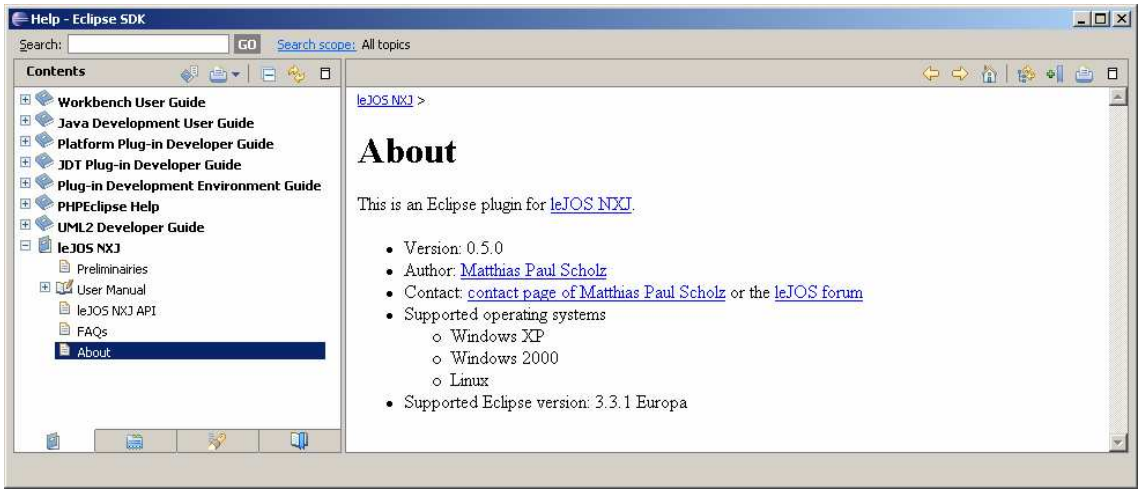
Then your program will be running in your brick!

2.3.2.5.4.- NXJ Plug-in documentation

The plug-in includes an excellent documentation integrated with eclipse. To read it, click in *Help > Help Contents*:

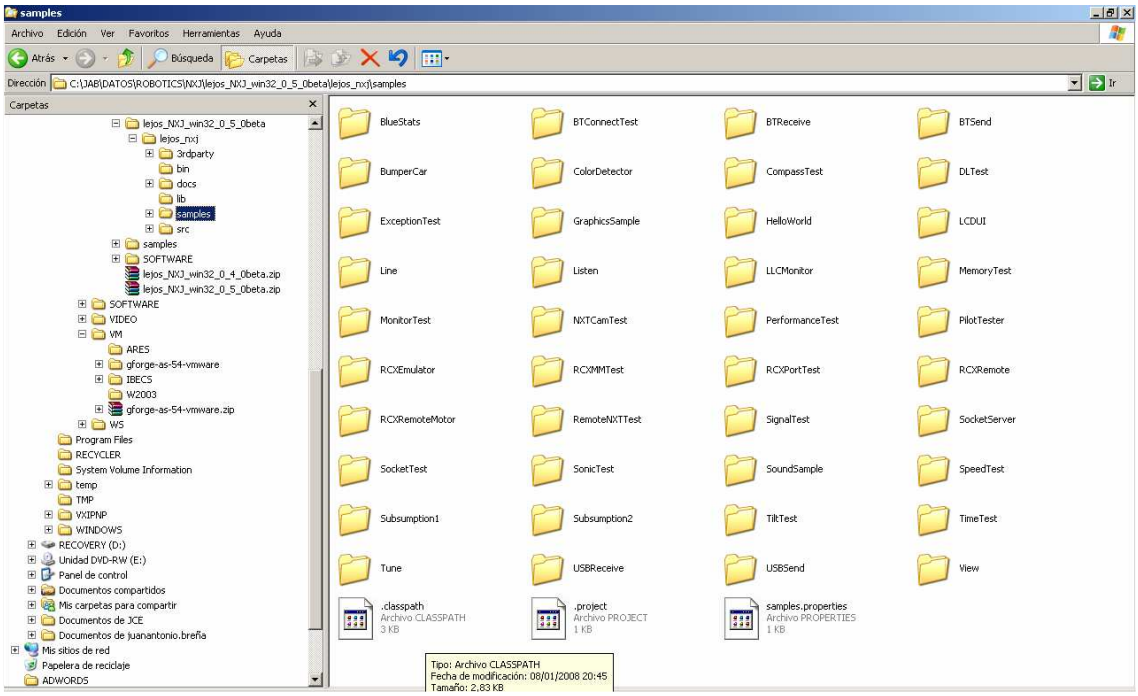


Then you will see a new section named leJOS NXJ.



2.4.- Developing your first program with NXJ

When you download your NXJ, it includes a set of NXJ examples. If you see the file tree, you will notice that you can experiment with NXJ several hours.



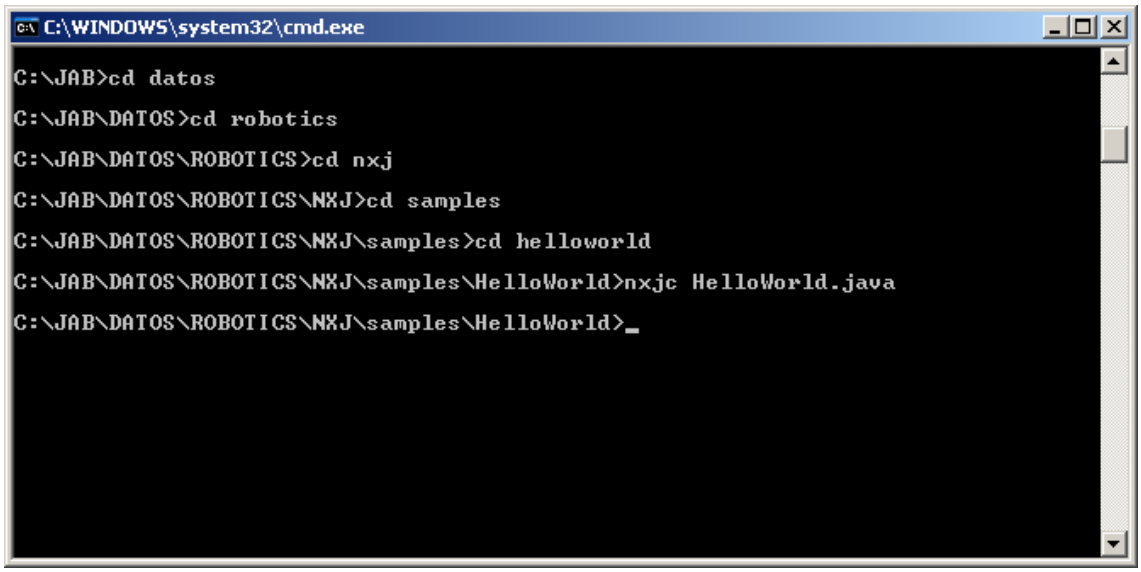
A good beginning is execute a very basic Example, HelloWorld.java. Open the file HelloWorld.java in your favourite editor or Java IDE to study the structure of any Java program

```
import lejos.nxt.*;

public class HelloWorld
{
  public static void main (String[] aArg)
  throws Exception
  {
     LCD.drawString("Hello World",3,4);
     Thread.sleep(2000);
```

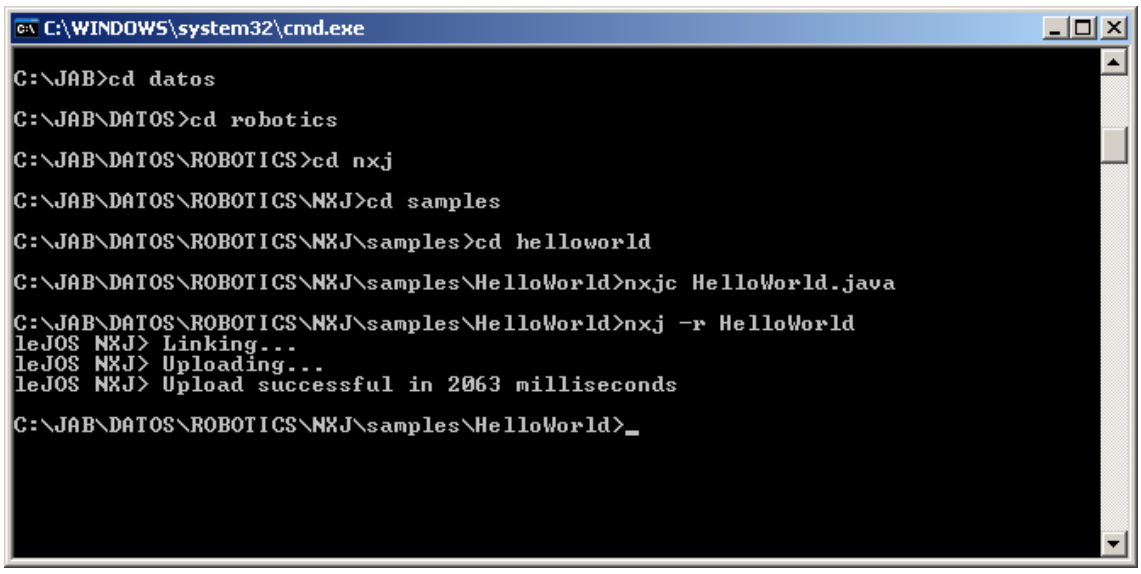
```
  }
}
```

To execute this example in your NXT brick:

Step1: Compile your NXJ Program
Open console windows and compile your program. To compile any NXJ program use the command nxjc <program>.java



Step2: Execute your program into your NXT Brick
With NXJ technology if you want to send a compiled program to your NXT brick use the command nxj <program>



Once you have seen the development cycle in NXJ world we are going to learn several features about this technology.

2.5.- NXJ Packages

NXJ has the following packages:

Package	Description
java.awt	Minimal AWT package for Rectangle class
java.io	Input/Output support
java.lang	Core Java classes
java.util	Utilities
javax.microedition.io	J2ME I/O
javax.microedition.lcdui	J2ME LCD User Interface classes.
lejos.navigation	Navigation classes
lejos.nxt	Access to NXT sensors, motors, etc.
lejos.nxt.comm	NXT communication classes
lejos.nxt.remote	Remote NXT access over Bluetooth
lejos.rcxcomm	Emulation of RCX communication classes
lejos.subsumption	Support for Subsumption architecture
lejos.util	More utility classes

NXJ API in Javadoc format here:
http://leJOS .sourceforge.net/p_technologies/nxt/nxj/api/index.html

2.6.- Basic concepts

2.6.1.- Introduction

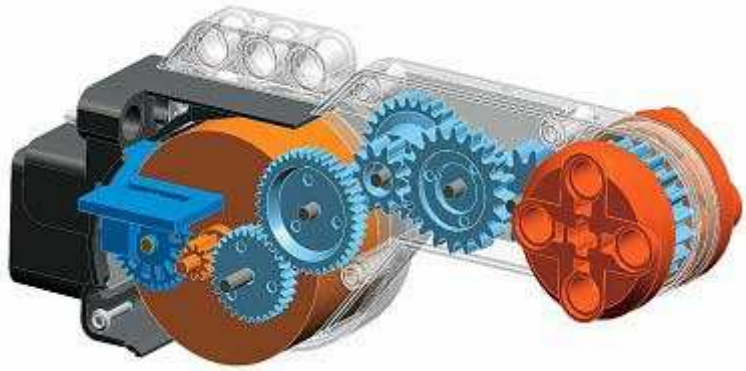
Develop robots with NXT technology is very easy. NXT brick has 3 motor ports and 4 sensor ports. In this section of this eBook, we will discover how to build a basic robots with basic sensors and motors.

2.6.2.- How to develop with Java and NXJ

2.6.3.- Using Motors in your robot

If you need to build a mobile robot with NXT, you can use NXT motors or other motors that Lego has. If you need further information about Lego Motors, I recommend the following article:
http://www.philohome.com/motors/motorcomp.htm

In this section, we only use NXT Motor:



To use a NXT Motor we will explain the following code:

```
import lejos.nxt.*;

public class TestMotorSpecialEdition
{
  public static void main (String[] aArg)
  throws Exception
  {
    String m1 = "Motor A: ";

       Motor.A.setSpeed(100);
       Motor.A.forward();

    for(;;) {
       LCD.clear();
       LCD.drawString(m1,0,1);
       LCD.drawInt(Motor.A.getTachoCount(),9,1);
       Thread.sleep(1000);
       LCD.refresh();
    }
  }
}
```

This Java Class is a evolution of the original TestMotor.java that you can fin in any NXJ release in the folder TestMotor.

When you code any NXJ program and you have to control any motor, you always has to indicate what motor you need to program any action. In this example, you set the speed of the Motor A with the instruction:

```
Motor.A.setSpeed(100);
```

After you indicate that when the program starts then the Motor A will turn using the instruction:

```
Motor.A.forward();
```

Further information about Motor class here:
http://lejos.sourceforge.net/p_technologies/nxt/nxj/api/lejos/nxt/Motor.html

Normally if you are going to use a unique motor, the methods that you could use are:

1. forward()
2. backward()
3. rotate()
4. rotateTo()
5. setPower()
6. setSpeed()
7. smoothAcceleration()
8. stop()

If you need to use 2 NXT Motor to control the robot's navigation, we recommend you that you use leJOS navigation API.

2.6.4.- Basic Sensors

2.6.4.1.- Introduction

Not Available

2.6.4.2.- Ultrasonic Sensor

The Ultrasonic Sensor helps your LEGO® MINDSTORMS® NXTrobot judge distances and "see" where objects are! Using the NXT Brick, the Ultrasonic Sensor is able to detect an object and measure its proximity in inches or centimeters

The Ultrasonic Sensor uses the same scientific principle as bats: it measures distance by calculating the time it takes for a sound wave to hit an object and return – just like an echo.

Large sized objects with hard surfaces return the best readings. Objects made of soft fabric or that are curved [like a ball] or are very thin or small can be difficult for the sensor to detect.

To explain how to use Ultrasound Sensor with NXJ, we will explain the example SonicTest.java that you can find in the folder SonicTest

```
import lejos.nxt.*;

/**
 * Simple test of the Lego UltraSonic Sensor.
 *
 * @author Lawrie Griffiths
 *
 */
public class SonicTest {

      public static void main(String[] args) throws Exception {
            UltrasonicSensor sonic = new
UltrasonicSensor(SensorPort.S1);

            while(!Button.ESCAPE.isPressed()) {
                  LCD.clear();
                  LCD.drawString(sonic.getVersion(), 0, 0);
                  LCD.drawString(sonic.getProductID(), 0, 1);
                  LCD.drawString(sonic.getSensorType(), 0, 2);
                  Thread.sleep(200);
                  LCD.drawInt(sonic.getDistance(), 0, 3);
                  LCD.refresh();
                  Thread.sleep(500);
            }
      }
}
```

To use Ultrasound Sensor, you have to create a instance of UltrasonicSensor:

```
UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);
```

You have to indicate in what port you have pluged the sensor. In the example Ultrasound is pluged into port 1.

If your program need to know the distance from the sensor to any object then you need to use the method:

```
sonic.getDistance()
```

Further information about Motor class here:
http://lejos.sourceforge.net/p_technologies/nxt/nxj/api/lejos/nxt/UltrasonicSensor.html

2.6.4.3.- Compass Sensor

Not Available

2.6.5.- Create a GUI with NXJ

2.6.5.1.- Introduction

Not Available

2.6.5.2.- Packages

Not Available

2.6.5.3.- Alternatives

Not Available

2.7.- Advanced concepts

2.7.1.- Introduction

Not Available

2.7.2.- Robotic Navigation

2.7.2.1.- Introduction

Not Available

2.7.3.- Multi threading

Not Available

2.7.4.- Communication protocols

2.7.5.- PC Communications

Not Available

2.7.6.- GPS

Not Available

2.8.- Advanced Sensors

2.8.1.- Introduction

Not Available

2.8.2.- NXTCam

2.8.2.1.- Introduction

NXTCam is the new Lego Mindstorm sensor designed to add artificial vision features. NXTCam is not a usual sensor as Lego Ultrasonic sensor. Before plugging this sensor, it is necessary to train it.

The NXTCam provides following capabilities:

- Track up to 8 different colorful objects at 30 frames/second
- Configure the NXTCam using USB interface on Windows XP, Windows Vista.
- Supports two tracking modes: Object tracking and Line tracking.
- Provide real-time tracked object statistics (number of objects, color of objects, bounding box coordinates or line coordinates) through a standard NXT sensor port.
- Tracked image resolution of 88 x 144 pixels at 30 frames/second
- Perform full-resolution (176 x 144) pixels color image dumps to PC via USB port.
- Low power consumption (the entire system only draws 60 mA)
- Uses NXT compatible I2C protocol for communications.
- Supports Auto Detecting Parallel Architecture (ADPA) for NXT sensor bus. This means that NXTCam can coexist with LEGO or third party digital sensor on the same NXT port. ADPA support enables user to employ several sensors on the same port without the need of external sensor multiplexer, reducing the overall size without compromising the functionality.

Mindsensor has created a sourceforge project to de create a tool that you need to install and use to train your NXTCam. http://nxtcamview.sourceforge.net/

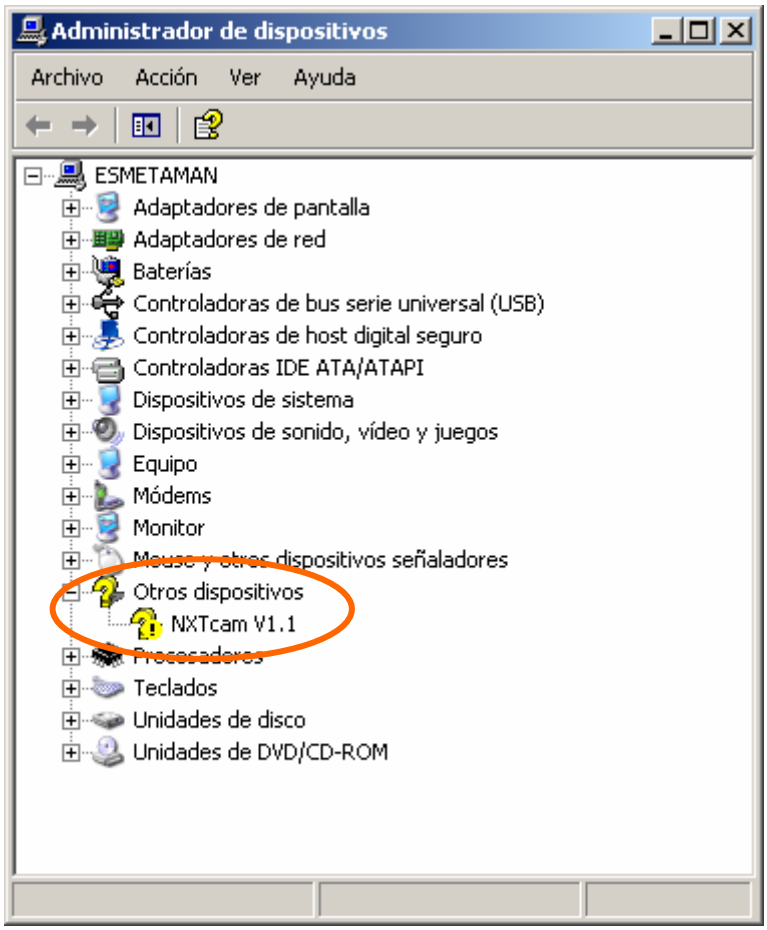
NXTCamView

In this chapter we are going to explain:

1. Install NXTCam driver
2. Install NXTCamView software
3. Learn to use NXTCamView
4. Learn the class NXTCam in NXJ
5. Learn NXTCam API
6. Learn to train NXTCam with NXTCamView
7. Detect a object with NXJ
8. Create a Radar robot with NXTCam

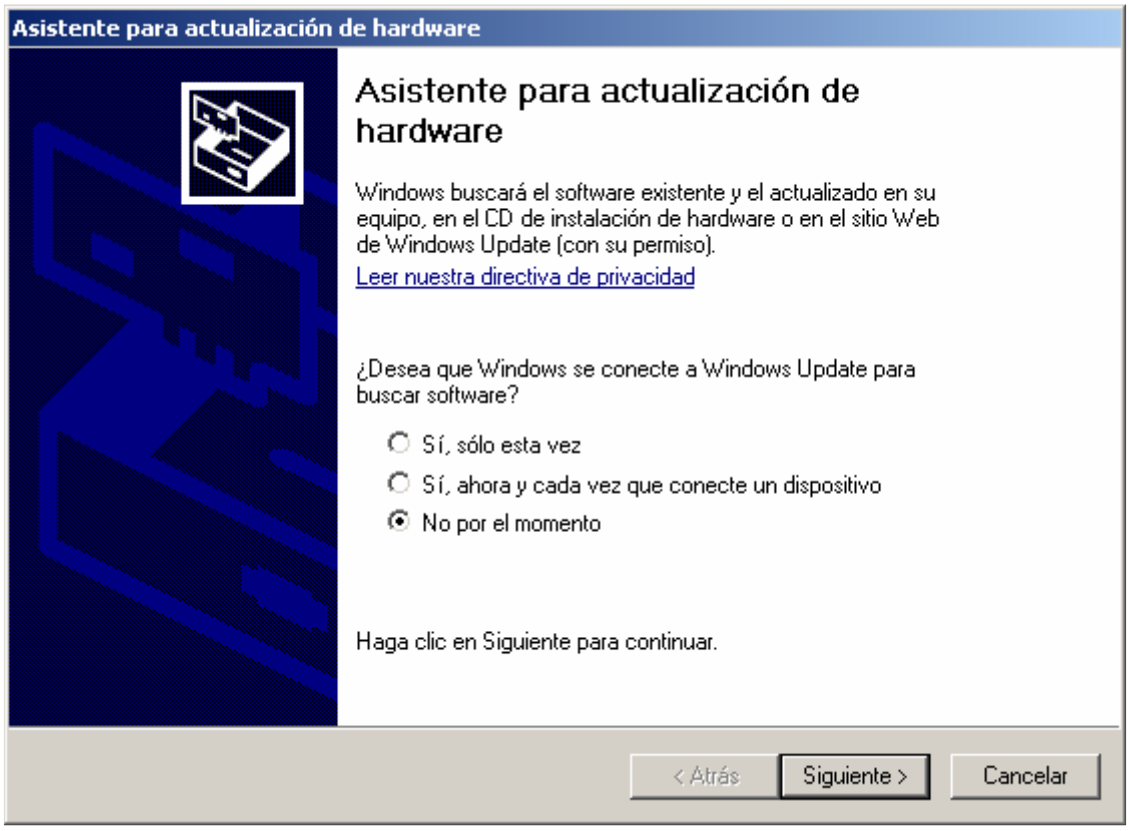
2.8.2.2.- Install NXTCam driver

The first time when you connect your NXTCam sensor with your computer, you notice in Device Manager Windows, that this USB device is not recognized by your Operating System:

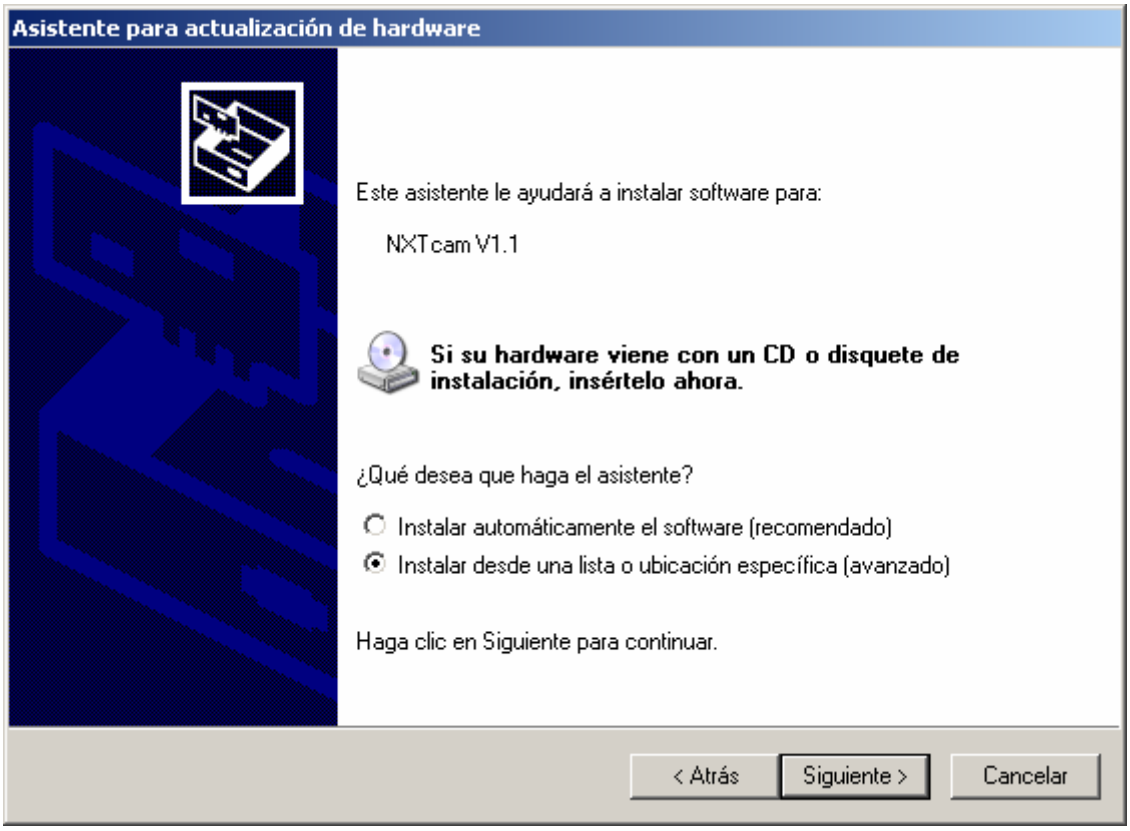


To associate this device with the correct driver, it is necessary to indicate Windows the location of NXTCam Drivers using the assistant.

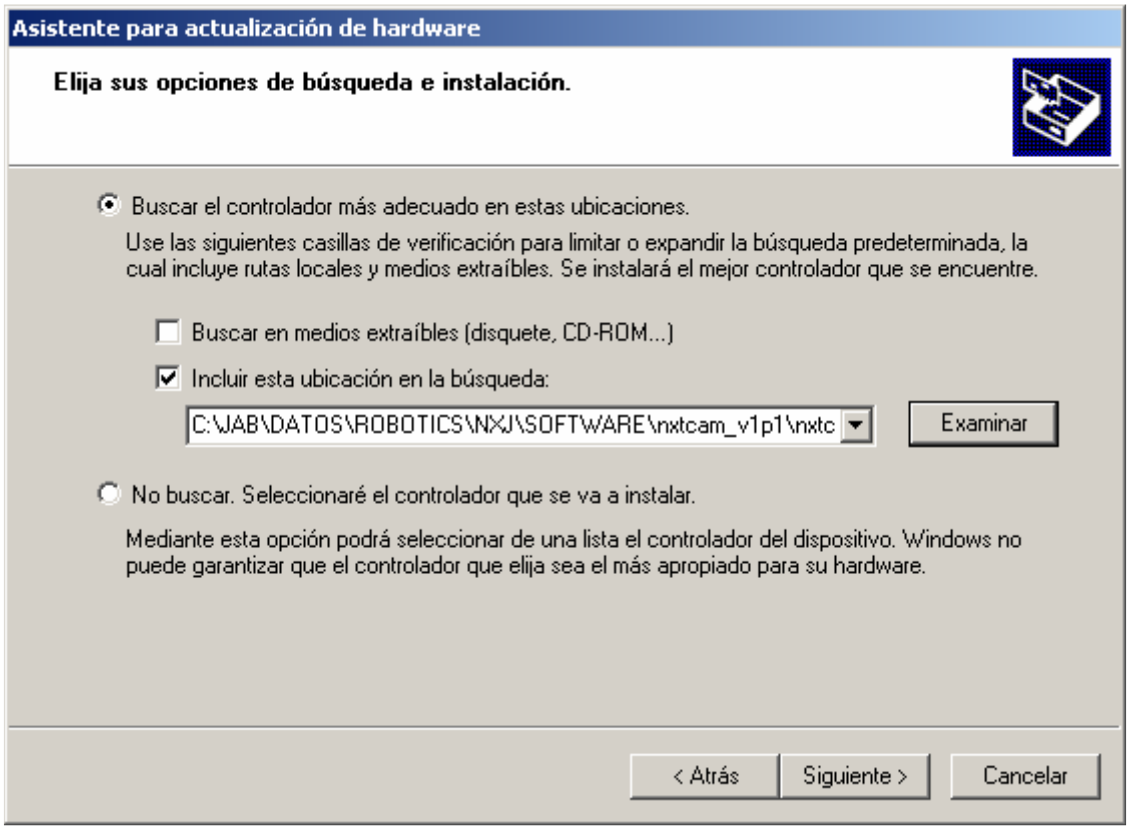
Step1: Choose the option to not use Windows Update to discover NXTCam driver. You can download the driver from Mindsensor website. Use the following URL:



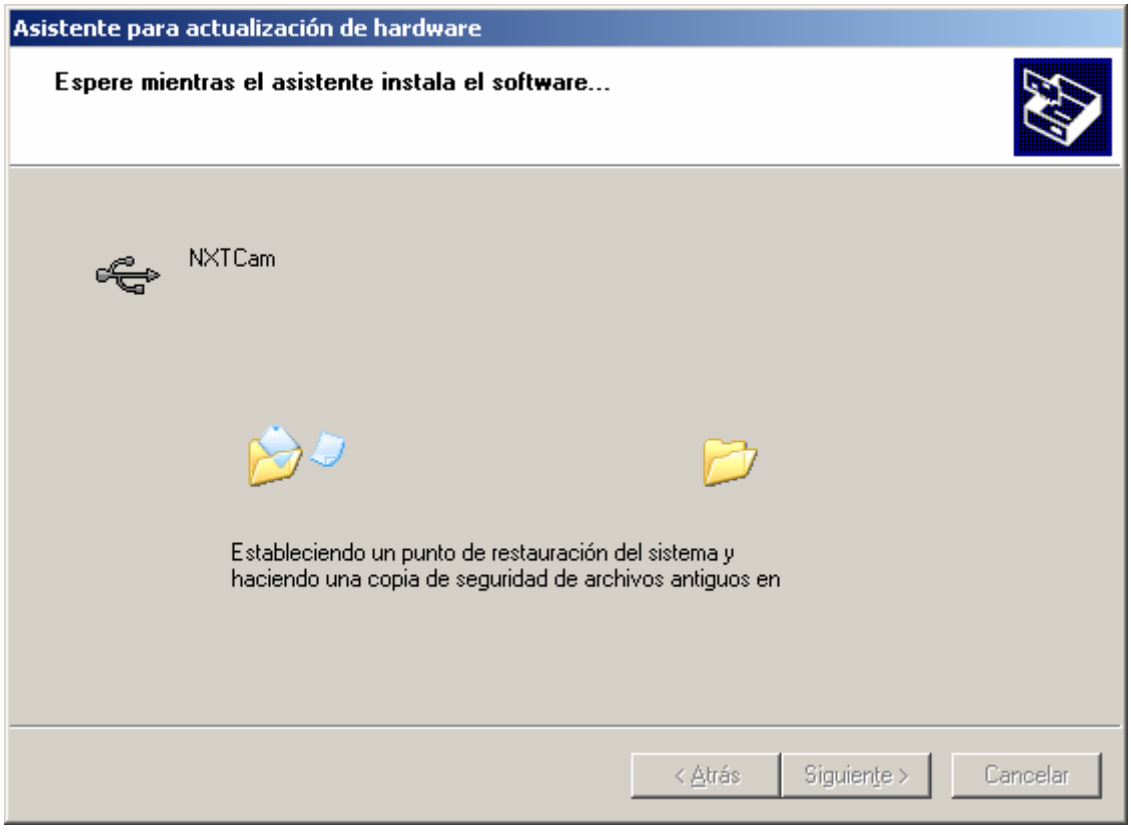
Step2: Edit the path where you have stored NXTCam driver.
Indicate that you know the location where you have stored the driver.



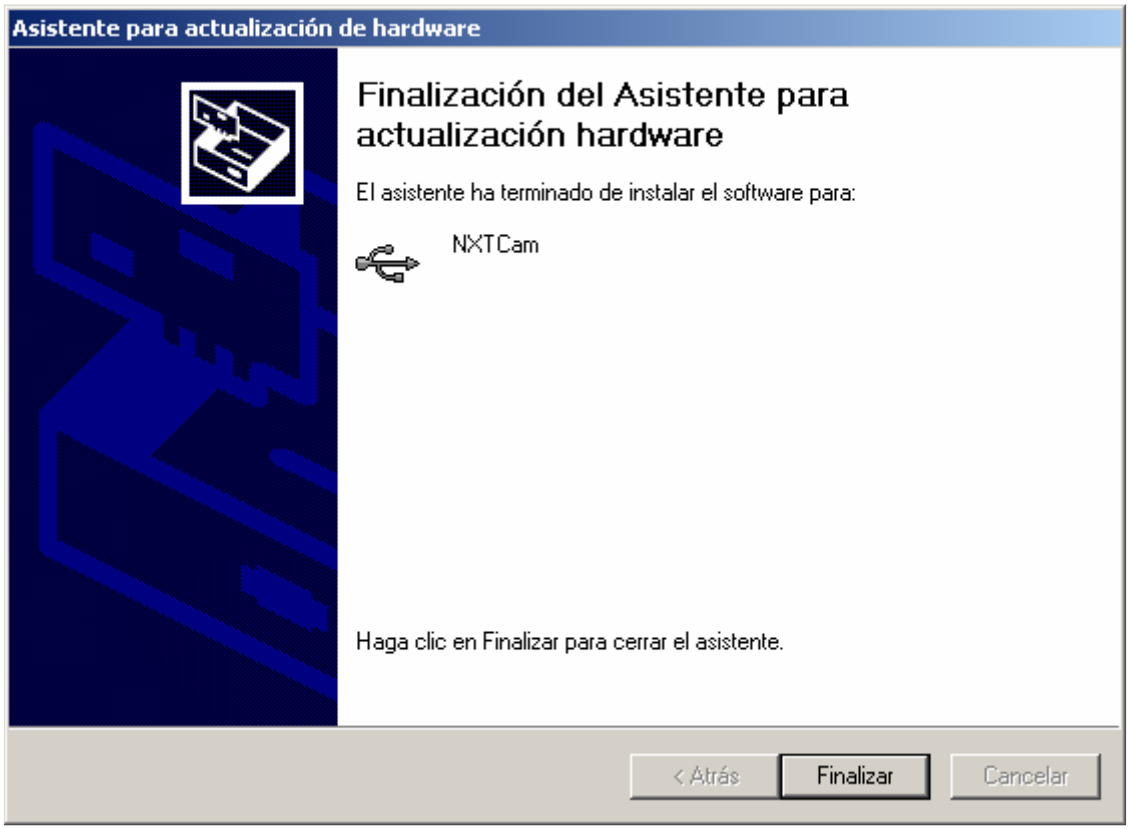
Type the path:



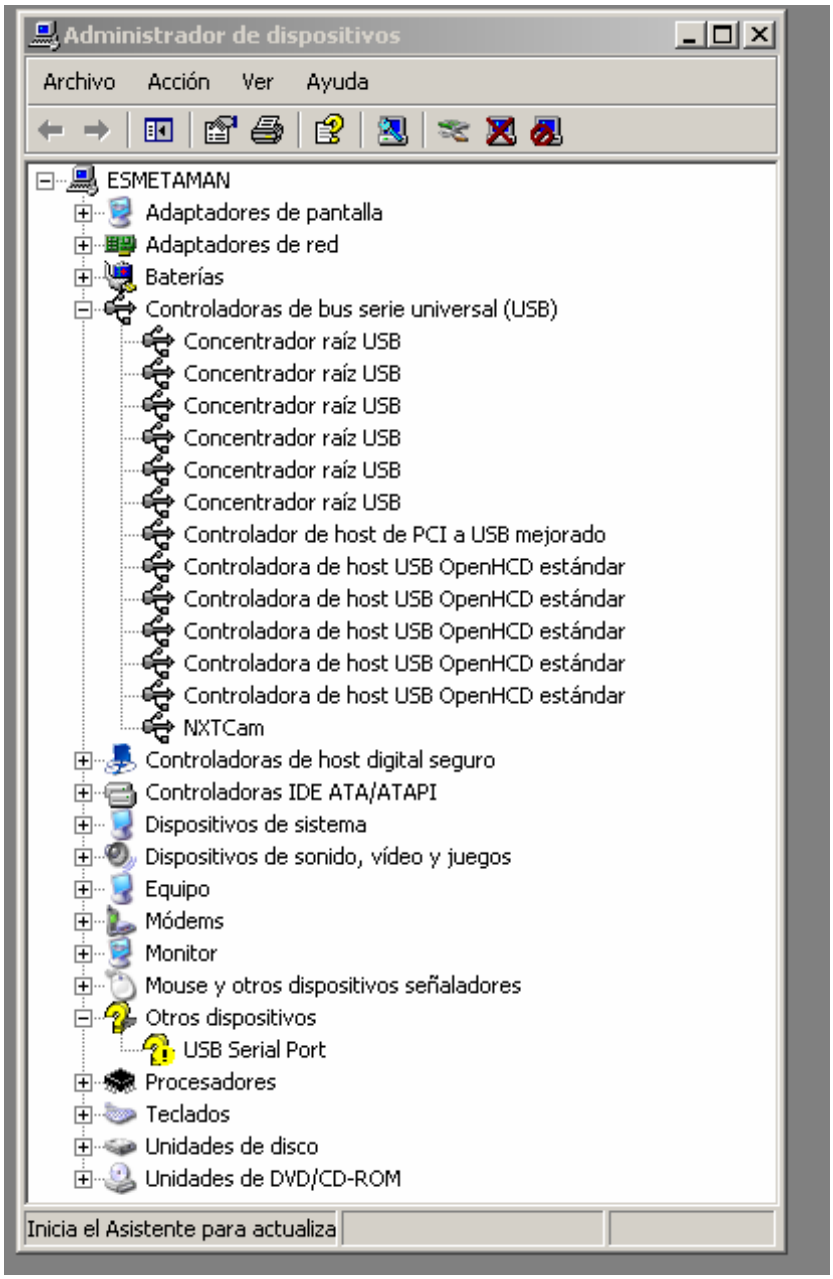
When you click in Next button, Windows will associate NXTCam with the driver.



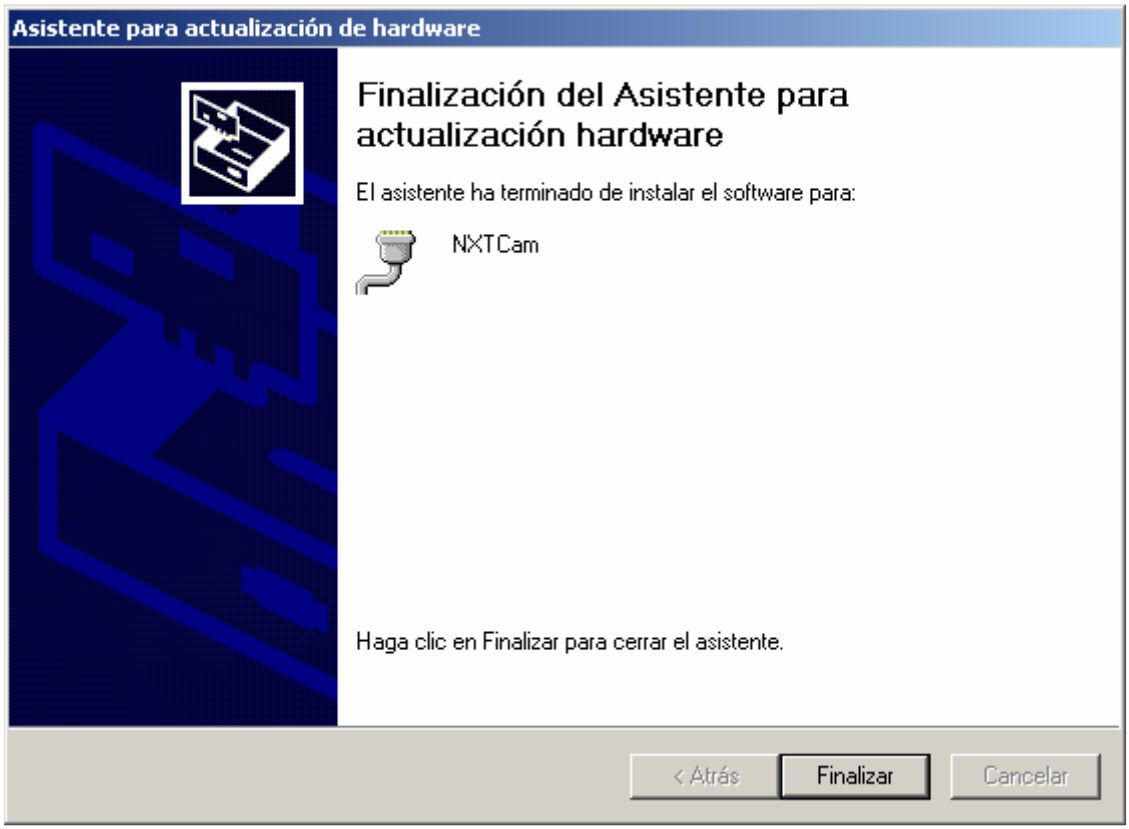
In this moment, your device has been recognized.



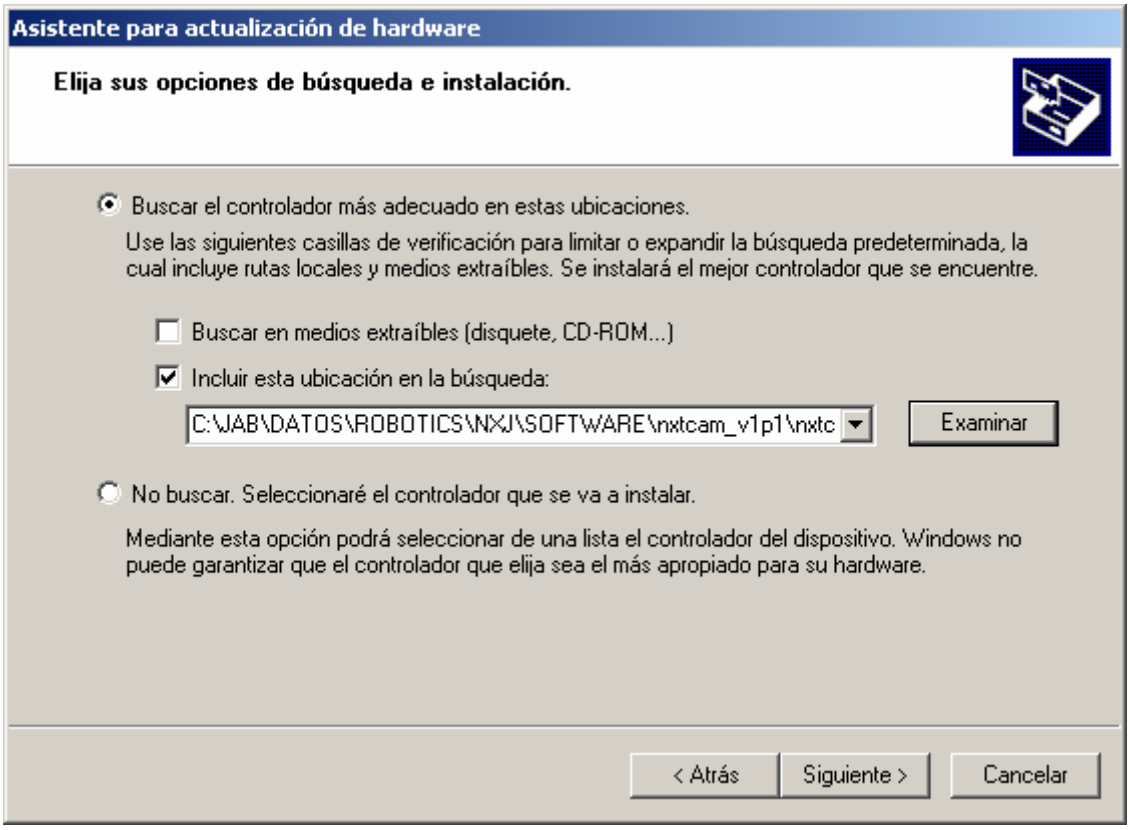
If we try to see Device Manager Window again you will see a new problem, it is necessary to install a driver for USB Serial Port, then we have to repeat the previous task twice to finish.



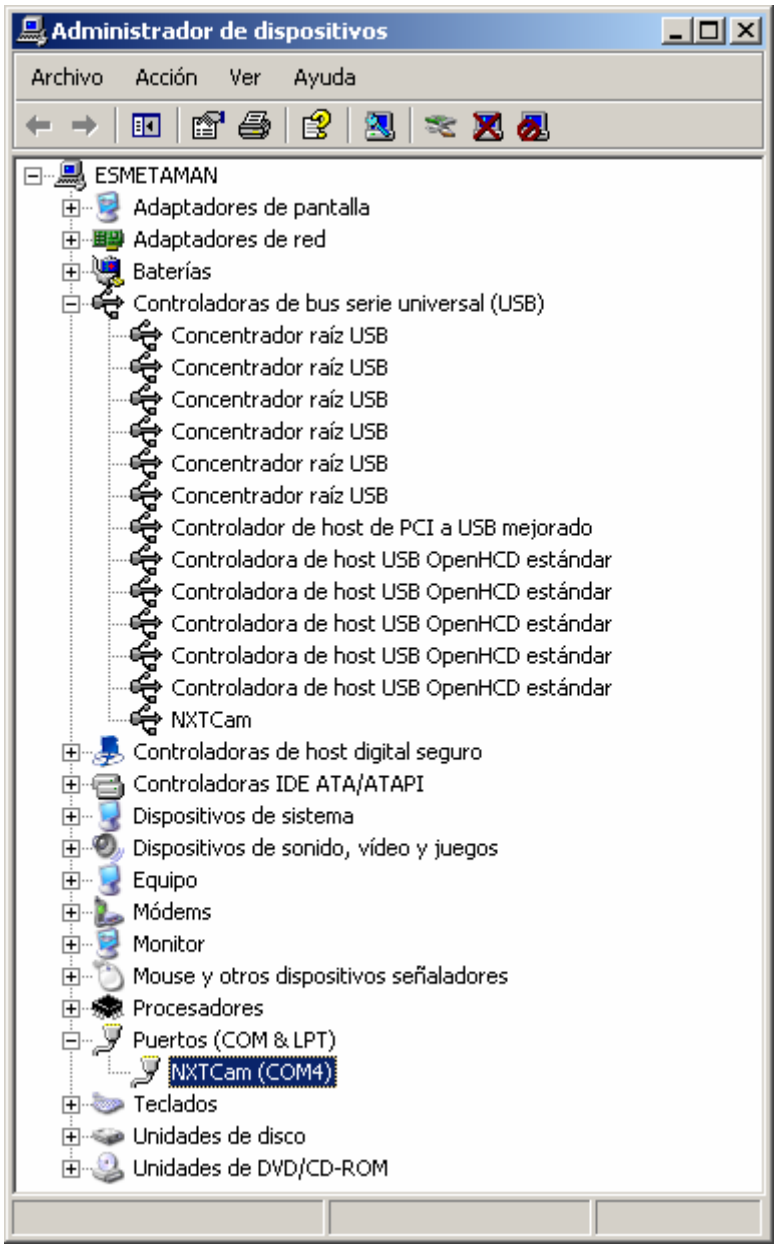
Select the node USB Serial Port in the tree menu and right-click to update the driver then follow the indications showed by the assistant:



Select the same path you use when you installed NXTCam Driver.



When you finish the process, check again your Device Manager Window, you will not see any problem with NXTCam

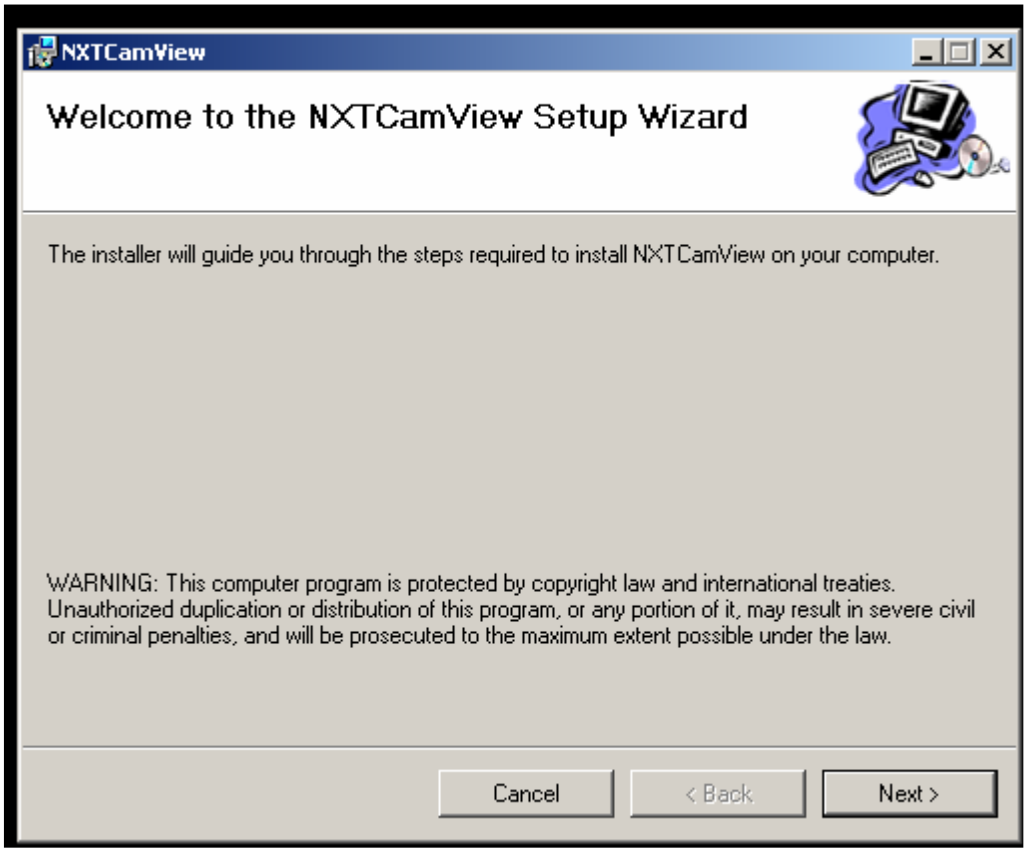


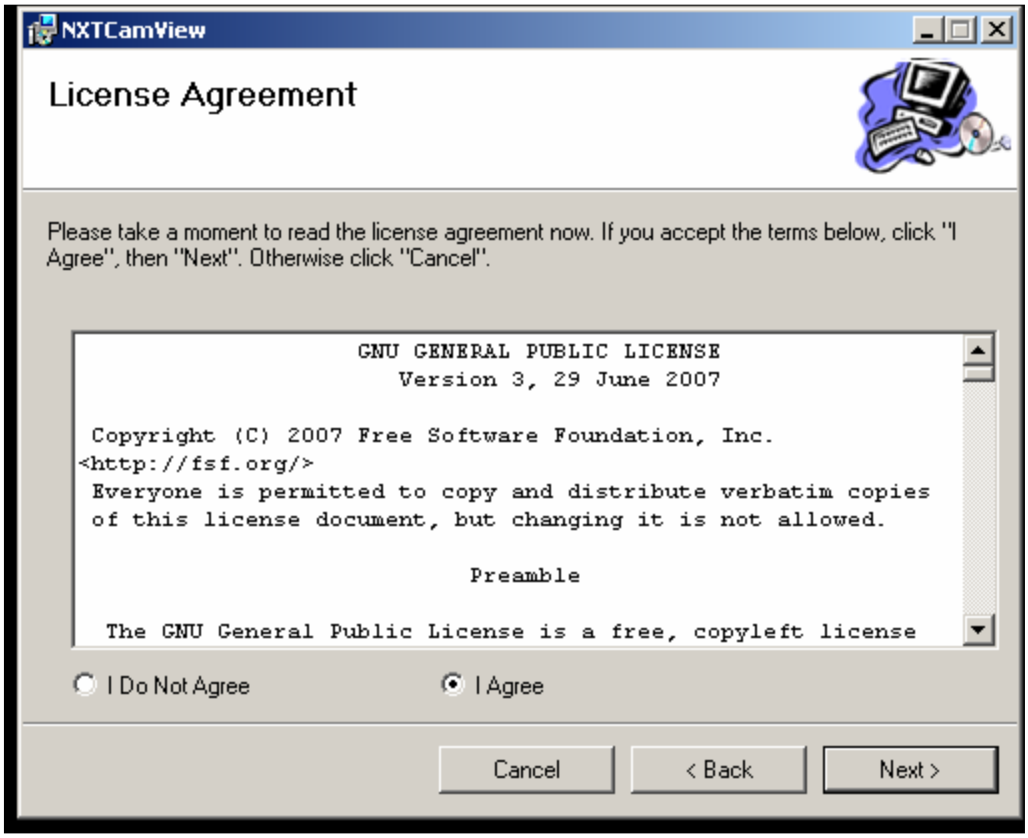
2.8.2.3.- Install NXTCamView

NXTCamView is a .NET application developed by Mindsensor community and stored in the website www.sourceforge.net .Latest release of NXTCamView can be downloaded in the following URL: http://nxtcamview.sourceforge.net/

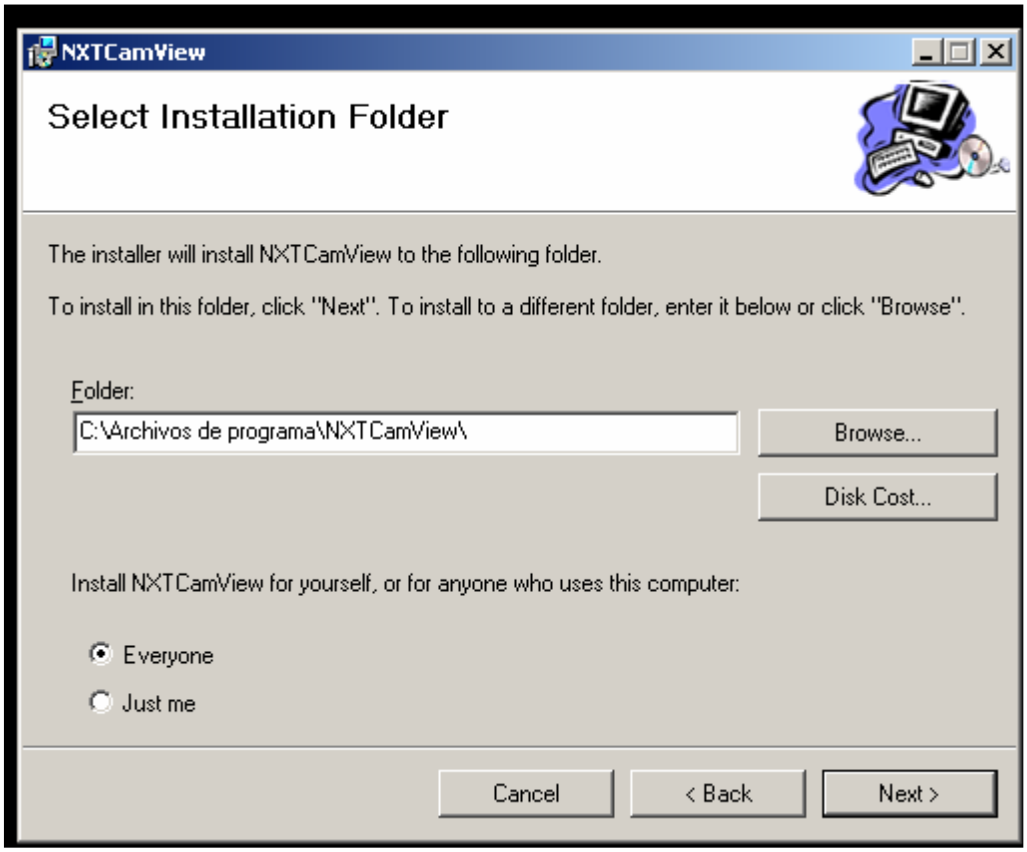
When you download the software and init the installation, NXTCamView offer you an assistant in the installation's process.

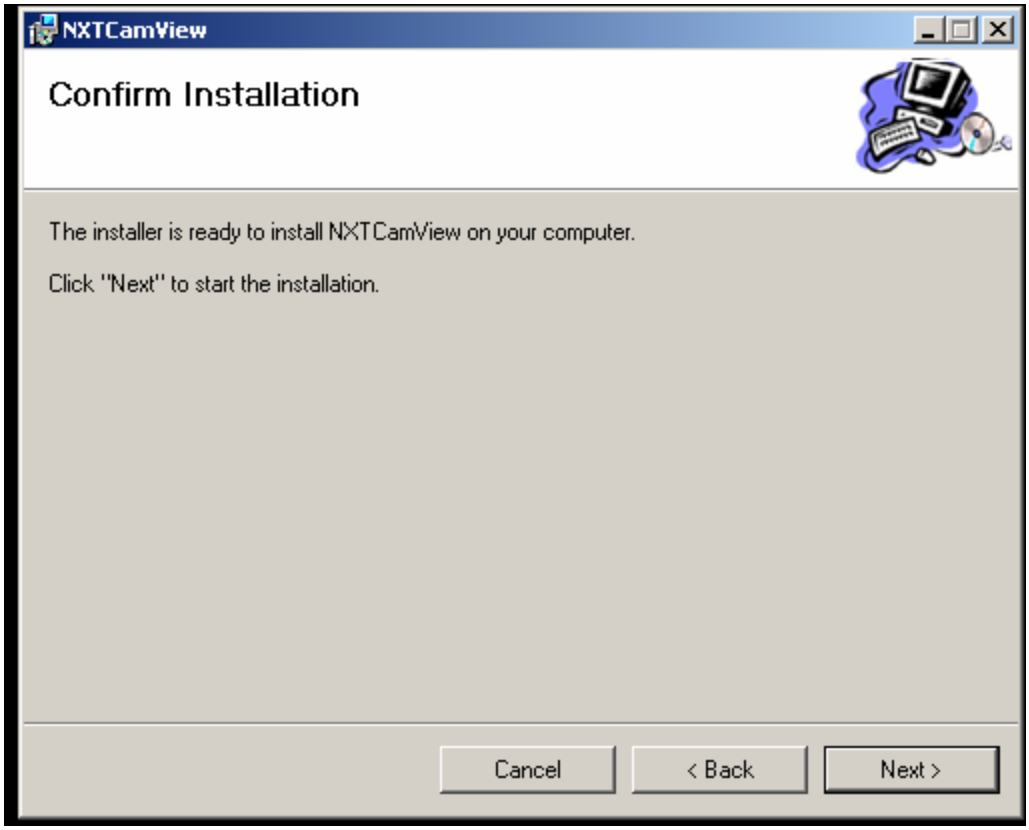
Step1: Accept licence:





Step2: installation path





2.8.2.4.- Using NXTCamView to Train NXTCam

NXTCam is a sensor that manages Color patterns then to establish when you use the tool NXTCamView.

In the following URL, http://nxtcamview.sourceforge.net/DemoScreenCam.htm, there are videos to establish color patterns.

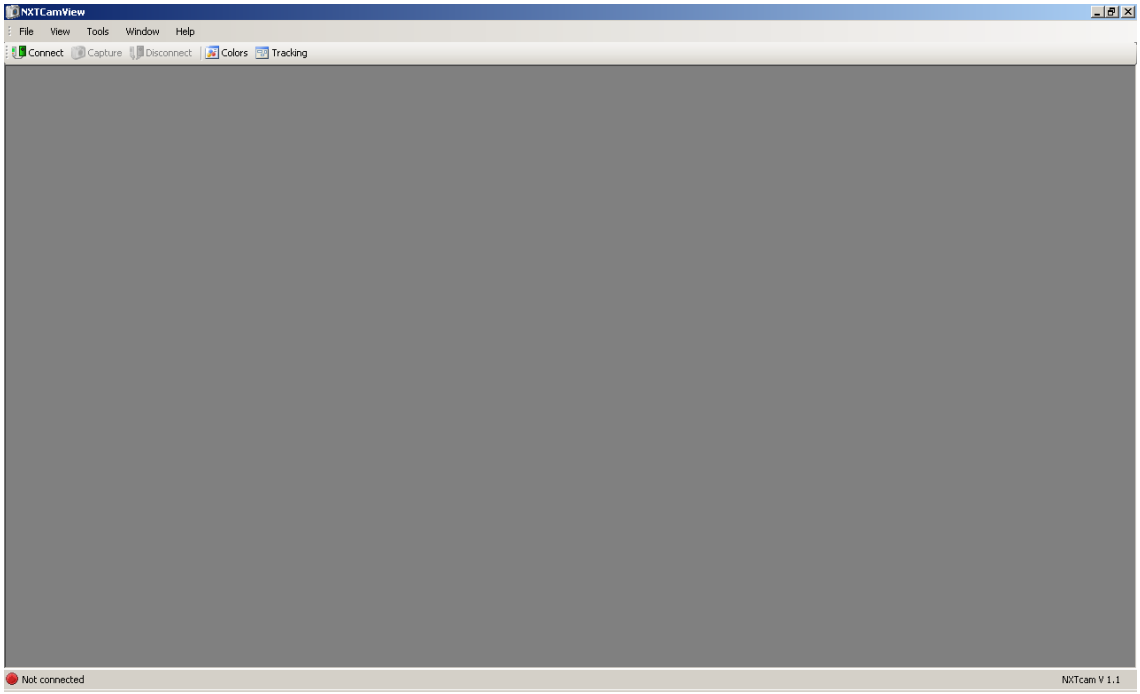
In this paper, we learn to detect the following objects:

1. Lego Air Tank
2. Fluorescent Text liner

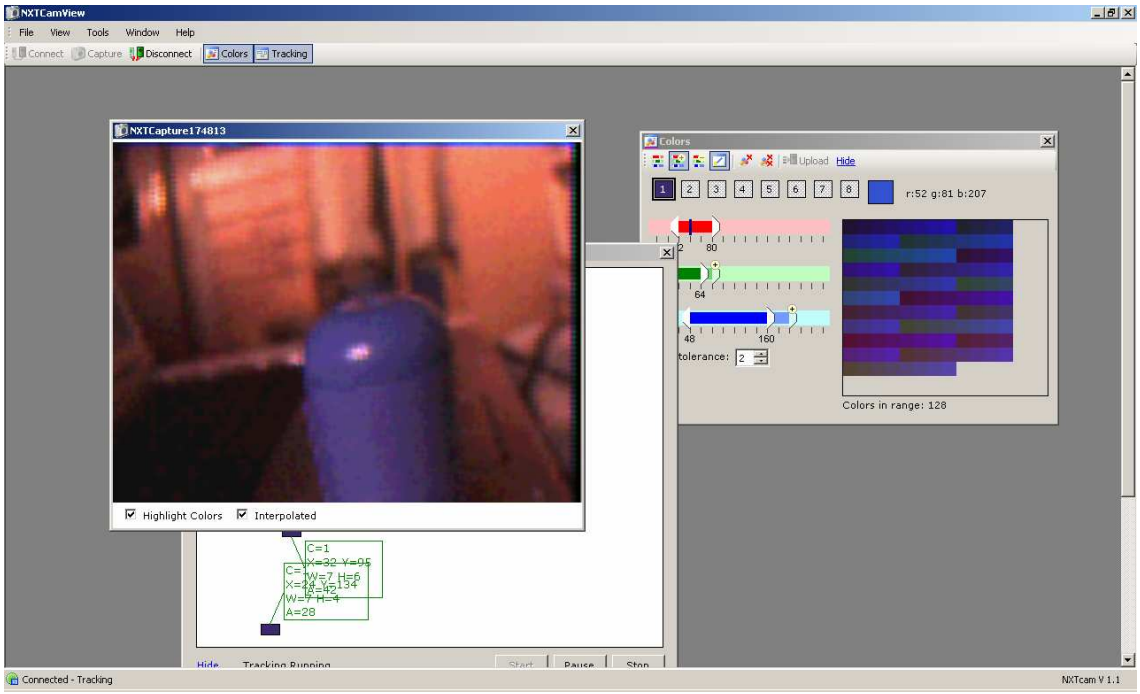
2.8.2.4.1.- Lego Air Tank

When you use Pneumatic Pieces, one typical piece is a Lego Air Tank:

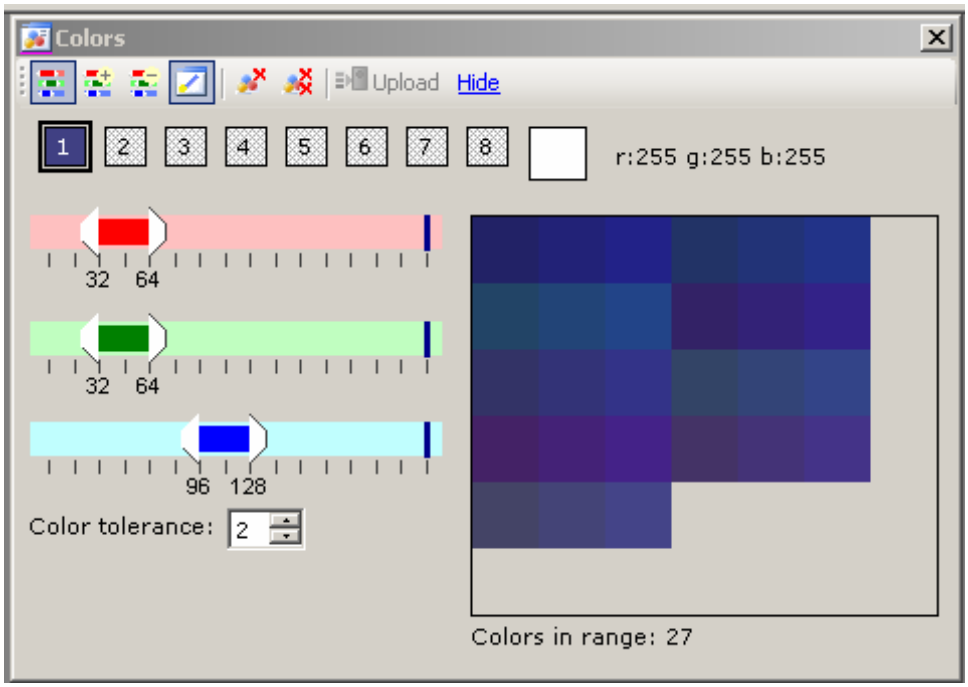
If we want to add a color map to detect Lego Air Tank Pieces then execute NXTCamView and connect NXTCam in your computer.



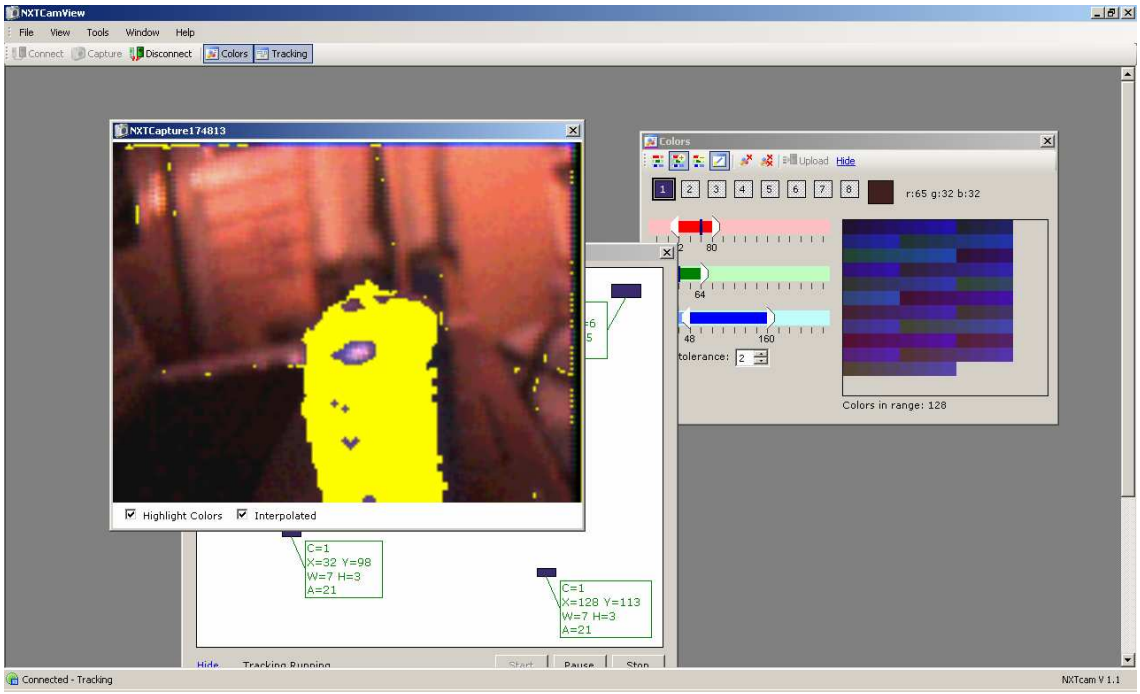
Once you have connected NXTCam, make an image capture:



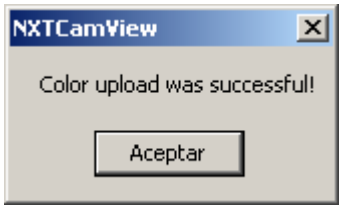
When you have an image that you have taken with NXTCam, create a color pattern using this tool:



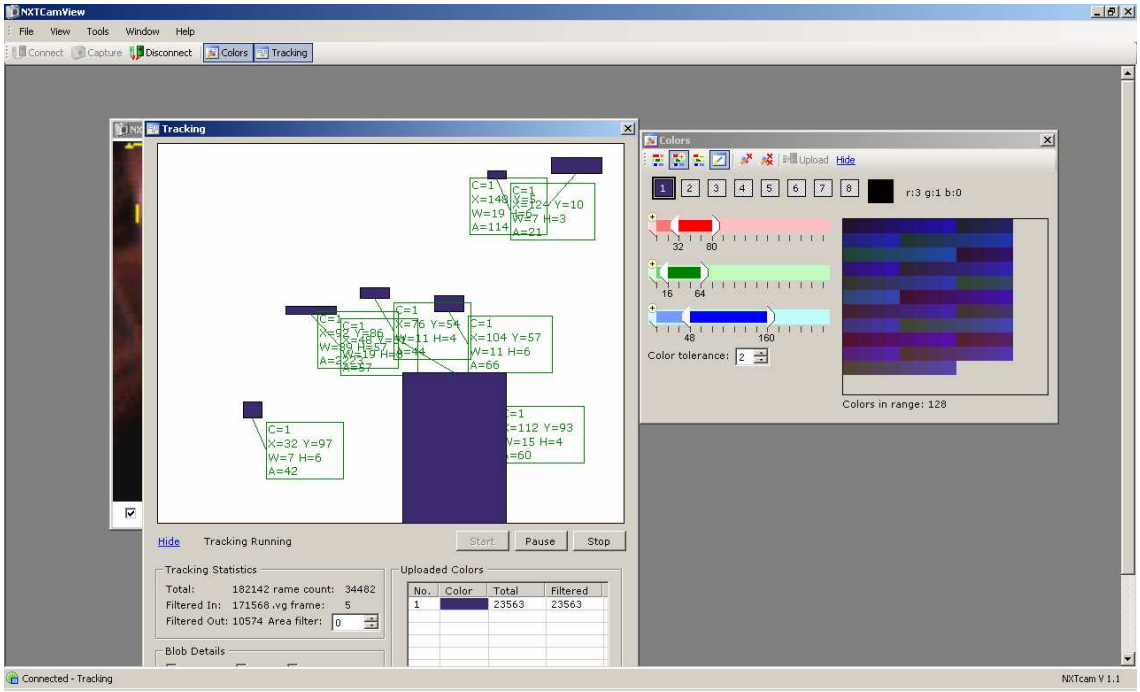
At the end of the process, you have a color pattern as the following:



Then you have to upload the color pattern to the NXTCam using the button **update**. When the process goes well, you will see the following alert:



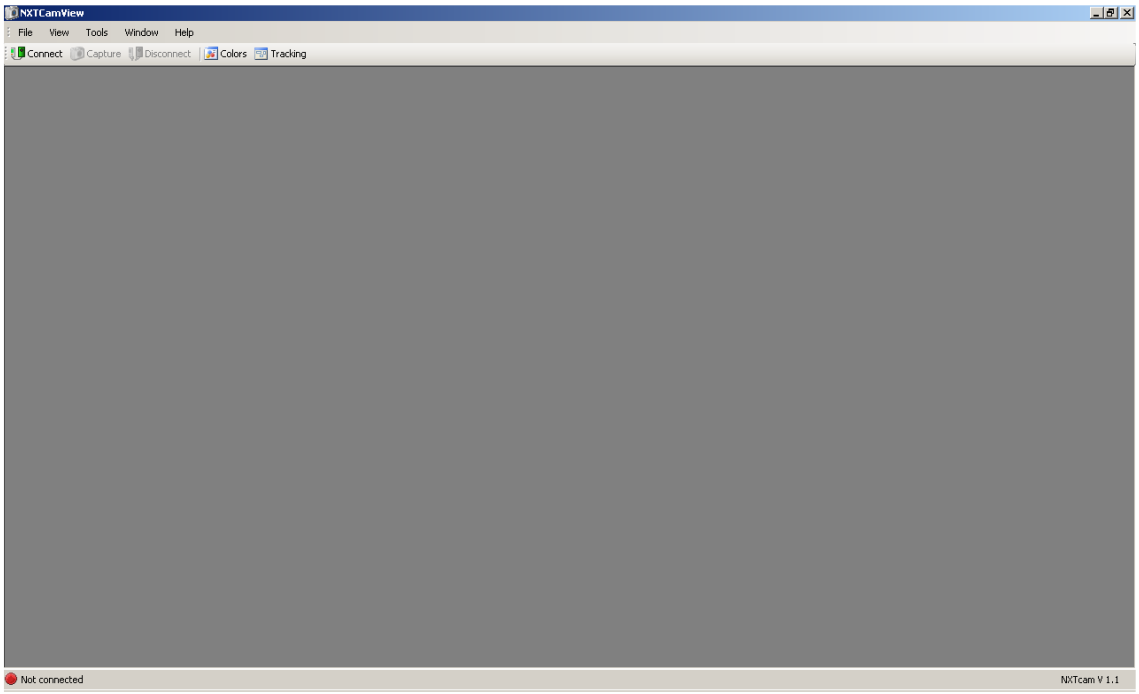
Test your color map pattern using the option tracking:



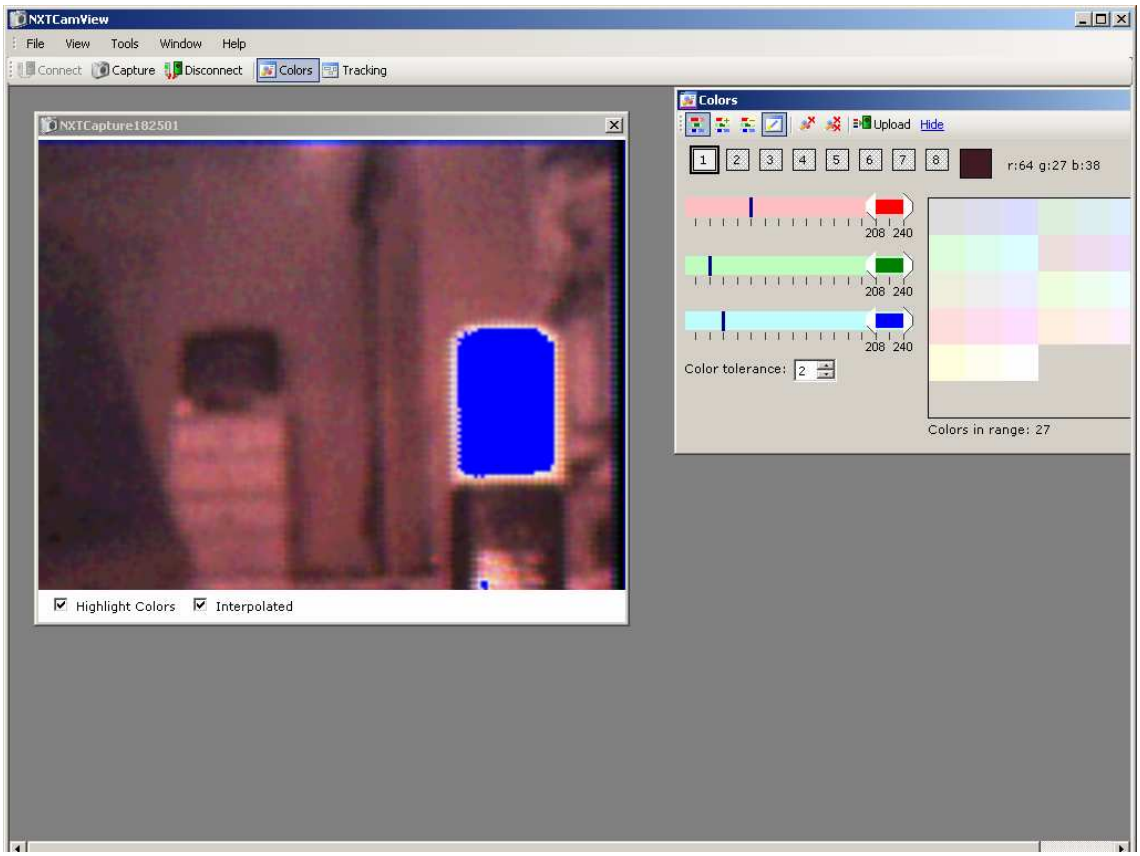
2.8.2.4.2.- Fluorescent Text liner

It is very common to have a Fluorescent Text liner at home or in your office, but our training it is a good item because the color is not usual in the environment then we create a pattern to detect this object.

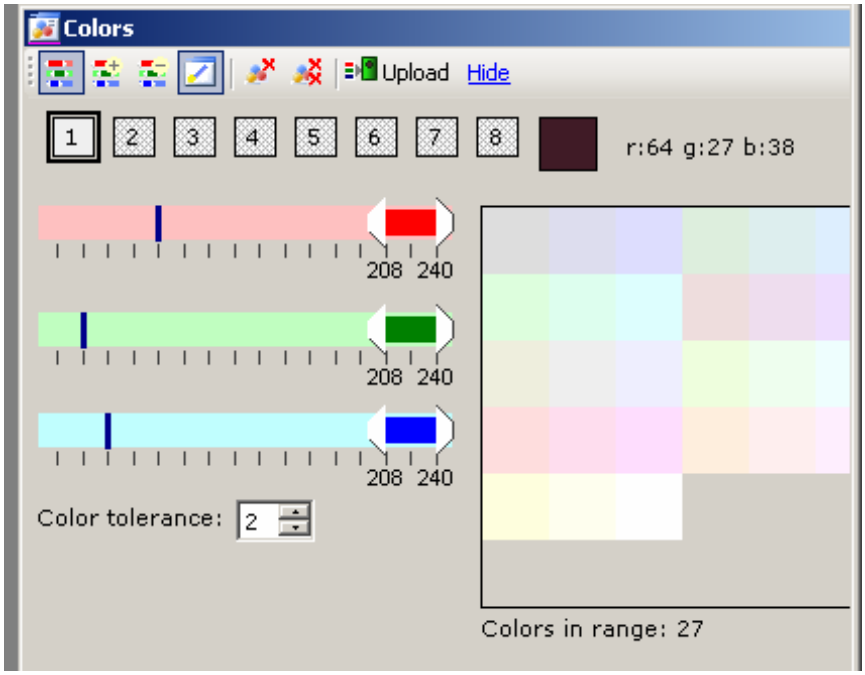
If we want to add a color map to detect Fluorescent Text liner then execute NXTCamView and connect NXTCam in your computer.



Once you have connected NXTCam, make an image capture:

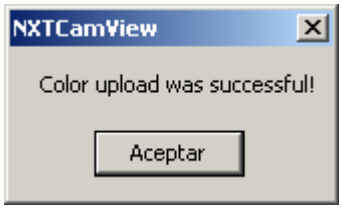


When you have an image that you have taken with NXTCam, create a color pattern using this tool:

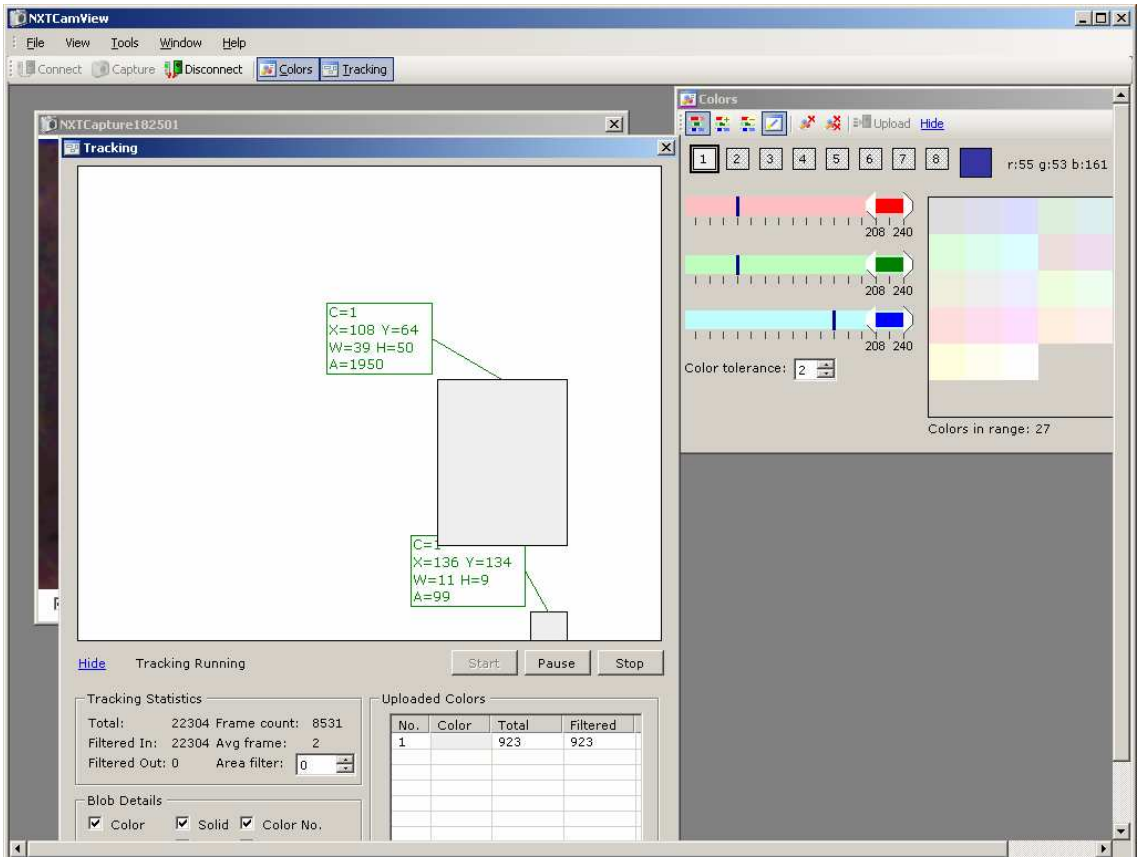


At the end of the process, you have a color.

Then you have to upload the color pattern to the NXTCam using the button **update**. When the process goes well, you will see the following alert:



Test your color map pattern using the option tracking:



Notes:

1. It is better if the object to track is different in relation to the environment
2. If you object is totally different, NXTCam will not detect false objects (Noise)

2.8.2.5.- NXTCam NXJ API

NXJ allows managing NXTCam using the class NXTCam the current public methods are the following:

Method Summary	
int	**getNumberOfObjects**() Get the number of objects being tracked
int	**getObjectColor**(int id) Get the color number for a tracked object
Rectangle	**getRectangle**(int id) Get the rectangle containing a tracked object
void	**sendCommand**(char cmd) Send a single byte command represented by a letter

- **GetNumberOfObjects:** NXTCam is able to track a maximum of 8 objects in real time. This method returns the number of objects tracked by the sensor.
- **GetObjectColor:** When you track an object, it is possible to know the color detected in that object.
- **GetRectangle:** when you track an object, it is possible to return a rectangle Object to know the position and size.
- **SendCommand:** NXTCam allow doing several operations using I2C. Using this method you can interact with NXTCAM's I2C registers.

2.8.2.6.- NXTCam API

In the document Nxtcam-V11-User-Guide.pdf explain the way to read and write I2C registers to use correctly NXTCam sensors. The sensor has several registers but if we sort these registers, there are 3 kinds of registers:

- Configuration's Registers
- Tracking's registers
- Colormap's register

Configuration's Registers

Commands		Action
ASCII	Hex	
A	0x41	Sort tracked objects by size
B	0x42	Select Object tracking mode
C	0x43	Write to Camera Registers Use extreme CAUTION when using command C since this can stop your camera working properly. In case this happens, please power off your NXTCam and power it on again.
D	0x 44	Disable Tracking
E	0x45	Enable Tracking
G	0x47	Get the Color map from Camera Engine
H	0x48	Read data from the Camera Registers
I	0x49	Illumination on (Future)
L	0x 4C	Select Line tracking mode
N	0x4E	Set ADPA mode ON (setting stored in NVRAM)
O	0x4F	Set ADPA mode Off (default) (setting stored in NVRAM)
P	0x50	Ping camera Engine
R	0x52	Reset Camera Engine
S	0x53	Send the color map to camera Engine
T	0x54	Illumination Off
U	0x55	Sort tracked objects by color
V	0x56	Get camera Engine firmware version No
X	0x58	Do not Sort tracked objects

Tracking's Registers

Register	Read	Write	Comments
0x00-0x07	Software version - *(V1.10)*	-	
0x08-0x0f	Vendor Id - *mndsnsrs*	-	
0x10-0x17	Device ID - *NXTCam*	-	
0x41	-	Command	This register is command register. A command written here will be executed.
0x42	Number of objects detected	-	Shows how many objects are being tracked. Zero indicates that there are no objects being tracked.
0x43	1st object color	-	This is the first object color as per the sorting method selected.
0x44[1]	1st object - X upper left		Upper left X coordinate of first object
0x45	1st object - Y upper left		Upper left Y coordinate of first object
0x46	1st object - X lower right		Lower right X coordinate of first object
0x47[2]	1st object - Y lower right		Lower right Y coordinate of first object
0x48	2nd object color		
0x49-0x4C	2nd object co-ordinates		
0x4D	3rd object color		
0x4E-0x51	3rd object co-ordinates		
0x52	4th object color		
0x53-0x56	4th object co-ordinates		
0x57	5th object color		
0x58-0x5B	5th object co-ordinates		

Register	Read	Write	Comments
0x5C	6th object color		
0x5D-0x60	6th object co-ordinates		
0x61	7th object color		
0x62-0x65	7th object co-ordinates		
0x66	8th object color		
0x67-0x6A	8th object co-ordinates		
0x6B	No. of registers to Read	No. of registers to Write	This is the number of registers you need to read or write from camera image sensor
0x6C	1st Camera register Address	1st Camera register Address	
0x6D[3]	1st Camera register Data	1st Camera register Data	1st register Data read from image sensor or written to image sensor
.....	.....	.....	
0x7A	8th Camera register Address	8th Camera register Address	
0x7B	8th Camera register Data	8th Camera register Data	

Colormap's registers

0x80[4]	Color map data Red 0	Color map data Red 0	0x80 – 0xAF - These registers are used for Colormap data reading and writing

Register	Read	Write	Comments
0x80	Color map data Red 0	Color map data Red 0	
0x81	Color map data Red 1	Color map data Red 1	
0x82	Color map data Red 2	Color map data Red 2	
0x83	Color map data Red 3	Color map data Red 3	
0x84	Color map data Red 4	Color map data Red 4	
0x85	Color map data Red 5	Color map data Red 5	
0x86	Color map data Red 6	Color map data Red 6	
0x87	Color map data Red 7	Color map data Red 7	
.....	.....	.....	
0x8F	Color map data Red 15	Color map data Red 15	
0x90	Color map data Green 0	Color map data Green 0	
0x91	Color map data Green 1	Color map data Green 1	
.....	.....	.....	
0x9F	Color map data Green 15	Color map data Green 15	
0xA0	Color map data Blue 0	Color map data Blue 0	
0xA1	Color map data Blue 1	Color map data Blue 1	
.....	.....	.....	
0xAF	Color map data Blue 15	Color map data Blue 15	

2.8.3.- Lattebox NXTe

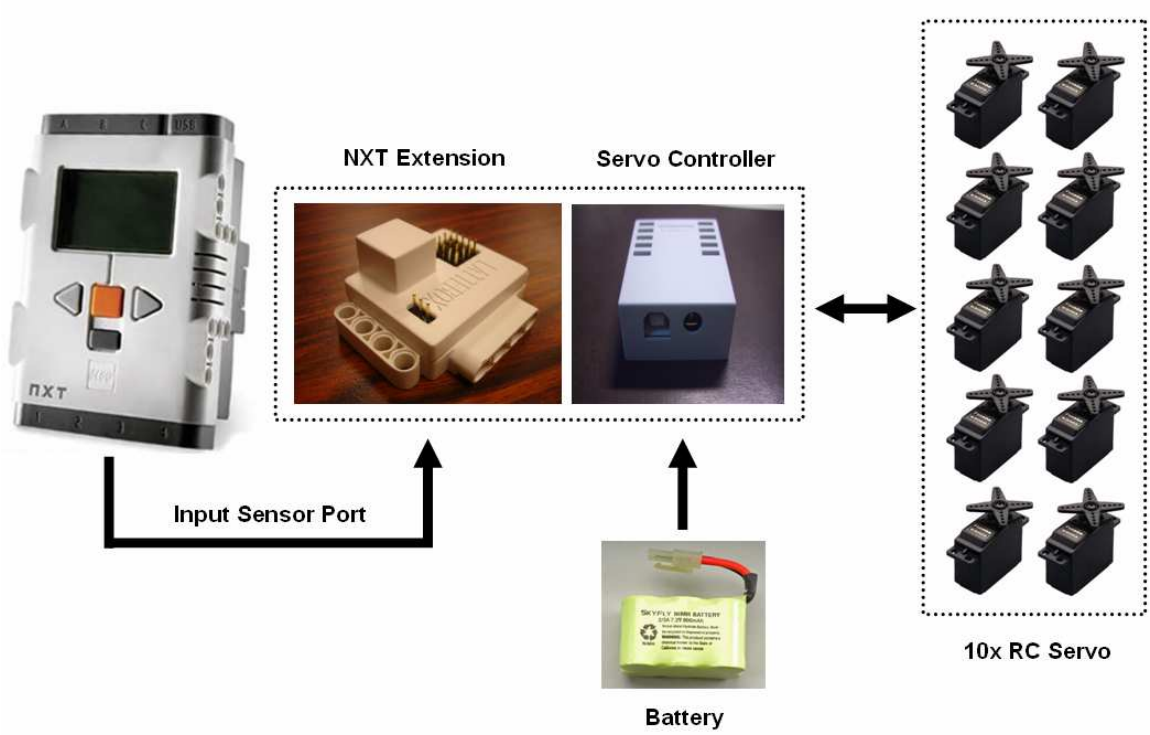
2.8.3.1.- Introduction

In 2008, Lattebox a hi-tech company located in Taiwan, launch a new kind of NXT device, NXTe. NXTe allows controlling RC servos easily. NXT brick has 4 sensor port inputs 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.

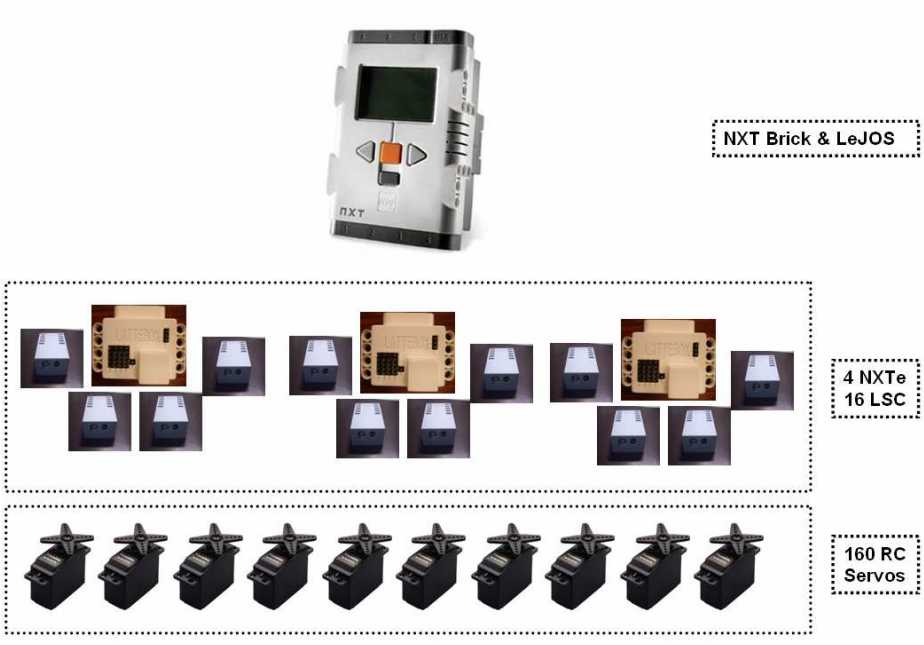
2.8.3.2.- Lattebox NXTe architecture

The NXTe architecture is the following:



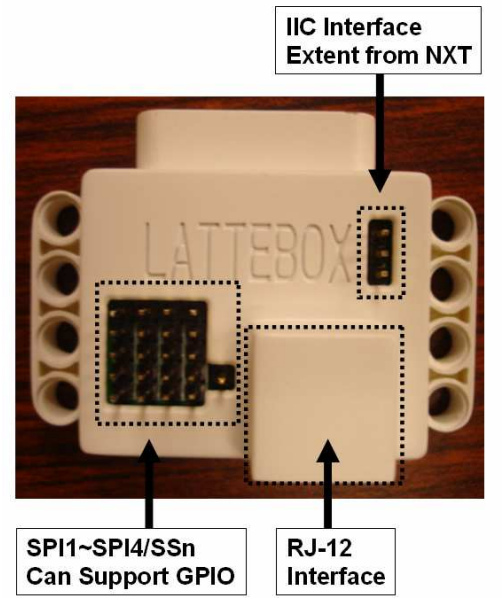
If you notice, with NXTe technology you can control:

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



2.8.3.2.1.- NXT Extension, NXTe

NXT Extension is new device developed by Lattebox to stablish 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.

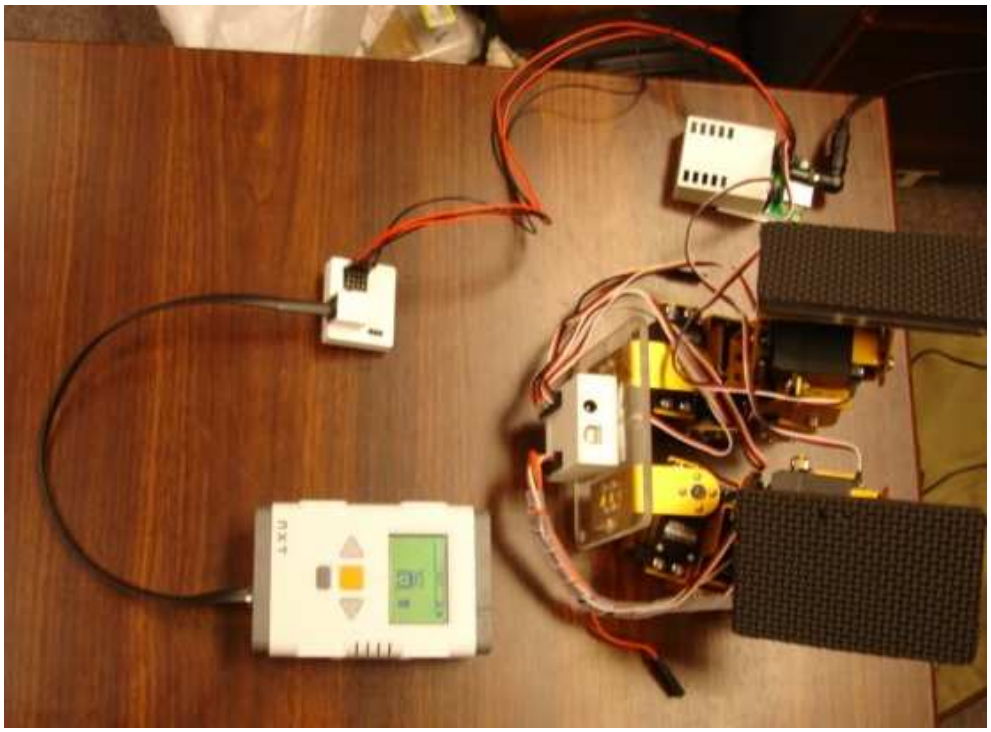


2.8.3.2.2.- Servo Controller

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 a any servo into LSC, you have to know pin details:

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

2.8.3.3.- 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.

2.8.3.3.1.- 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 travelling 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.

2.8.3.4.- LeJOS and Lattebox

2.8.3.4.1.- Lattebox Support in LeJOS

Currently leJOS support NXTe but it is neccesary to improve:

- Load a unique servo
- Unload all servos in a LSC
- Unload a unique servo
- Test NXTe with DC Motors
- Control Servo 2-10 (Currently only manage 1 Servo)
- Manage speed in Servos

2.8.3.4.2.- Example using lattebox classes with leJOS

If you use leJOS, currently exist in Beta Phase a set of classes to manage NXTe, LSC and Servos easily.

```
import lejos.nxt.*;
```

```
public class LatteboxTest{
	private static NXTe NXTeObj;
	private static DebugMessages dm;
	private static int angle;
	private static int angle2;
	private static int motion;

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

		try{

			NXTeObj = new NXTe(SensorPort.S1);//NXTe Controller
pluged in Port1
			NXTeObj.addLSC(0);
			dm.echo("Calibrating LSC");
			//Servo 1 connected in location 1
			NXTeObj.LSC(0).addServo(1,"SAVOX, Digital SC-0352");
			//Servo 2 connected in location 3
			NXTeObj.LSC(0).addServo(3,"SAVOX, Digital SC-0352");
			//NXTeObj.LSC(0).addServo(2,"HITEC, HS-785HB");
			NXTeObj.LSC(0).calibrate();
			dm.echo("Load all servos");
			NXTeObj.LSC(0).loadAllServos();
			NXTeObj.LSC(0).Servo(0).setMinAngle(0);
			NXTeObj.LSC(0).Servo(0).setMaxAngle(2000);
			NXTeObj.LSC(0).Servo(1).setMinAngle(0);
			NXTeObj.LSC(0).Servo(1).setMaxAngle(2000);


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

				if (Button.LEFT.isPressed()){
					NXTeObj.LSC(0).Servo(0).goToMinAngle();
					NXTeObj.LSC(0).Servo(1).goToMinAngle();
					while(NXTeObj.LSC(0).Servo(0).isMoving()
== true){}
					angle =
NXTeObj.LSC(0).Servo(0).getAngle();
					angle2 =
NXTeObj.LSC(0).Servo(1).getAngle();
					dm.echo("Goto Min");
					dm.echo(angle);
				}

				if (Button.ENTER.isPressed()){

	NXTeObj.LSC(0).Servo(0).goToMiddleAngle();

	NXTeObj.LSC(0).Servo(1).goToMiddleAngle();
					while(NXTeObj.LSC(0).Servo(0).isMoving()
== true){}
					angle =
NXTeObj.LSC(0).Servo(0).getAngle();

					angle =
NXTeObj.LSC(0).Servo(1).getAngle();
```

```
                              dm.echo("Goto Middle");
                              dm.echo(angle);
                              dm.echo(angle2);
                        }

                        if (Button.RIGHT.isPressed()){
                              NXTeObj.LSC(0).Servo(0).goToMaxAngle();
                              NXTeObj.LSC(0).Servo(1).goToMaxAngle();
                              while(NXTeObj.LSC(0).Servo(0).isMoving()
== true){}
                              angle =
NXTeObj.LSC(0).Servo(0).getAngle();
                              angle =
NXTeObj.LSC(0).Servo(1).getAngle();

                              dm.echo("Goto Middle");
                              dm.echo(angle);
                              dm.echo(angle2);
                        }
                  }

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

            dm.echo("Test finished");
      }
}
```

2.9.- NXJ Robots

2.9.1.- Introduction

The projects that I include in this ebook are the following:

- RC Car
- Radar

2.9.2.- RC Car

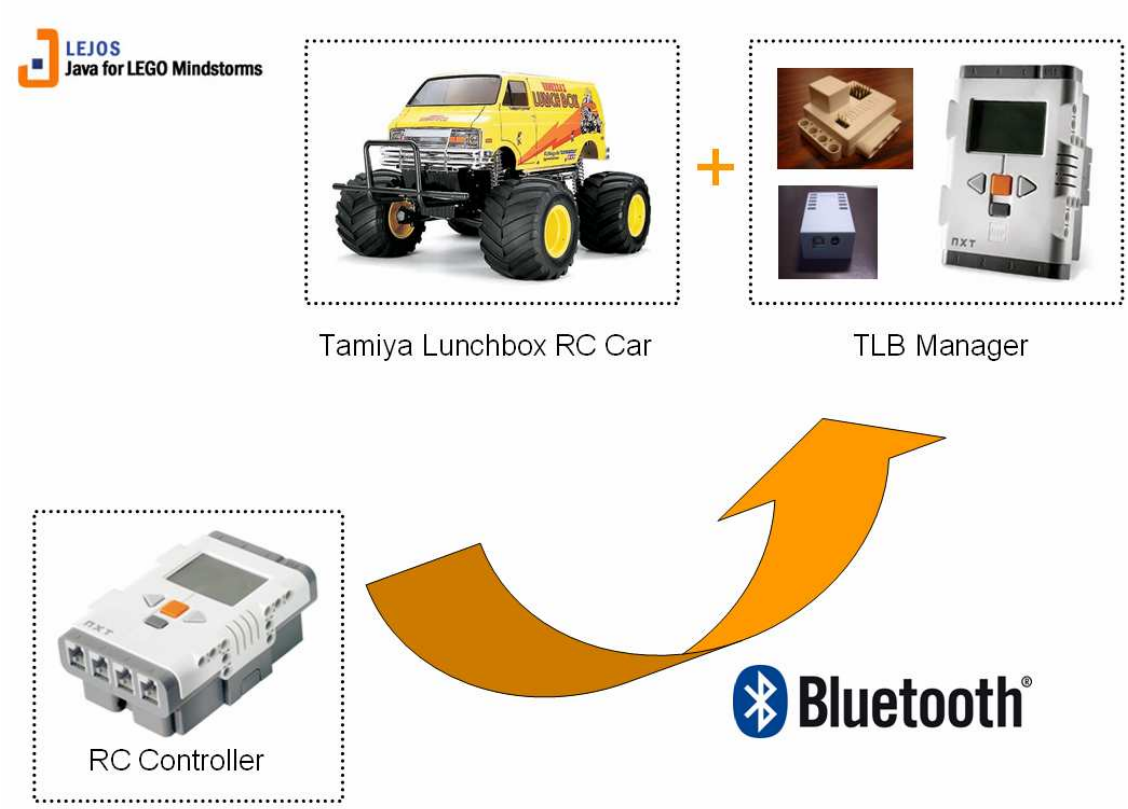
2.9.2.1.- Introduction

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.

2.9.2.2.- Architecture

The component architecture about this project is the following:



For this project I have used a classica RC Model, a 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.

2.9.2.3.- Software components

Software components used in this solution are listed above:

1. RC Controller
2. TLB Manager
3. TLB Communication Protocol 1.0
4. RC Diagnostics

2.9.2.3.1.- RC Controller

RC Controller is a leJOS solution developed to communicate with LTB Manager using bluetooth. This component use TLB Communication Protocol 1.0.

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

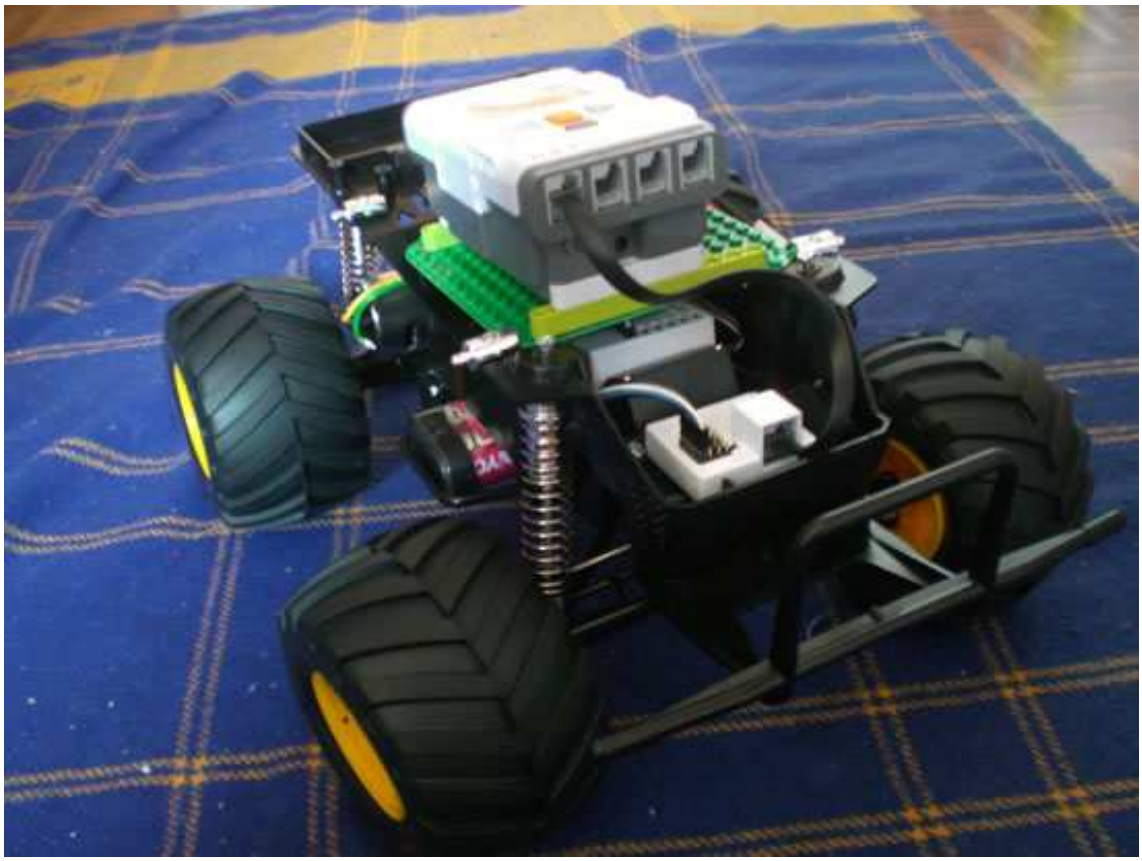
1. Locomotion Power
2. Direction
3. Steering

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.

2.9.2.3.2.- TLB Manager

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

1. Manage bluetooth communications
2. Manage TLB



2.9.2.3.3.- TLB Communication Protocol 1.0

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

If the communication use 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

2.9.2.3.4.- 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 neccesary 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 neccesary 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 stablish maximum level, minimum level, security zone to run.

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

2.9.2.4.- Futher information

Further information about this project here:
http://www.juanantonio.info/jab_cms.php?id=228

2.9.3.- Radar

Not Available

2.10.- Robotic Researches

2.10.1.- Introduction

Not Available

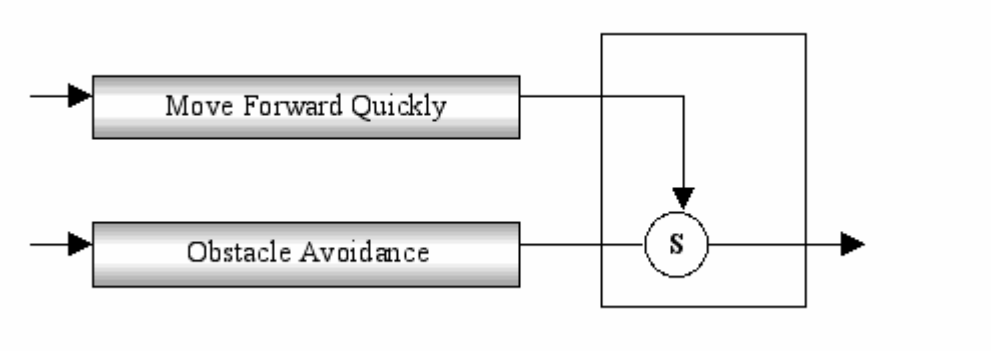
2.10.2.- Subsumption architecture

2.10.2.1.- Introduction

Subsumption architecture is a methodology for developing Artificial intelligence robots. It is heavily associated with behavior-based robotics. The term was introduced by Rodney Brooks and colleagues in 1986. Subsumption has been widely influential in autonomous robotics and elsewhere in real-time AI.

Subsumption architecture is a way of decomposing complicated intelligent behavior into many "simple" behavior modules, which are in turn organized into layers. Each layer implements a particular goal of the agent, and higher layers are increasingly more abstract. Each layer's goal subsumes that of the underlying layers, e.g. the decision to move forward by the eat-food layer takes into account the decision of the lowest obstacle-avoidance layer.

For example, a robot's lowest layer could be "avoid an object", on top of it would be the layer "wander around", which in turn lies under "explore the world". The top layer in such a case could be "create a map", which is the ultimate goal. Each of these horizontal layers access all of the sensor data and generate actions for the actuators — the main caveat is that separate tasks can suppress (or overrule) inputs or inhibit outputs. This way, the lowest layers can work like fast-adapting mechanisms (e.g. reflexes), while the higher layers work to achieve the overall goal. Feedback is given mainly through the environment.



2.10.2.2.- Design the robot's intelligence with Subsumption Architecture

Not Available

2.10.2.3.- Subsumption package in NXJ

Currently NXJ has an exclusive package to develop robots with this architecture. NXJ has 2 classes to control the whole process:

1. Behavior
   a. Action()
   b. Suppress()
   c. takeControl()
2. Arbitrator

2.10.2.4.- Subsumption example

When you download latest NXJ release, in samples folder there is a Subsumption example named BumperCar.

This example has the following classes:

1. BumperCar.java
2. DriveForward.java
3. HitWall.java

Now I will explain the code:

BumperCar.java

```
import lejos.subsumption.*;
import lejos.nxt.*;

public class BumperCar {
   public static void main(String [] args) {
      Behavior b1 = new DriveForward();
      Behavior b2 = new HitWall();
      Behavior [] bArray = {b1, b2};
      Arbitrator arby = new Arbitrator(bArray);
      Motor.A.setSpeed(200);
      Motor.C.setSpeed(200);
      arby.start();
   }
}
```

When you design any robot's logic using Subsumption Architecture, you must to define the behaviors. In this example, BumberCar has been designed with 2 behaviors:

1. HitWall
2. DriveForward

Every behavior must be code in a single class.

DriveForward.java

```
import lejos.subsumption.*;
import lejos.nxt.*;

public class DriveForward implements Behavior {

   public boolean takeControl() {
      return true;
   }

   public void suppress() {
      Motor.A.stop();
      Motor.C.stop();
   }

   public void action() {
      Motor.A.forward();
      Motor.C.forward();
   }
}
```

HitWall.java

```
import lejos.nxt.*;
import lejos.subsumption.*;

public class HitWall implements Behavior {

   TouchSensor touch;

   public HitWall()
   {
         touch = new TouchSensor(SensorPort.S2);
   }

   public boolean takeControl() {
      return touch.isPressed();
   }

   public void suppress() {
      Motor.A.stop();
      Motor.C.stop();
   }

   public void action() {
      // Back up:
      Motor.A.backward();
      Motor.C.backward();
      try{Thread.sleep(1000);}catch(Exception e) {}
      // Rotate by causing only one wheel to stop:
```

```
        Motor.A.stop();
        try{Thread.sleep(300);}catch(Exception e) {}
        Motor.C.stop();
    }
}
```

If you notice every behavior is necessary to define 3 methods:

1. takeControl()
2. suppress()
3. action()

The method takeControl is used in the Subsumption architecture to indicate when the behavior is active. In the case of the behavior HitWall, the behavior is active when the Touch Sensor is pressed then the method action runs.

If we analyze the example in any moment the lowest behavior, Driveforward is running because the method takeControl always returns true, but if in any moment, the robot is pressed the Touch Sensor, in this case exist a conflict between 2 takeControls, but the behavior HitWall has a higher priority in relation to DriveForward, then the HitWall's action runs.

2.10.3.- Neural Networks

Not Available

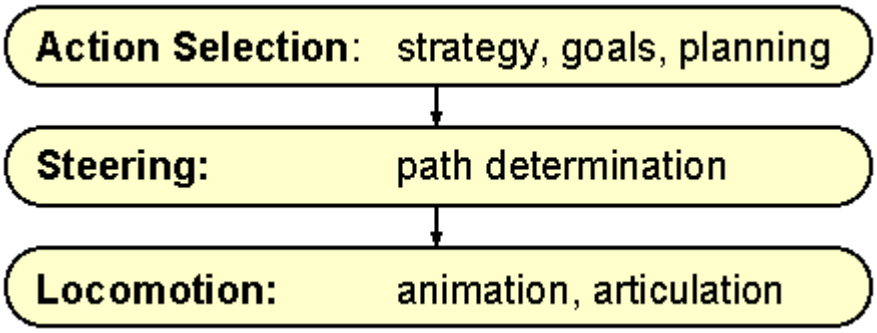
2.10.4.- Kalman filter

Not Available

2.10.5.- Steering behaviors

2.10.5.1.- Introduction

Steering Behaviors are the next logical step in the further development of the 'boids' model created by Craig Reynolds in 1986. The 'boids' computer based model is a way to simulate the coordinated movements seen in flocks of birds or fish. The name 'boids' identifies here one instance of the simulated creatures. The original flocking model was based on three distinct behaviors. The 'Separation' behavior was used to prevent situations where members of the swarm were crowding each other. 'Aligment' was used to steer all elements of the flock in a common direction. The third behavior, 'Cohesion', steered a single 'boid' to the average position of nearby flockmates. Reynolds was able to create a realistic simulation of the natural behaviors of swarm through a skillful combination of these three simple behaviors.



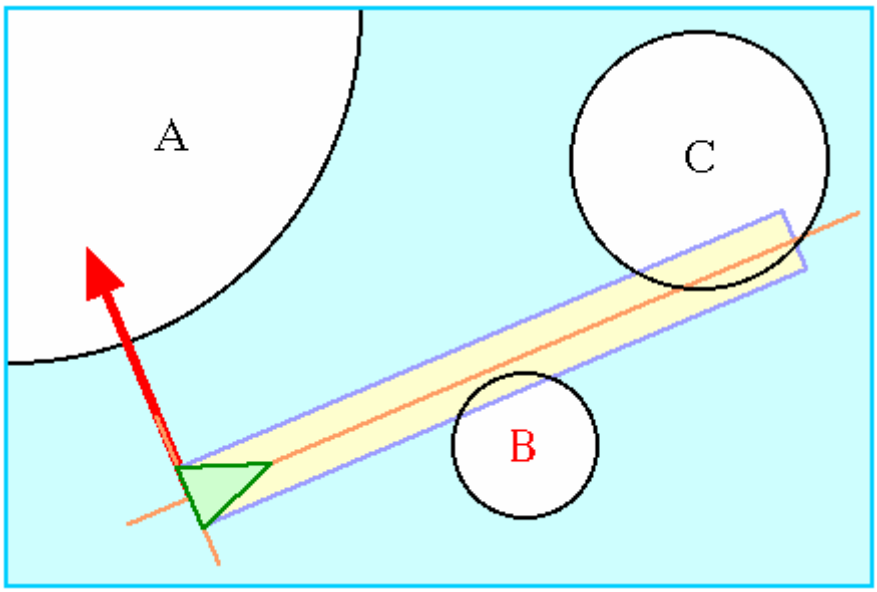
2.10.5.2.- Behaviors

The behaviors researched by Craig Reinolds and Robin Green are:

1. Simple behaviors for individuals and pairs:
    a. Seek and Flee
    b. Pursue and Evade
    c. Wander
    d. Arrival
    e. Obstacle Avoidance
    f. Containment
    g. Wall Following
    h. Path Following
    i. Flow Field Following
2. Combined behaviors and groups:
    a. Crowd Path Following
    b. Leader Following
    c. Unaligned Collision Avoidance
    d. Queuing (at a doorway)
    e. Flocking (combining: separation, alignment, cohesion)

We consider very interesting the behavior Obstacle Avoidance in a Robotic context. See the behavior Obstacle Avoidance in action:
http://www.red3d.com/cwr/steer/Obstacle.html

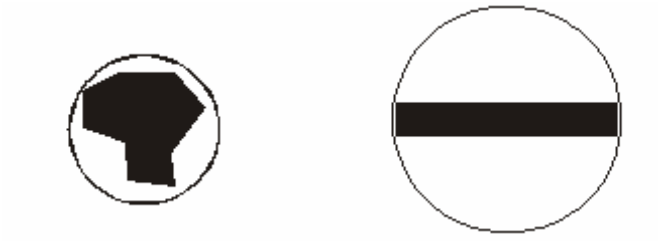
2.10.5.3.- Obstacle Avoidance



Application

To steer a vehicle through a predefined landscape in a realistic way it is necessary to move it the way a spectator would expect it to. Therefore it is necessary to define a behavior that allows the handling of obstacles. In normal life humans and animals avoid bumping into obstacles without gibing it a second thought. When somebody feels the craving for a cold beer, he will not move to the fridge in a straight line. Unconsciously he will choose a path that will lead him around all the obstacles in the world, like tables, chairs and other things lying around in a well kept household, so he is not forced to give up his primary task to nurse his broken toe. The obstacle avoidance behavior implements this unconscious avoiding of predefined obstacles.

Implementation
First thing when implementing the obstacle avoidance behavior is the definition of the obstacle within the virtual environment. Since this simulation is based inside a two dimensional scene, using simple geometric forms as substitution for complex real life obstacles. The possibly simplest representation of an obstacle is the circle. The circle should enclose the whole obstacle. It should be taken care not to waste too much unnecessary space on the representation. For long walls a circle can be a really bad approximation. The advantages in using a circle are the simple formulas used in determining intersections with other geometric bodies since those are used a lot in the behavior. A disadvantage is the bad approximation of the complex shape. Unfortunately this is the case for most shapes used in this simulation.

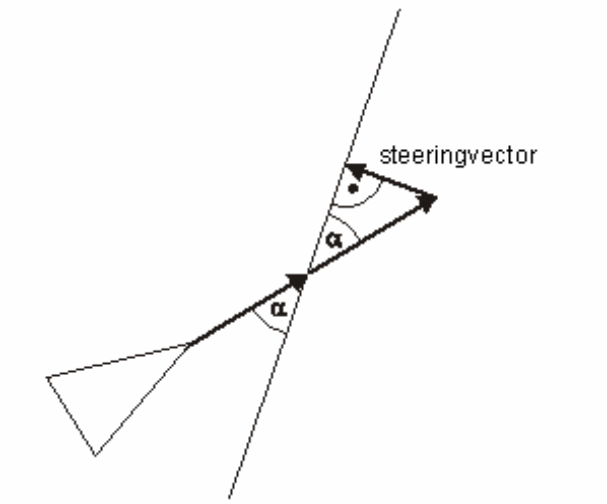


Since the disadvantages of using circles as placeholders clearly outweigh the advantages, the choice was made to use polygons instead. The advantage being that elements of real life, like chairs or tables for instance, can be better approximated. The results achieved in two-dimensional space can be applied to three-dimensional space without any problems.

The first step for the Obstacle Avoidance behavior is to find out which objects are to be found in its close vicinity. To get this information it queries the Neighborhood object whose job is to speed up spatial queries. By reducing the number of objects to be tested in the first step further test is sped up already. In large scenes the time used for making the spatial query is certainly less than the time used for the following intersection tests.

Calculating the angle of intersection and the distance

The getCollisionDetails(..) function is used to calculate the angle of intersection and the distance to the obstacle. The steering vector is based on the resulting values. To achieve this one, the testing vector is first enlarged by the distance to the intersection and after that projected vertically back onto the edge of the polygon. This vector is then used for the steering force.

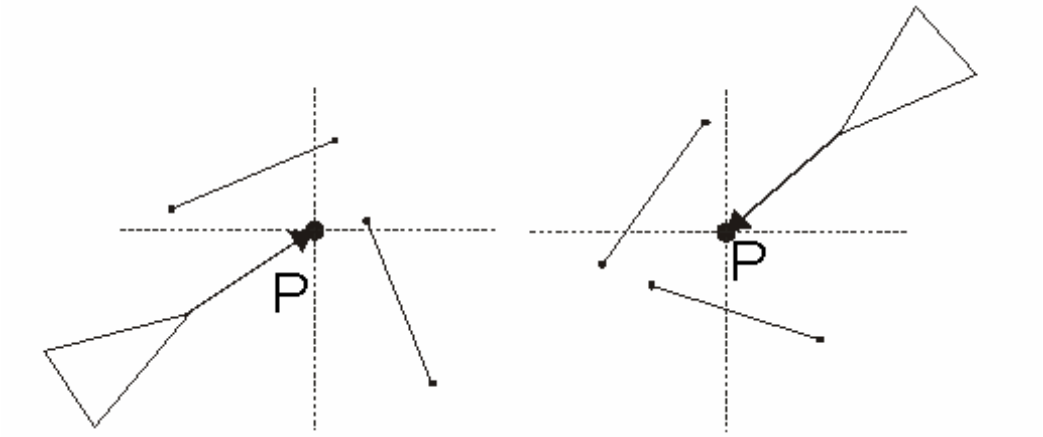


Optimization using "exclusion tests"

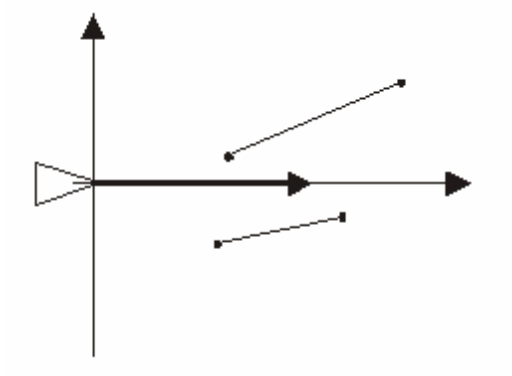
If there are obstacles within close vicinity of the vehicle is does not necessary imply that the testing vector is actually intersecting all of them. For instance if its position to one of the sides of the vehicle it will definitely not intersect the testing vector.

Since the Obstacle Avoidance behavior is used by many vehicles and calculating the steering force is a very computation intensive task, using so called "exclusion tests" can significantly speed up calculations. This kind of test only consist of simple "if" statements and are therefore pretty fast. For each edge of an obstacle these tests have to be performed.

Test 1:
First thing is calculating a test point P at the end of the testing vector. Now all edges with points outside of the defined area can be removed from further testing (see picture).

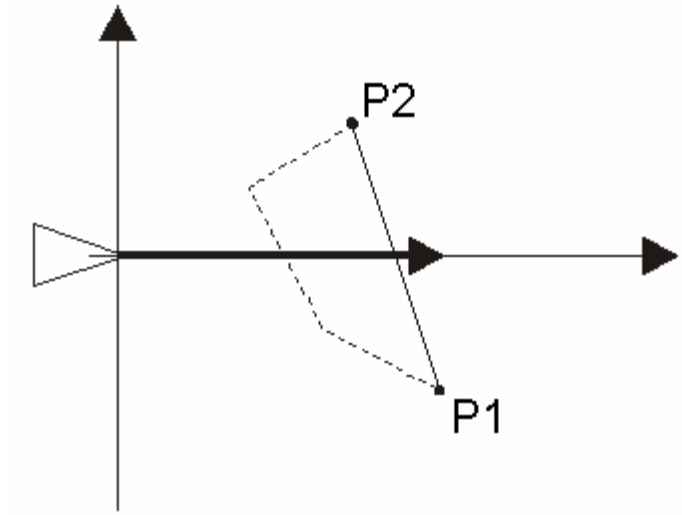


Test 2:
If the edge has passed the first test it will be transformed into the local space of the vehicle. The testing vector now represents the x-axis. If both endpoints of the edge are either above or below the x-axis, the edge does not intersect the testing vector. These edges can be discarded.

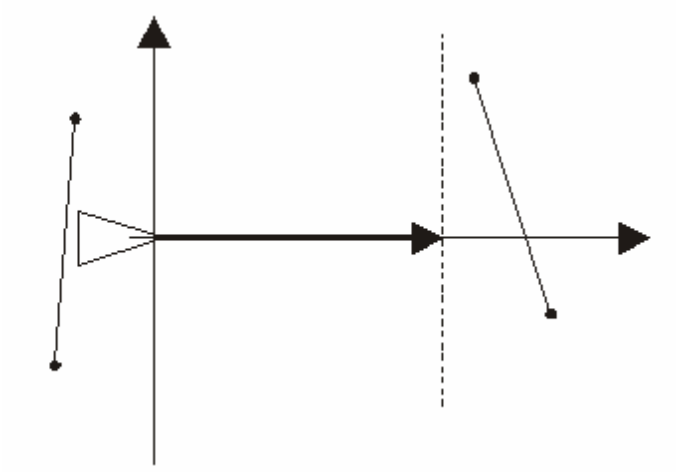


Test 3:

The third test is used to determine if the backside of the edge is facing the vehicle. This is often the case with small obstacles. These edges can be discarded as well. The edge points of a polygon are defined counterclockwise. Therefore edges with point 1 above and point 2 below the x-axis can be disregarded.



Test 4:
The fourth test is used to determine if both of the ends of the edge have x positions greater than the length of the testing vector. Also edges having x positions left of the y-axis can be discarded. In this case the obstacle is behind the vehicle.



After these four tests most of the edges have been discarded and some computation time has been saved. For the remaining edges the angle and distance for the intersection point has to be calculated. Basis for this is the formula: $y = a(x - x_1) + y_1$

Here x1 and y1 define the coordinates of the first edge point. The slope a is calculated using $a = \frac{(y_2 - y_1)}{(x_2 - x_1)}$. If this is inserted into the first formula, solved for x and y is set to zero, the resulting formula will be $x = \frac{(-y_1) \cdot (x_2 - x_1)}{(y_2 - y_1)} + x_1$

The resulting value for x is the distance of the intersection with the obstacle form the vehicle. To get the angle the arcustangens function is used.

Comment:
Since obstacles can be overlapping, all edges of these obstacles have to be considered in the tests. The edge with the smallest distance for the intersection is used for calculating the steering force. Therefore always the nearest obstacle is used.

2.10.5.4.- Creating a NXJ robot implementing the behaviour Obstacle Avoidance

Not Available

2.10.6.- Parallel Architectures

2.10.6.1.- Introduction

Parallel or Concurrent programming is thought to be one of the advanced ways to write computer programs. Concurrent programming running multiple tasks at the same time is able to improve the performance of the software application. When working with robots it would help if the robot would respond to sensor faster. One-way of speeding up the reaction of motors from sensor data is to build the robot controllers that make use of parallelism. Parallel programming for NXT Lego Mindstorm has been implemented at Napier University Edinburgh with JCSP re and LeJOS for Lego Mindstorm.

The University of Kent has been working with parallelism on the NXT's using Occam pi programming language. The research carried out by Professor Jon Kerridge, Alex Panayotopoulos and Patrick Lismore at Napier University Edinburgh resulted in achieving two working, line following robots that were built using a process orientated design pattern, JCSP re (communicating sequential processing for Java robot edition), leJOS and Bluetooth.

2.10.6.2.- JCSP re

JCSP re stands for "Communicating Sequential Processes for Java, Robot Edition" this is a reduced version of the original JCSP packages and work done at the University of Kent. Alex Panayotopoulos a Masters research student at Napier University Edinburgh has been working on JCSP re for his Masters dissertation and his the task was to reduce the JCSP packages to a suitable workable implementation that could be used to build highly concurrent robot controllers on a small robotic environment such as the NXT.

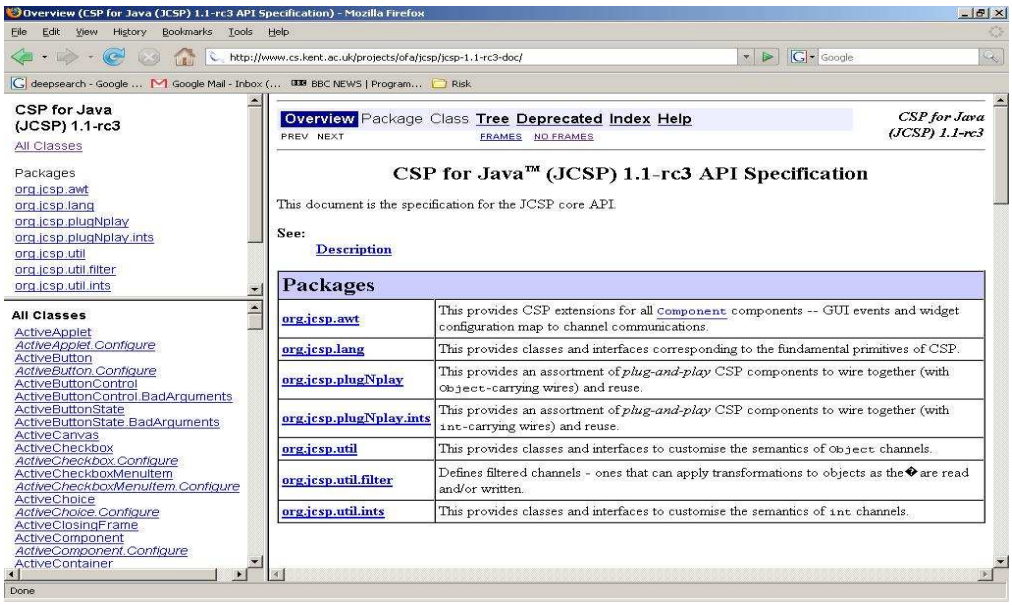
JCSP is a Java implementation of Communicating Sequential Processes; it consists of a library of packages that allows software developers to build concurrent programs. It originates from work initially done by Charles Antony Richard Hoare (Tony Hoare). Communicating Sequential Processes (CSP) was first documented in a paper released by Tony Hoare in 1978. CSP is a formal language for describing patterns of interactions in highly parallel systems. CSP is part of mathematics called process calculi that provides a high level description of interactions, communications and synchronization between independent active processes. CSP and work done by Tony Hoare played a big influence in the well-known Occam pi programming language, which also is being used on the Lego NXT robots. JCSP gives java software developers an environment to develop independent active processes that can be run concurrently. One of the fundamental concepts of JCSP is the idea of processes. JCSP enables software developers to create active processes that encapsulate data structures and algorithms that will act on the encapsulated data. An active process is defined as a CSP Process; a CSP process is independent and stands alone its data is private. CSP Processes communicate with its environment through well-defined "channels" of communication. The end result of a JCSP concurrent program is that all the processes are synchronized.

The JCSP library is quite large and thus not suitable for a small operating environment like that of the NXT robot. JCSP re however eliminates all the irrelevant packages and classes that have no use in a robot. Most obvious is the graphics related packages. The JCSP packages have active processes such as "Active Button", "Active Canvas", and "Active Applet". As active processes are the basis for working with JCSP it is necessary to design and develop active processes suitable for working with leJOS and the NXT.

JCSP and JCSP re have fundamental classes that enable developers to take advantage of the underlying CSP model for concurrency. JCSP re has classes such as Parallel, Alternative, One2OneChannel, and One2OneChannelInt to name a few.

You can browse JCSP API in the following URL:

http://www.cs.kent.ac.uk/projects/ofa/jcsp/
http://www.cs.kent.ac.uk/projects/ofa/jcsp/jcsp-1.1-rc3-doc/



You can browse JCSP re API in the following URL:

http://www.patricklismore.com/jcspre/index.html

2.10.6.3.- JCSP re Stack

The JCSP re Stack is the following

RobotController & User Interface processes
Process Components: ActiveMotor, ActiveLightSensor, …
JCSP re
LeJOS NXJ
Lego NXT Brick

The table above shows the hardware/software layers involved with building concurrent software for NXT robots. At the bottom is the NXT hardware on top of it sits the LeJOS NXJ virtual machine with the LeJOS API. JCSP re sits above the LeJOS API that is classes like parallel, alternative, One2OneChannel. Above the JCSP re sits Active processes those that are needed for the NXT robot. ActiveLightSensor, ActiveButton, ActiveMotor and above that sits user specific concurrent processes. ActiveUserInterface, ActiveRobotController.

2.10.6.4.- The Process Orientated Design Pattern

Using JCSP for developing software will naturally lead to following the Process Orientated Design pattern. The concept is framed around building software constructed with networks of processes. The approach aims to get developers thinking about the major functionality in the software and designing the software so that the major functions and logic are encapsulated in a process. Designing processes in such ways that if needed to processes can interact with other processes that contain other important functionality through synchronized channels. This approach allows simple processes and networks of processes to be scaled up to form highly concurrent systems. The processes that are built using JCSP are objects of a class implementing the CSProcess interface. This interface looks like this

```
public interface CSProcess {

  public void run()
  {

  }

}
```

Every process will implement this interface. This interface provides a run method. The run method is where developers will place their serial code. The code placed within the body of the run method will determine how the process behaves.

2.10.6.5.- LeJOS & JCSP re for building robot controllers

LeJOS and JCSP are both implemented in Java; LeJOS on its own is an excellent way to work with the NXT robots. It allows Java programmers to quickly target the NXT platform and it is also an excellent way for someone new to programming to learn quickly. JCSP and concurrent programming itself for traditional programmers is thought to be harder to learn and implement but this is not the case. Making use of the well-designed LeJOS API it's quite easy to build the "Active Processes" needed for the NXT with JCSP re. JCSP is a java implementation of the CSP model for concurrency. JCSP is an efficient and relatively simple way to build concurrent software. It has a quick learning curve, software developers don't need to understand the underlying concurrency model they just benefit from just using it.

The major active processes needed are:

- ActiveButton
- ActiveMotor
- ActiveDisplay
- ActiveBinaryFilter
- ActiveBluetoothTransmit
- ActiveLightSensor

It's clear from the names of the processes what they would be responsible for. Lets take "ActiveMotor" this process needs to encapsulate data and algorithms that will respond to data coming in on its channel to move the motor forward or backward. A quick look at beginning structure of an ActiveMotor process is shown here.

```
import lejos.nxt.Motor;
import org.jcsp.nxt.lang.CSProcess;
import org.jcsp.nxt.lang.ChannelInputInt;

public class ActiveMotor implements CSProcess {
    private ChannelInputInt in;
    private Motor m;

    public ActiveMotor( Motor m, ChannelInputInt in ) {
        this.m = m;
        this.in = in;
    }
}
```

2.10.6.6.- Line follower robot with JCSP re & leJOS

2.10.6.6.1.1.- Introduction

The main focus developing software to control a line following robot is identifying the main processes and understanding the way in which the independent process communicate with each other. Software developers must know when to read from a communication channel and write to a channel. All processes implement the CSProcess interface so the classes all begin the same way and all contain a method called run. Start by defining processes that are necessary for the successful running of the robot. That is identify data that needs to be captured from a sensor or output on a port encapsulate this within a process. These are process like ActiveMotor, ActiveLightSensor and ActiveBinaryFilter. Next Define how each process interacts and work out how these processes are controlled. Then start defining a robot controller process to co-ordinate process interaction and program logic control flow. Software developers using JCSP re and LeJOS link the processes together by reading and writing to processes channels when they need to. Writing to a processes channel when a certain function has ended or reading from a processes channel is the responsibility of the software developer to know and understand this. I know that once my NXT has connected to another NXT via Bluetooth I can progress and start the light sensor calibration. In the code this is done by writing an integer, a 1 on the Bluetooth processes output channel. LeJOS allows software developers to write source code in Java to control the robot. JCSP re allows software developers use LeJOS and exploit the CSP concurrency model for parallel systems by constructing "active" or "live" processes. The software developer does not need to know the mathematics or the underlying CSP concurrency model they just need to know how to implement it. Building active processes of highly reduced concurrent software processes compliments LeJOS and LeJOS compliments the NXT Mindstorm robots. The end result should be a NXT

robot with much improved response and execution time using LeJOS and JCSP re to solve the line following robot problem with concurrent software.

Every process channel must have an "in" and "out". If you try to write to a channel that does not have an in channel then the program will crash. Building and encapsulating logic within processes cleanly separates functionality. All the user interface logic is handled through the ActiveUserInterface process. When that process wants to send a string to the NXT's LCD screen it writes a string to its output channel, which routes the string to the ActiveDisplay process. The ActiveDisplay process constantly waits and reads any data on its input channel. The only data that should be and will be coming into the ActiveDisplay process is strings. The logic contained within the ActiveDisplay process loops. It constantly waits for strings to come in on the input channel. When a string is available on a channel the processes logic reads it and using the correct class in LeJOS writes it to the LCD screen of the NXT.

The ActiveUserInterface process handles the interaction with the user. It will prompt the user with useful statements such as, "connecting", "Connected", "Black Level" and "White level". Once the program has executed a specific output statement an integer is sent from the ActiveUserInterface process on the output channel to the ActiveRobotController so that the program can progress on to the next task. For example this is what happens when the robot starts.

Robot starts
Robot outputs "Black Level"
Robot waits for user to position sensors on something black and **press the orange button** to read in the black value
Robot outputs "White Level"
Robot waits for user to position sensors on something white and **press the orange button** to read in the white value
Robot outputs "Speed 90"
If a user pushes the right arrow button on the NXT they will increase the speed
If the user just pushes the orange button the speed will be set
Robot displays "GO"
Robot waits for user to press the orange button to start the robot

Robot enters a running state and will respond to the black and white colors. For the Robot to work correctly a black line should be drawn on a white surface. As the program executes each step in the sequence above, integers and strings are being passed from process to process. The ActiveButton process sends integers to other processes based on the button pressed. Each button is assigned a different integer. The ActiveDisplay process receives strings from the ActiveUserInterface process. The ActiveRobotController sends integers to the ActiveMotor process and the ActiveUserInterface process. Everything is synchronized.

Once you have developed all the processes they all have to be linked together and all the processes run in parallel. This is done with in the main method. The software developer declares "channels" just as they would variables shown below.

One2OneChannelInt buttonPress = new One2OneChannelIntImpl();

Once the channels have been declared the processes need to be declared. Processes are declared the same way you would with an object. Instead of variables being passed as parameters you add "channels" so in the example below I have declared an ActiveDisplay process that has a channel called "display" as the parameter. Which end of the channel is it, is this channel sending data out of the process or into the process. In this case the ActiveDisplay process is receiving

"strings" from the ActiveUserInterface process so the channel is calling the .in() method. If we look at how the ActiveUserInterface is declared you will see the opposite .out() method on the same channel being called this links these two processes together. As you see the processes are being declared like objects. In the examples below the processes are called "screen" and "ui". These processes will be added to the parallel object par by calling a method called .addProcess().

ActiveDisplay screen = new ActiveDisplay (display.in());

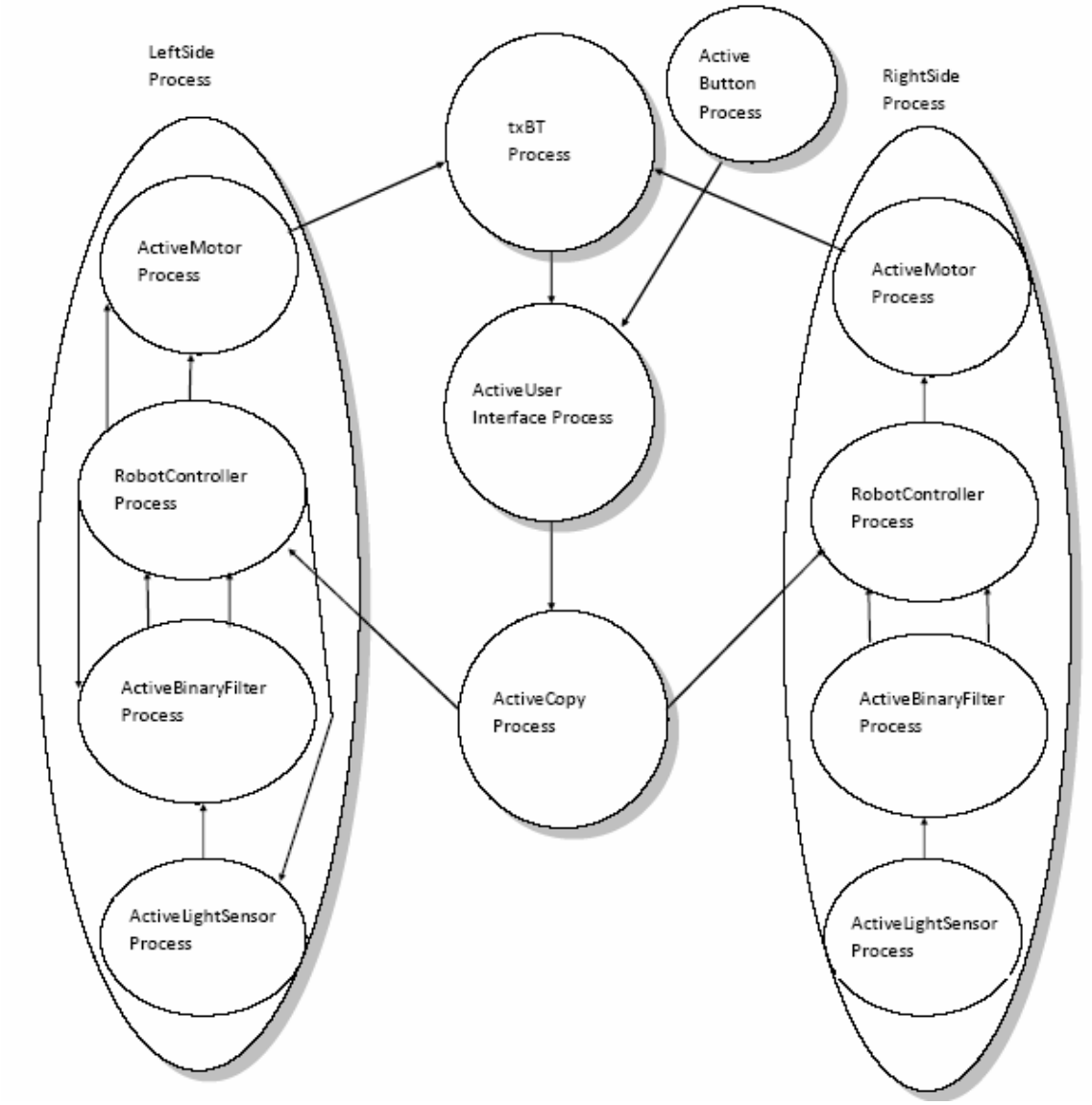
ActiveUserInterface ui = new ActiveUserInterface (keyPress.in(), display.out(),

buttonPress.out());

When all the channels and processes have been declared and linked together with the correct channels, all the processes need to be added to the parallel object and then the parallels run method needs to be called. This means that in your main method in your main class you have several processes and communication channels declared. The channels have been added as parameters to the correct processes and then all processes are added to the Parallel object and they are all executed concurrently.

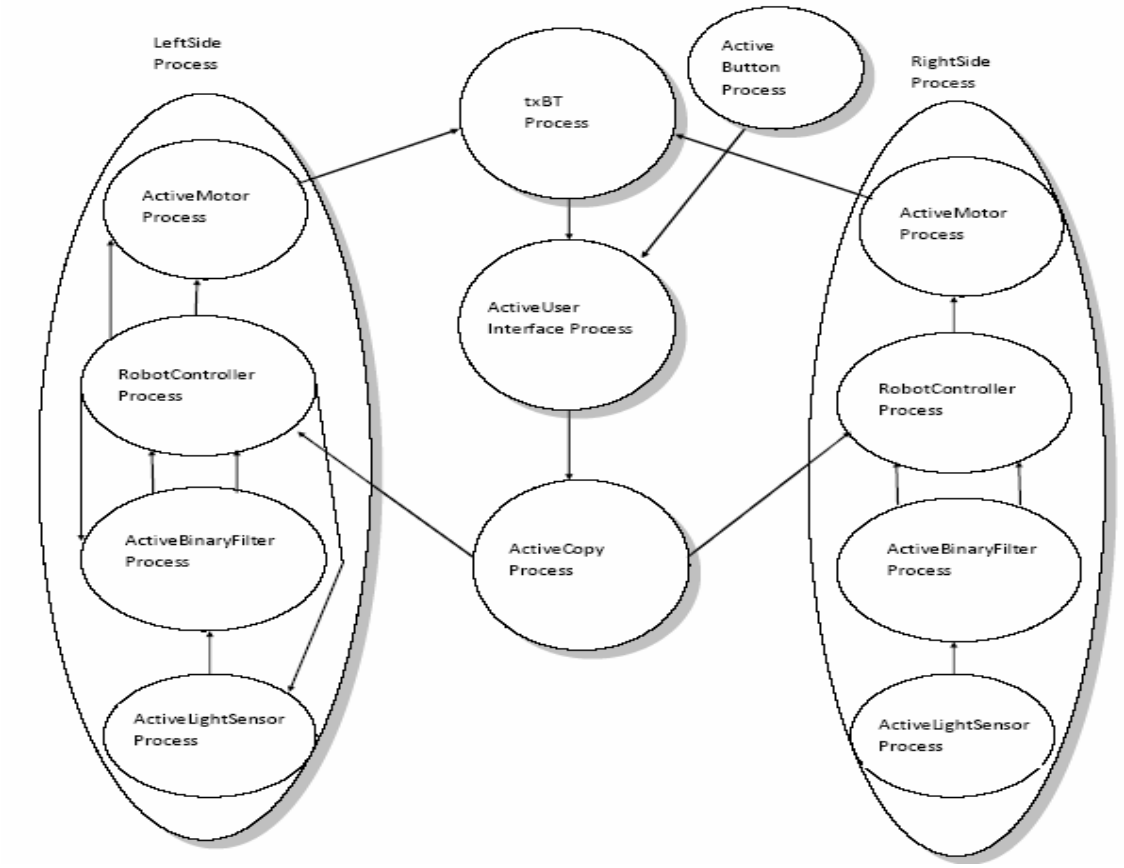
```
Parallel par = new Parallel();

par.addProcess(ui);
par.addProcess(screen);
par.run();
```

2.10.6.7.- Line follower high level overview



The left side process shows the processes and channels during start up and calibration and the right side processes shows the process in normal operation.

Robot One overview diagram



2.10.6.7.1.- Code Example

One example using Parallel Architectures is the following:

```
import lejos.nxt.Motor;
import lejos.nxt.SensorPort;

import org.jcsp.nxt.lang.Parallel;
import org.jcsp.nxt.lang.*;
import org.jcsp.nxt.plugnplay.TwoCopy;
import org.jcsp.nxt.io.*;

public class One {

      public static void main( String[] args ) {

            One2OneChannelInt         buttonPress         =         new
One2OneChannelIntImpl();
            One2OneChannelInt keyPress = new One2OneChannelIntImpl();
            One2OneChannel display = new One2OneChannelImpl();
            One2OneChannelInt leftCopy = new One2OneChannelIntImpl();
            One2OneChannelInt rightCopy = new One2OneChannelIntImpl();
            One2OneChannelInt outBT = new One2OneChannelIntImpl();
            One2OneChannelInt         inFromRight         =         new
One2OneChannelIntImpl();
            One2OneChannelInt inFromLeft = new One2OneChannelIntImpl();

            Parallel par = new Parallel();
            //Lcpbtr btR = new Lcpbtr();
            //RobotSide   rightSide   =   new   RobotSide(SensorPort.S2,
Motor.A, rightCopy.in(), 1,inFromRight);
```

```
            RobotSide   rightSide   =   new   RobotSide(SensorPort.S2,
Motor.A, rightCopy.in(),1);
            //RobotSide   leftSide   =   new   RobotSide(SensorPort.S3,
Motor.B, leftCopy.in(), 2, inFromLeft);
            RobotSide leftSide = new RobotSide(SensorPort.S3,  Motor.B,
leftCopy.in(), 2);

            ActiveButtons buttons = new ActiveButtons (keyPress.out());
            ActiveUserInterface    ui    =    new    ActiveUserInterface
(keyPress.in(), display.out(), buttonPress.out(), outBT.in());
            ActiveDisplay screen = new ActiveDisplay (display.in());
            TwoCopy    copy    =    new    TwoCopy    (buttonPress.in(),
leftCopy.out(), rightCopy.out());

            par.addProcess(rightSide);
            par.addProcess(leftSide);
            par.addProcess(copy);
            par.addProcess(ui);
            par.addProcess(screen);
            par.addProcess(buttons);
            par.run();


      }
}



import org.jcsp.nxt.lang.*;
import org.jcsp.nxt.io.ActiveLightSensor;

public class ActiveRobotController implements CSProcess {
      AltingChannelInputInt buttonPress;
      ChannelInputInt lightLevel;
      AltingChannelInputInt filterHigh;
      AltingChannelInputInt filterLow;
      ChannelOutputInt lightConfig;
      ChannelOutputInt filterConfig;
      ChannelOutputInt motorSpeed;
      int id;
      long interval;
      private ChannelOutputInt BTChannel;

      public ActiveRobotController( AltingChannelInputInt buttonPress,
                                    ChannelInputInt lightLevel,
                                    AltingChannelInputInt
filterHigh,
                                    AltingChannelInputInt
filterLow,
                                    ChannelOutputInt lightConfig,
                                    ChannelOutputInt filterConfig,
                                    ChannelOutputInt motorSpeed,
                                    int     id,     ChannelOutputInt
BTChannel) {
            this.buttonPress = buttonPress;
            this.lightLevel = lightLevel;
            this.filterHigh = filterHigh;
            this.filterLow = filterLow;
            this.lightConfig = lightConfig;
            this.filterConfig = filterConfig;
            this.motorSpeed = motorSpeed;
```

```
            this.id = id;
            this.BTChannel = BTChannel;
      }

      public void run(){
            int rotate = 0;
            int noRotate = 0;
            int altIndex = -1;
            buttonPress.read(); // configure black level
            //Display.print("B");
            //Display.print(id);
            lightConfig.write(ActiveLightSensor.LEVEL);
            //Display.print("C");
            //Display.print(id);
            int low = lightLevel.read();
            //Display.print("L");
            //Display.print(low);
            buttonPress.read(); // configure white level
            //Display.print("W");
            //Display.print(id);
            lightConfig.write(ActiveLightSensor.LEVEL);
            //Display.print("C");
            //Display.print(id);
            int high = lightLevel.read();
            //Display.print("H");
            //Display.print(high);
            int mid = low + ((high - low)/2);
            //Display.print("M");
            //Display.print(mid);
            filterConfig.write( mid ); // write mid point value to the
filter
            //Display.print("F");
            //Display.print(id);
            lightConfig.write(ActiveLightSensor.ACTIVATE);
            //Display.print("A");
            Alternative a = new Alternative( new Guard[] { filterHigh,
filterLow, buttonPress } );

            while (true) {
                  rotate = buttonPress.read(); // read speed of wheel
rotation
                  //Display.println( rotate );
                  buttonPress.read(); // the go signal
                  boolean going = true;
                  while (going) {
                        altIndex = a.select();
                        if (altIndex == 0) {
                              filterHigh.read();
                              motorSpeed.write(rotate);
                              BTChannel.write(1);
                        }
                        if (altIndex == 1) {
                              filterLow.read();
                              motorSpeed.write(noRotate);
                              BTChannel.write(0);
                        }
                        if (altIndex == 2) {
                              buttonPress.read(); // the stop signal
                              going = false;
                        }
                  }
```

```
            }
      }
}



import org.jcsp.nxt.lang.*;
import org.jcsp.nxt.io.ActiveButtons;

public class ActiveUserInterface implements CSProcess {
      ChannelInputInt inFromAB;
      ChannelOutput outToAD;
      ChannelOutputInt out;
      private ChannelInputInt inFromBT;


      public       ActiveUserInterface(ChannelInputInt       inFromAB,
ChannelOutput outToAD, ChannelOutputInt out, ChannelInputInt inFromBT)
{
            this.inFromAB = inFromAB;
            this.outToAD = outToAD;
            this.out = out;
            this.inFromBT = inFromBT;
      }


      public void run() {
            int [] speeds = {90, 120, 150, 180, 210, 240, 270, 300,
330, 360, 390, 420, 450, 480, 510, 540, 570 };
            outToAD.write("running...");
            inFromBT.read();
            outToAD.write("Black Level");
            outToAD.write("\n");
            inFromAB.read();
            out.write(1);
            outToAD.write("White Level");
            outToAD.write("\n");
            inFromAB.read();
            out.write(1);
            int p = 0;
            int s = speeds.length;
            while (true) {
                  outToAD.write("Speed: ");
                  outToAD.write(new Integer(speeds[p]) );
                  outToAD.write("\n");
                  int key = inFromAB.read();
                  while ( key != ActiveButtons.BUTTON_ENTER ) {
                        if     ((p     <     s)     &&     (key     ==
ActiveButtons.BUTTON_RIGHT) ) p = p + 1;
                        if     ((p     >     0)     &&     (key     ==
ActiveButtons.BUTTON_LEFT) ) p = p - 1;
                        key = inFromAB.read();
                  }
                  outToAD.write("New Speed: ");
                  outToAD.write(new Integer(speeds[p]) );
                  outToAD.write("\n");
                  out.write(speeds[p]);
                  outToAD.write("GO");
                  outToAD.write("\n");
                  inFromAB.read();
                  out.write(1);
```

```
                outToAD.write("STOP");
                outToAD.write("\n");
                inFromAB.read();
                out.write(-1);
                outToAD.write("\f");
            }
        }
}



import lejos.nxt.Motor;
import lejos.nxt.SensorPort;

import org.jcsp.nxt.filters.BinaryFilter;
import org.jcsp.nxt.io.*;
import org.jcsp.nxt.lang.*;


public class RobotSide implements CSProcess {
      ChannelInputInt buttonPress;
      Motor m;
      SensorPort s;
      int id;
      One2OneChannelInt BTChannel;

      Parallel par = new Parallel();

      public RobotSide( SensorPort s, Motor m, AltingChannelInputInt
buttonPress, int id) {
      //public RobotSide( SensorPort s, Motor m, AltingChannelInputInt
buttonPress, int id, One2OneChannelInt BTChannel) {

            this.id = id;

            One2OneChannelInt configL = new One2OneChannelIntImpl();
            One2OneChannelInt outL = new One2OneChannelIntImpl();
            One2OneChannelInt levelL = new One2OneChannelIntImpl();
            One2OneChannelInt         filterConfig         =         new
One2OneChannelIntImpl();
            One2OneChannelInt filterOutL = new One2OneChannelIntImpl();
            One2OneChannelInt filterOutH = new One2OneChannelIntImpl();
            One2OneChannelInt motorSpeed = new One2OneChannelIntImpl();
            par.addProcess(    new    ActiveLightSensor    (    s,    25,
configL.in(), levelL.out(), outL.out()  ) );
            par.addProcess(   new   BinaryFilter   (   filterOutL.out(),
filterOutH.out(), outL.in(), filterConfig.in()));
            par.addProcess( new ActiveRobotController( buttonPress,
                                                            levelL.in(),

filterOutH.in(),

filterOutL.in(),

configL.out(),

filterConfig.out(),

motorSpeed.out(), id, BTChannel.out()));
            par.addProcess( new ActiveMotor( m, motorSpeed.in()) );
```

```
	}

	public void run() {
		par.run();

	}
}
```

2.11.- Software Engineering

2.11.1.- Introduction

Not Available

2.11.2.- UML

Not Available

2.11.3.- Design your robot with UML

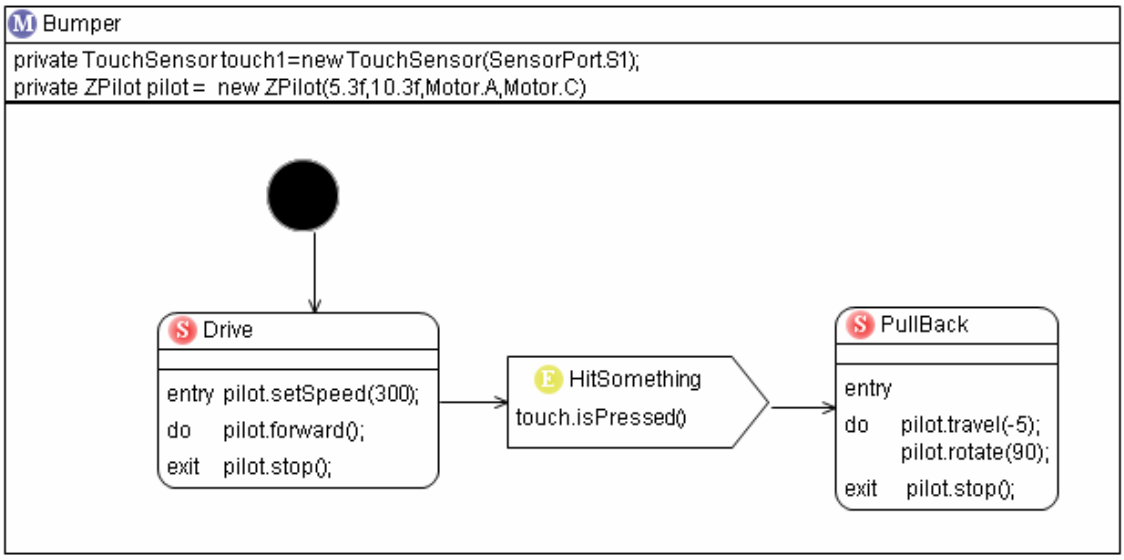
Not Available

3.- Tools

3.1.- LeJOS statemachine developer toolkit

3.1.1.- Introduction

LeJOS Statemachine development toolkit is a visual modelling of leJOS applications based on statemachines.



This toolkit use Eclipse IDE and some plugins. In this article we explain how to install the toolkit in your computer.

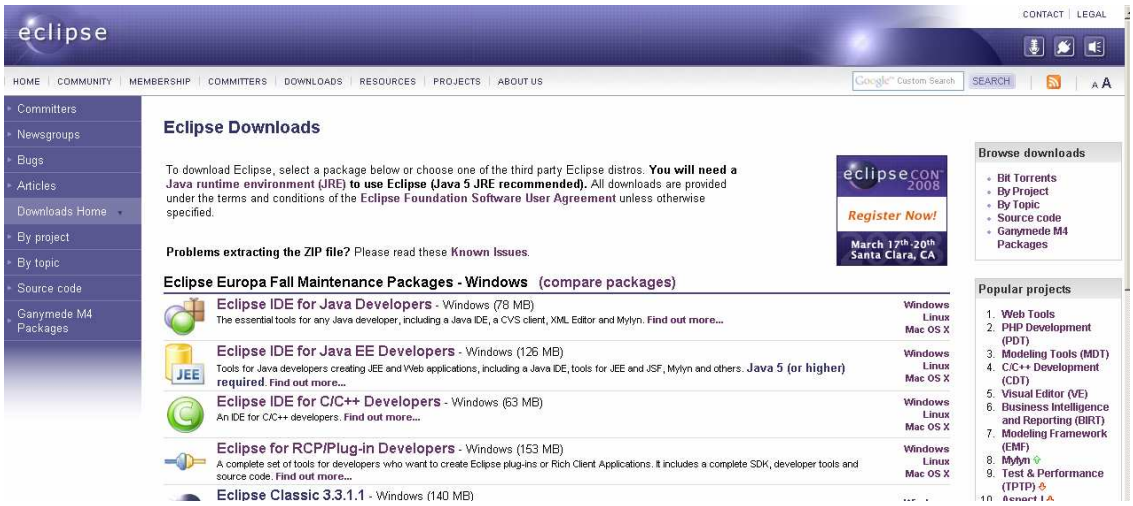
3.1.2.- Installing LeJOS statemachine developer toolkit

The installation is very easy the steps are:

1. Install Eclipse Europe 3.3
2. Install GMF
3. Install OWA
4. Install LeJOS statemachine developer toolkit

3.1.2.1.- Eclipse

Eclipse is an open source community development platform designed to manage software across the lifecycle. Download Eclipse Europa 3.3 Classic from eclipse's website. Use the following URL: http://www.eclipse.org/downloads/



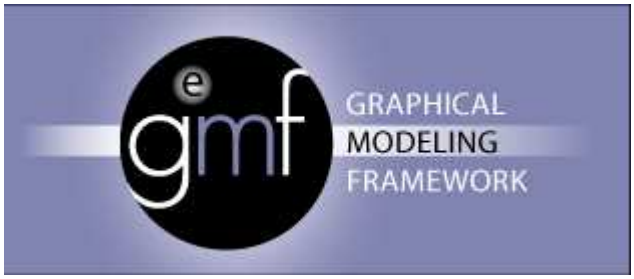
Once you have installed the IDE in your computer it is necessary to install the following features on eclipse:

- GMF, Graphics Modeling Framework
- OWA

Finally we will install leJOS statemachine development toolkit feature on Eclipse IDE

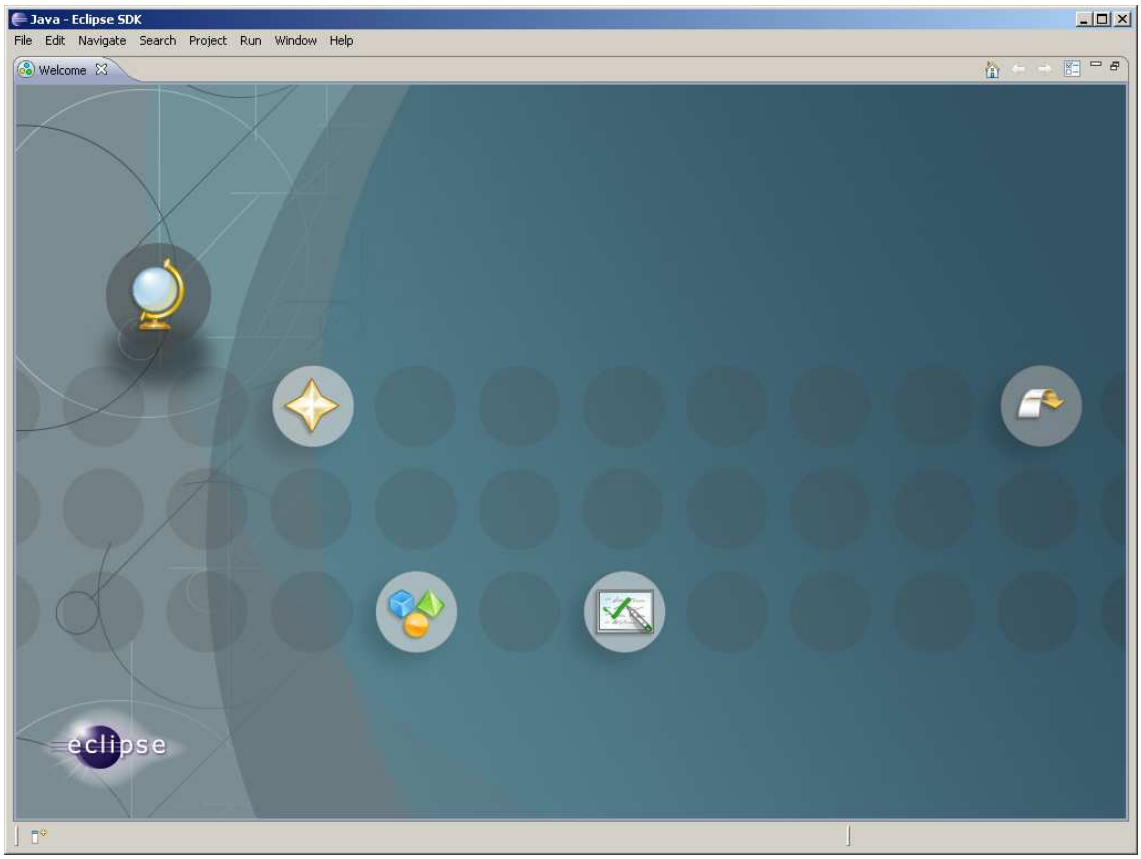
3.1.2.2.- GMF, Graphics Modelling Framework

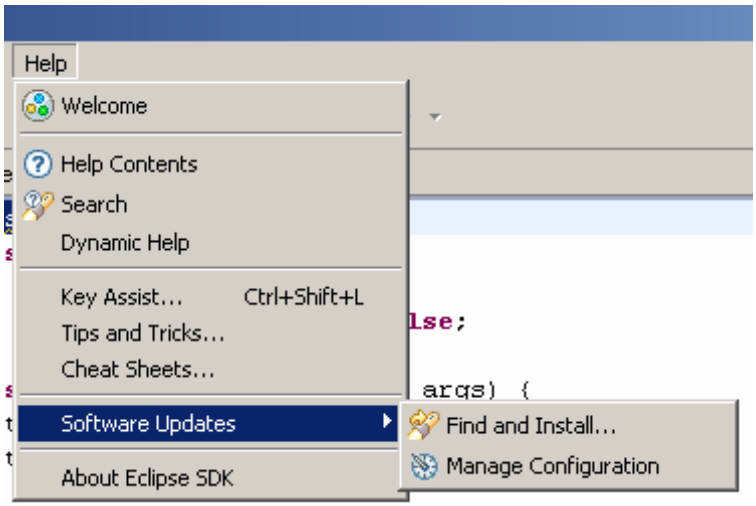
Eclipse Graphical Modelling Framework (GMF) provides a generative component and runtime infrastructure for developing graphical editors based on EMF and GEF. The project aims to provide these components, in addition to exemplary tools for select domain models which illustrate its capabilities



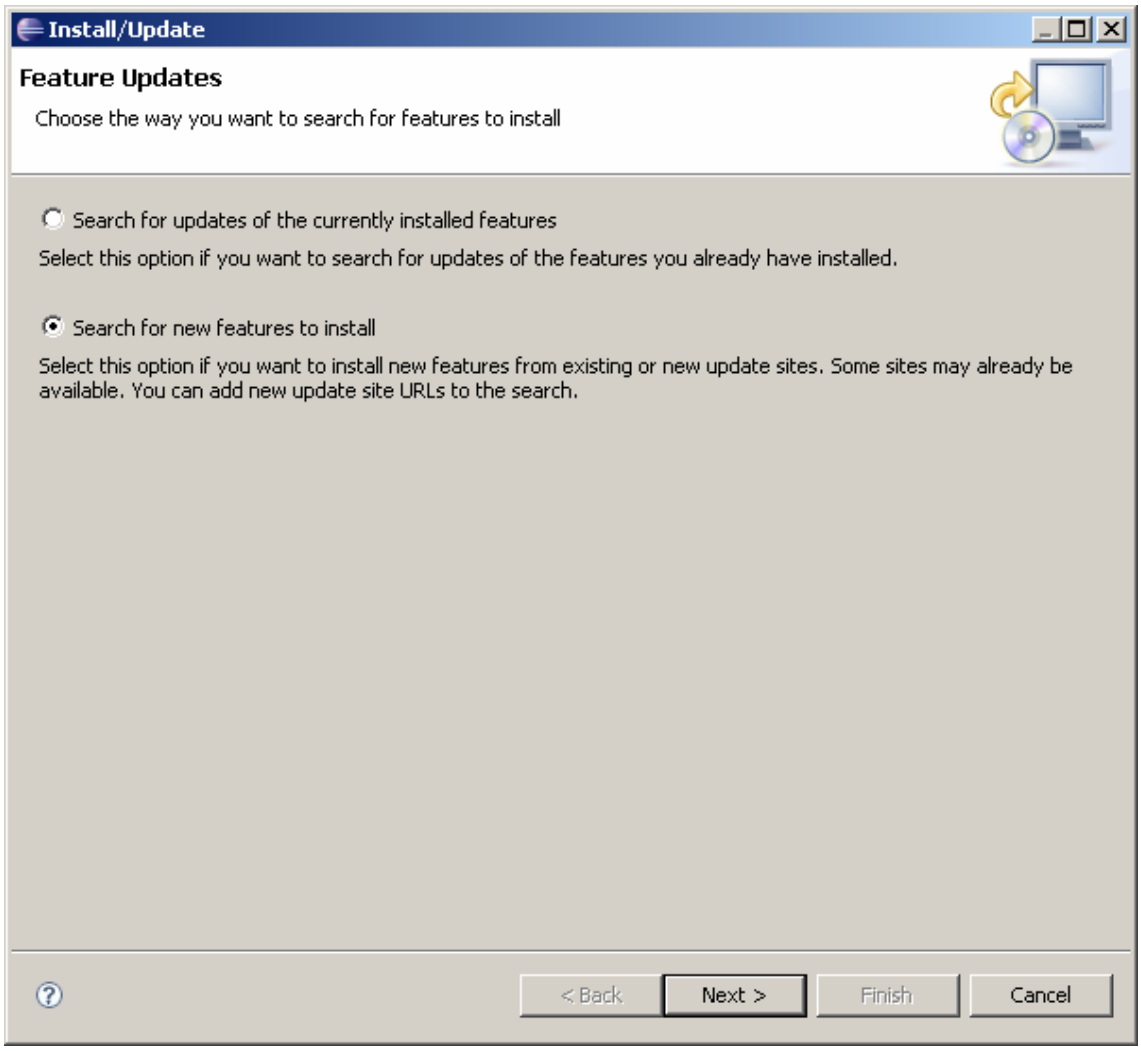
http://www.eclipse.org/gmf/
http://download.eclipse.org/modeling/gmf/downloads/index.php

The easiest way to install GMF is using the option on eclipse Find and install.

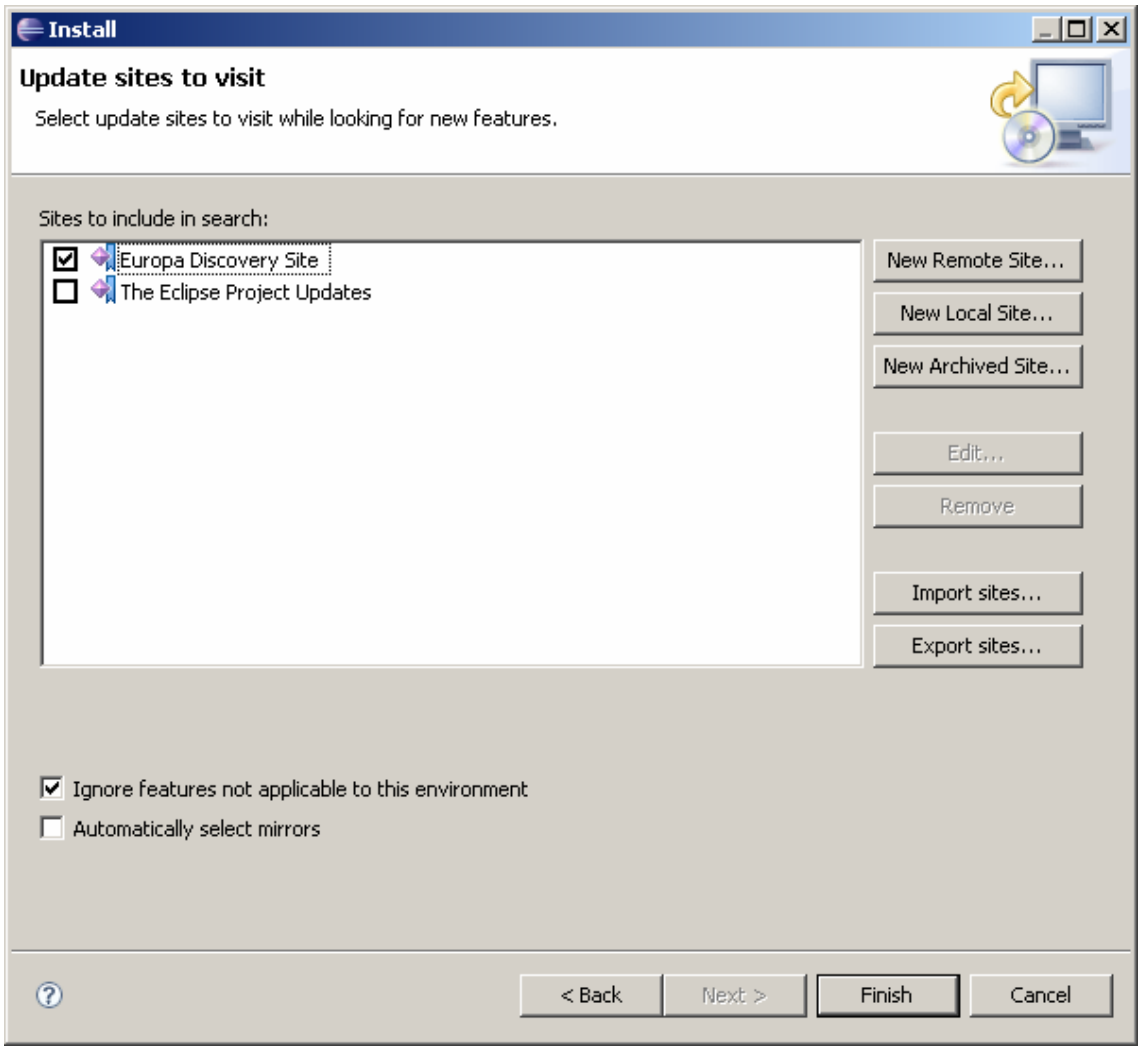




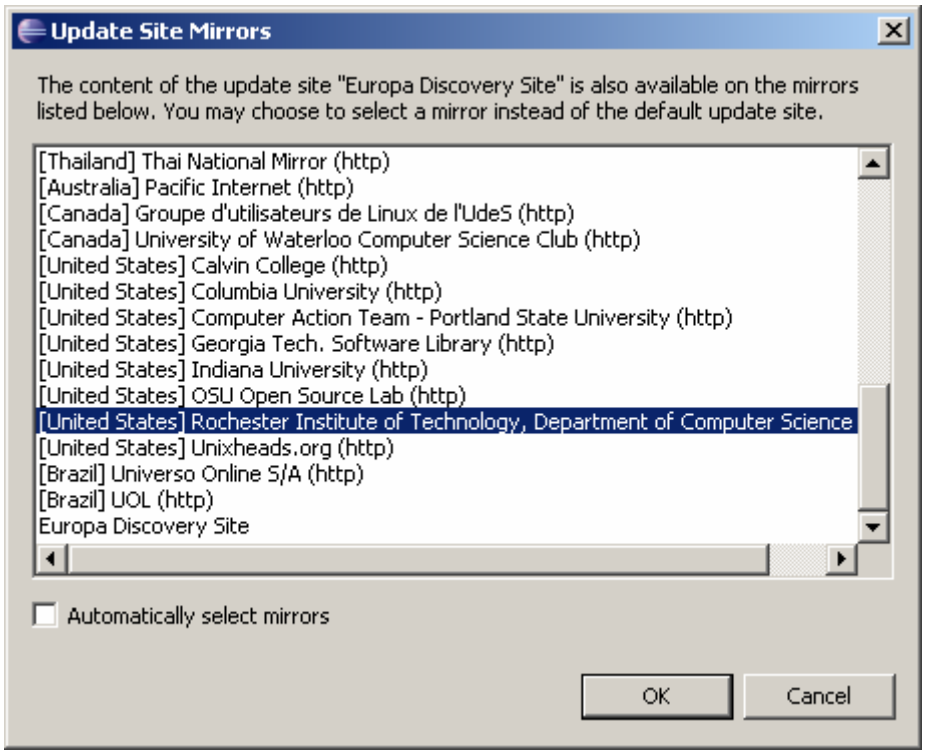
When you click in the option then, Eclipse will init an assistant to install new features.



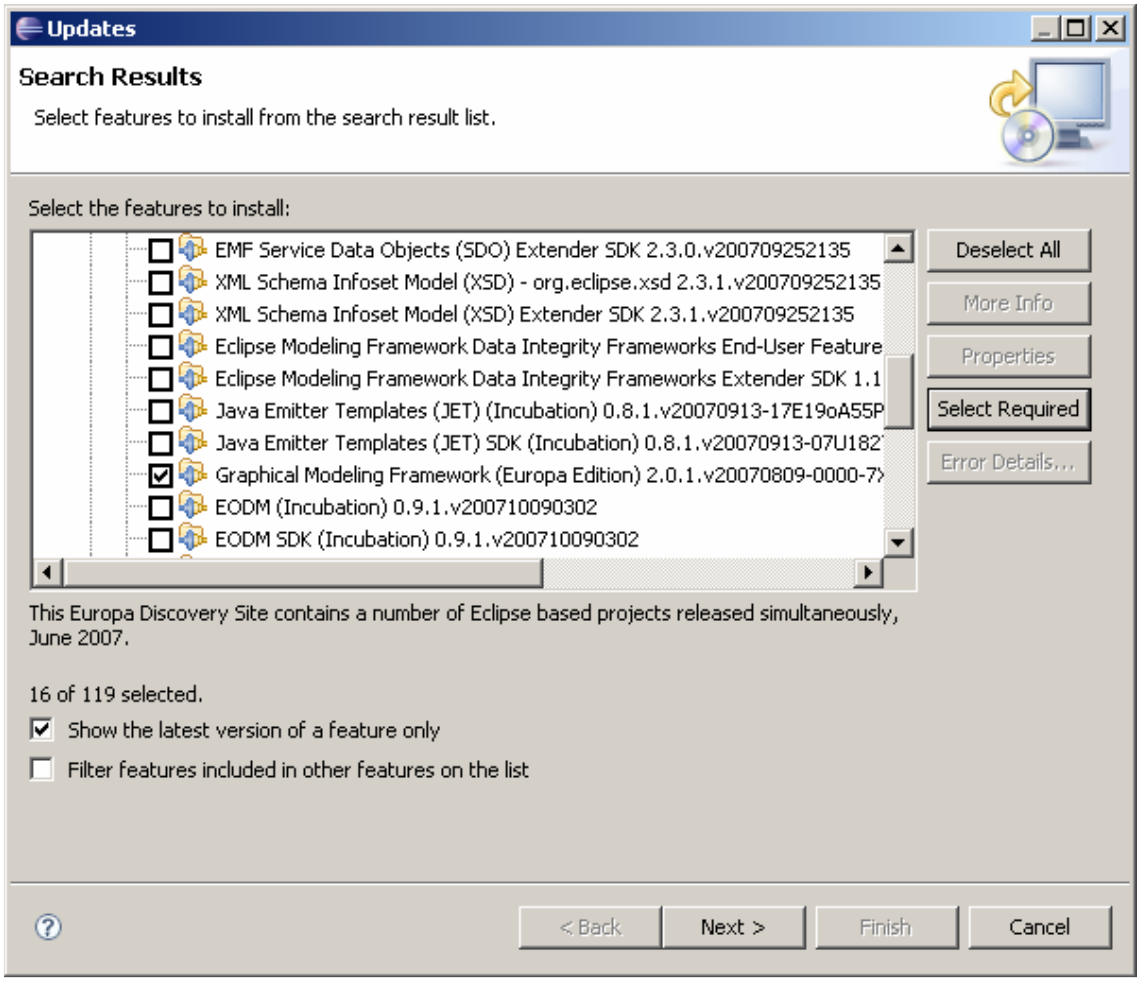
Select Search for new features to install and click in the button next



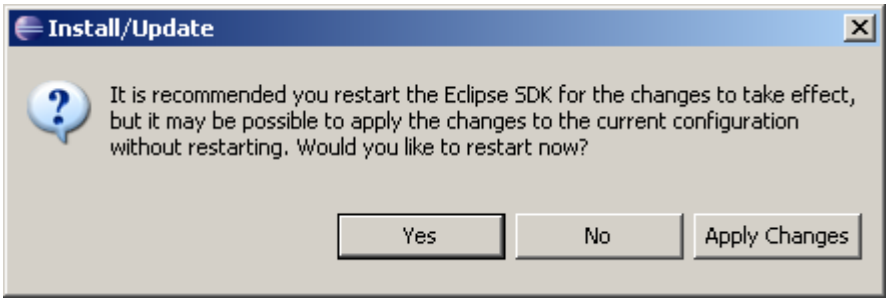
Select Europe Discovery Site and click in the button finish showing a list of server to search latest release of GMF



Select your favourite server and select the feature:



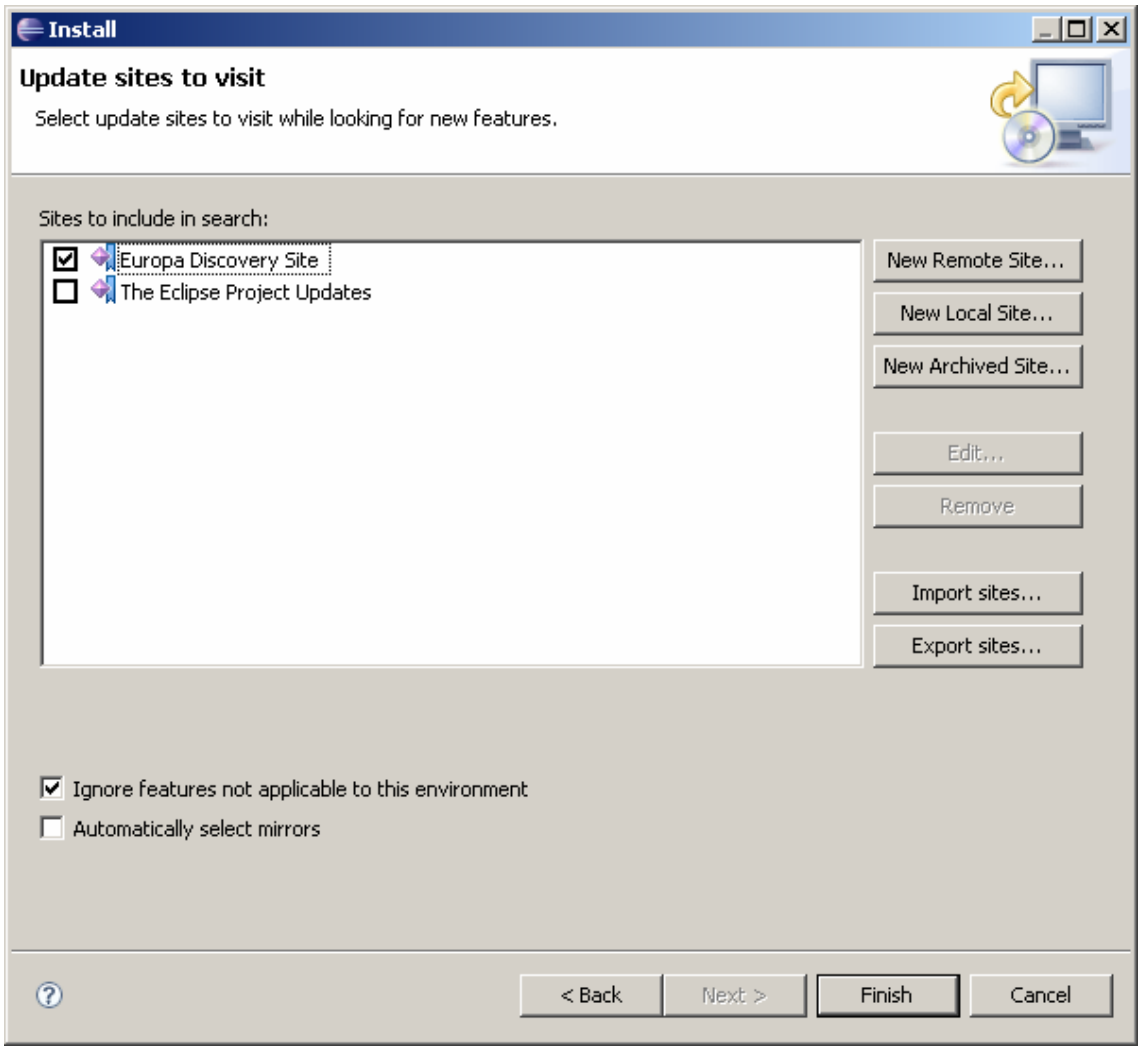
When you finish the installation process, it is necessary to restart the IDE.



3.1.2.3.- OAW, Open Architecture Ware

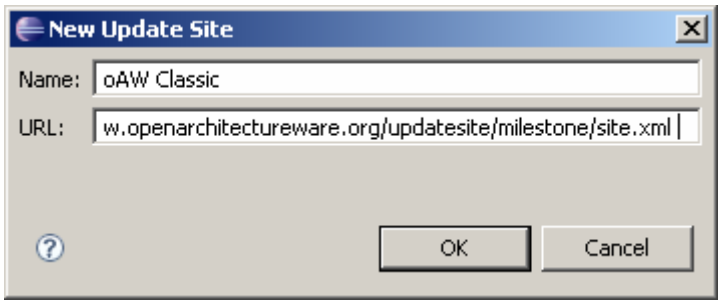
Open Architecture Ware (oAW) is a suite of tools to assist model-driven software development. It is a "tool for building MDSD/MDA tools". OAW is built upon a modular MDSD generator framework and is implemented in Java.

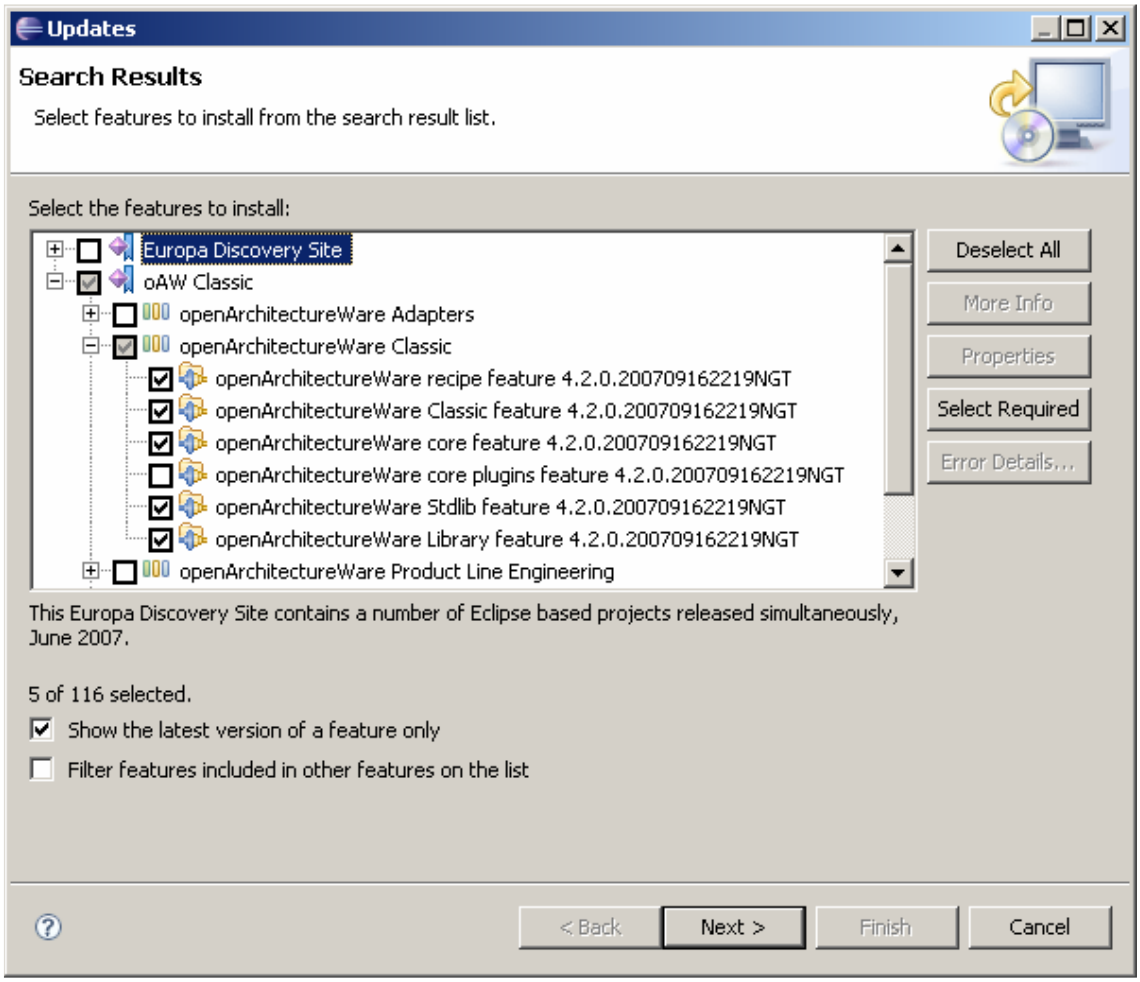
In this case, OWA should be installed with a different way in relation to GMF.



When you are in this window, click in "New remote site" button and edit a empty window with the following URL:

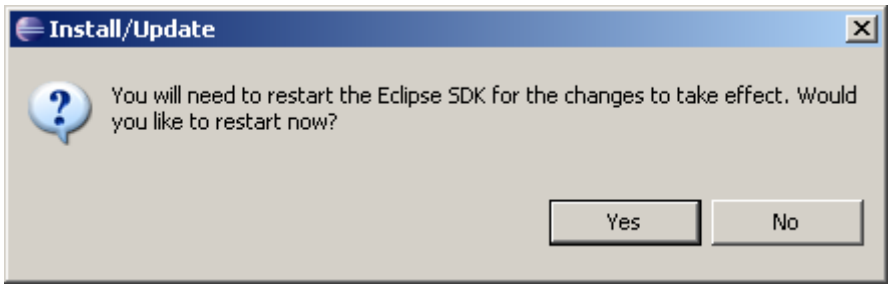
http://www.openarchitectureware.org/updatesite/milestone/site.xml





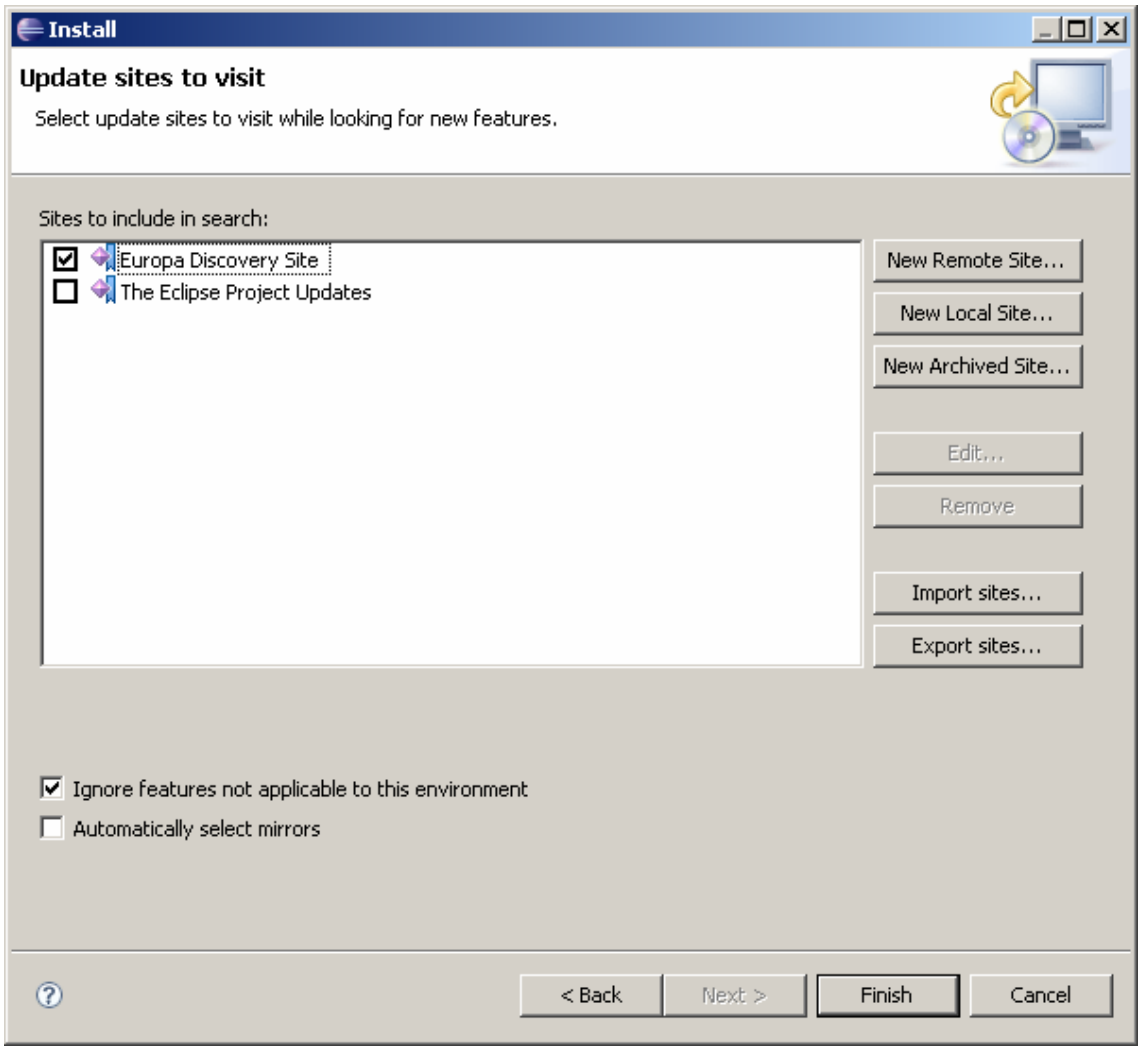
Expand the node and install the "oAW Classic feature" (you need to expand the oAW Classic one step further, to see the oAW Classic feature).When you click on select required, there will be two more features selected.

When you finish, restart Eclipse.



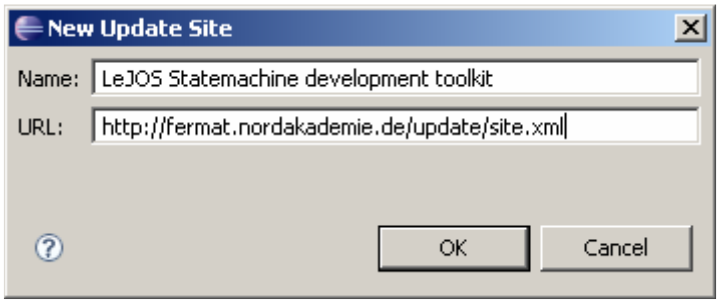
3.1.2.4.- LeJOS Statemachine development toolkit

Finally when you have installed the prerequisites, it is the moment to install the toolkit.

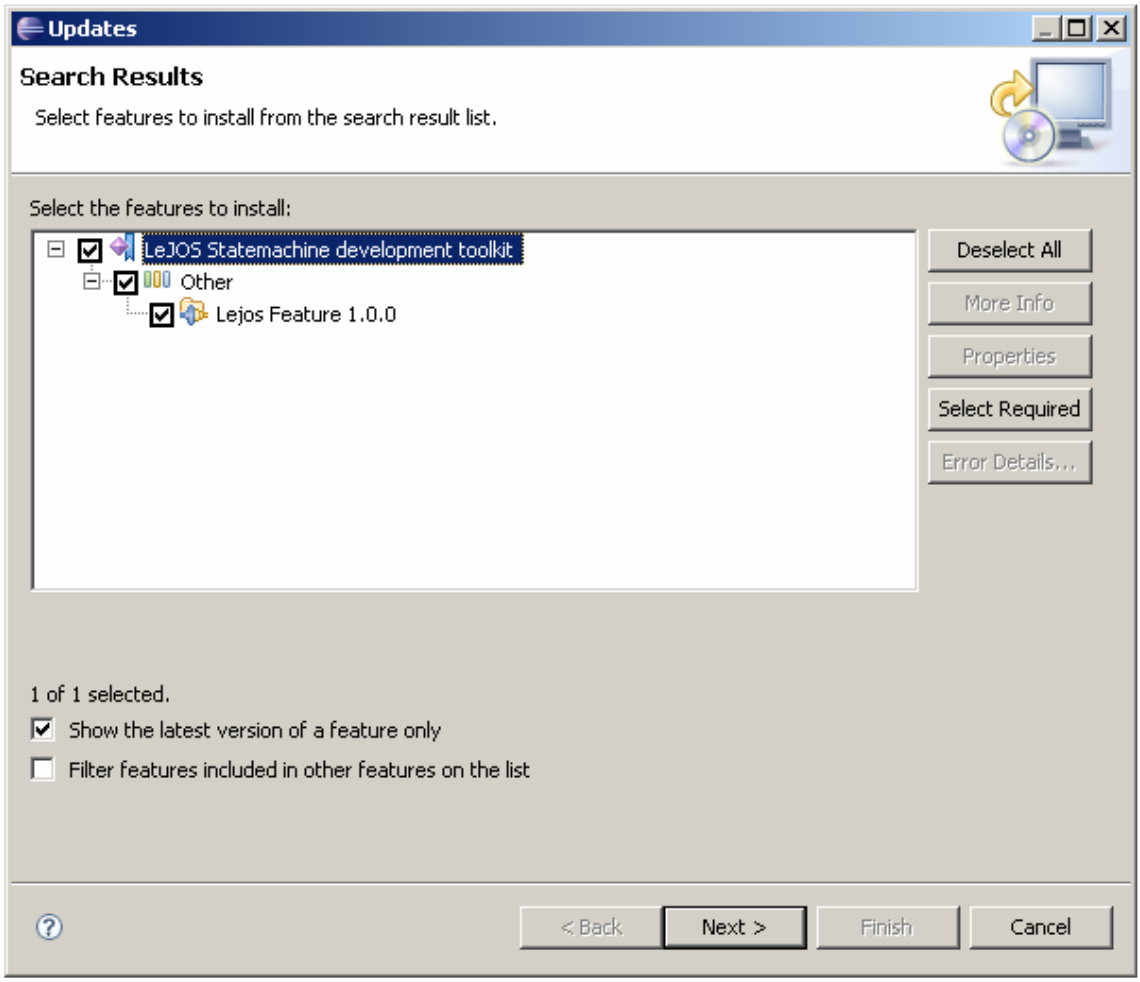


When you are in this window, click in "New remote site" button and edit an empty window with the following URL:

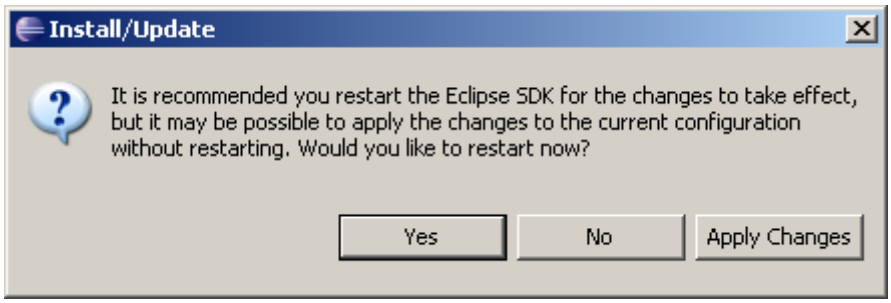
http://fermat.nordakademie.de/update/site.xml



When you confirm the data, you will see all features to install:



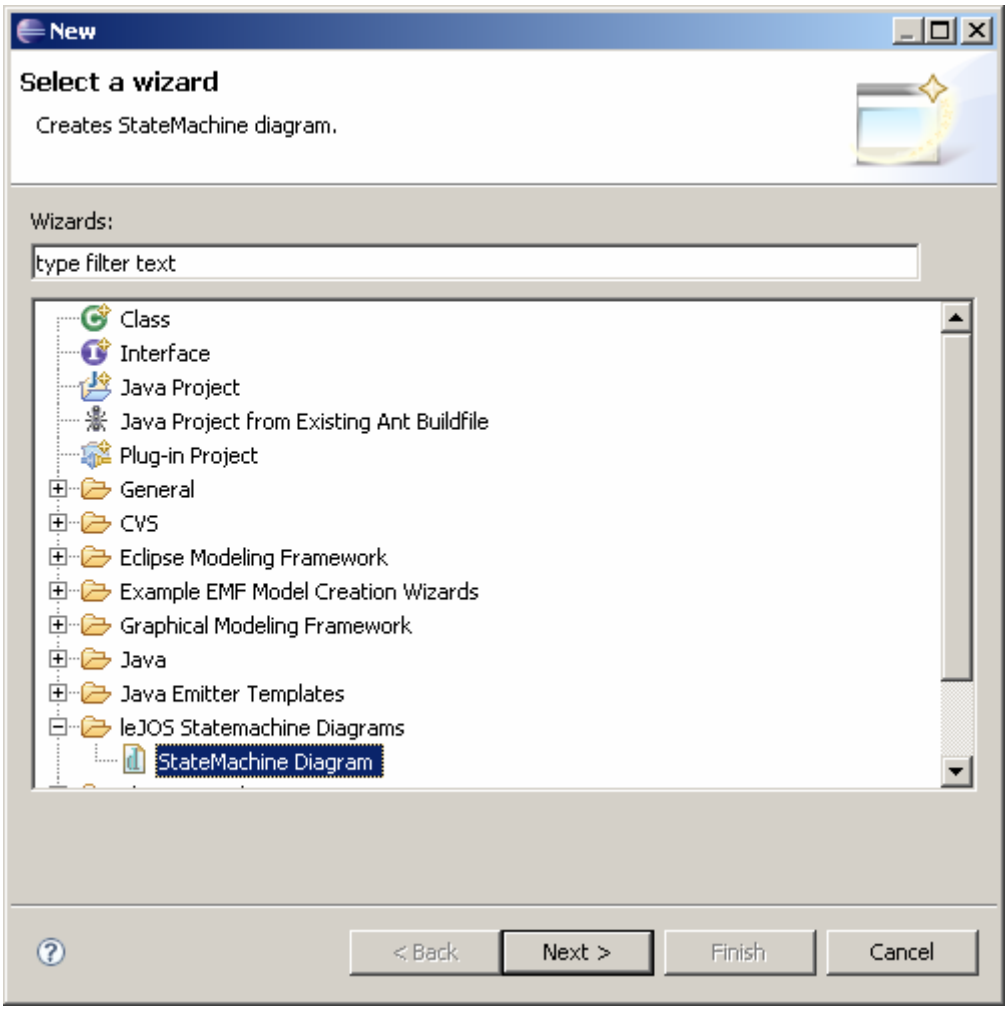
Click in "Next" button to install the toolkit. When you finish, restart the IDE.



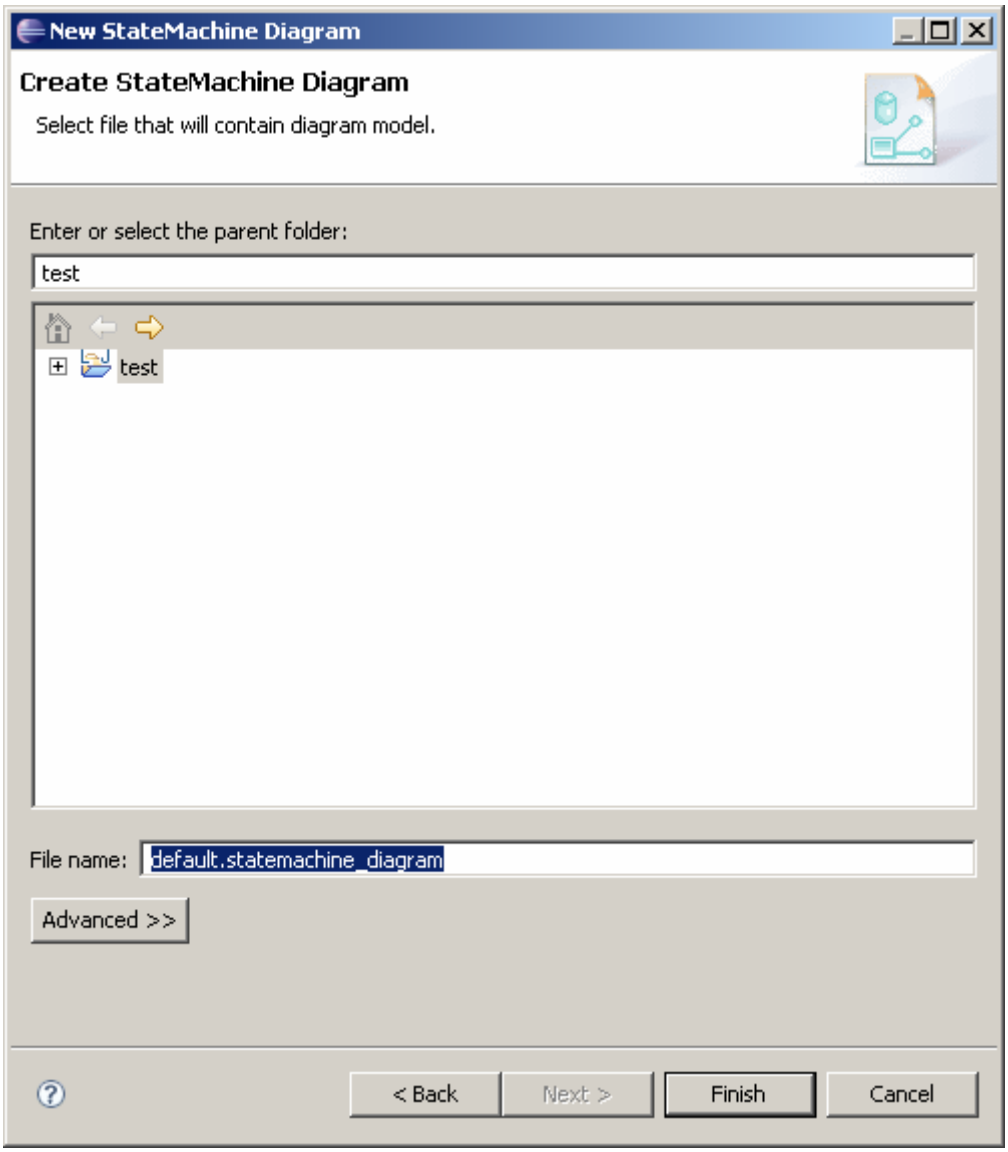
3.1.3.- Creating the first project with the toolkit

3.1.3.1.- Create a new project

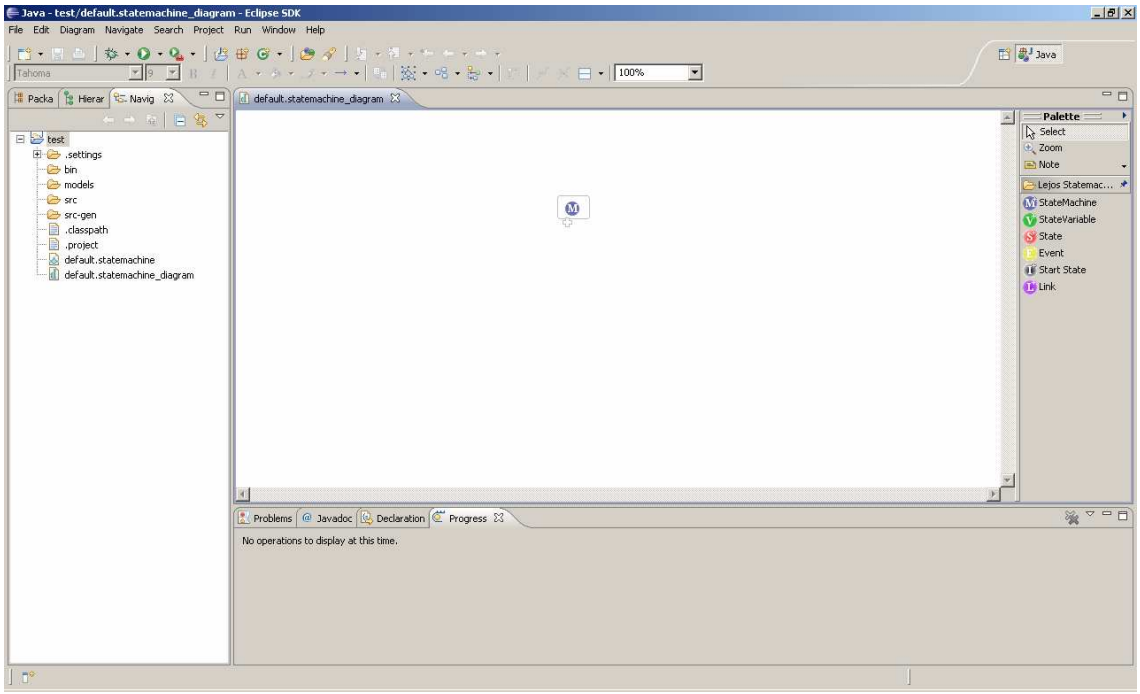
Init Eclipse and create a new project, select the option LeJOS statemachine project.



When you have created the project, it is necessary to create a Statemachine diagram:



When you finish, you will see the following interface.



NOTE: We will update this section as soon as possible.

3.1.4.- Videos

See the following videos using the toolkit.

http://de.youtube.com/watch?v=5_vjuUL8f1w
http://de.youtube.com/watch?v=HiP9AQ7WF8c
http://de.youtube.com/watch?v=YUdxvhgILAo

4.- FAQ

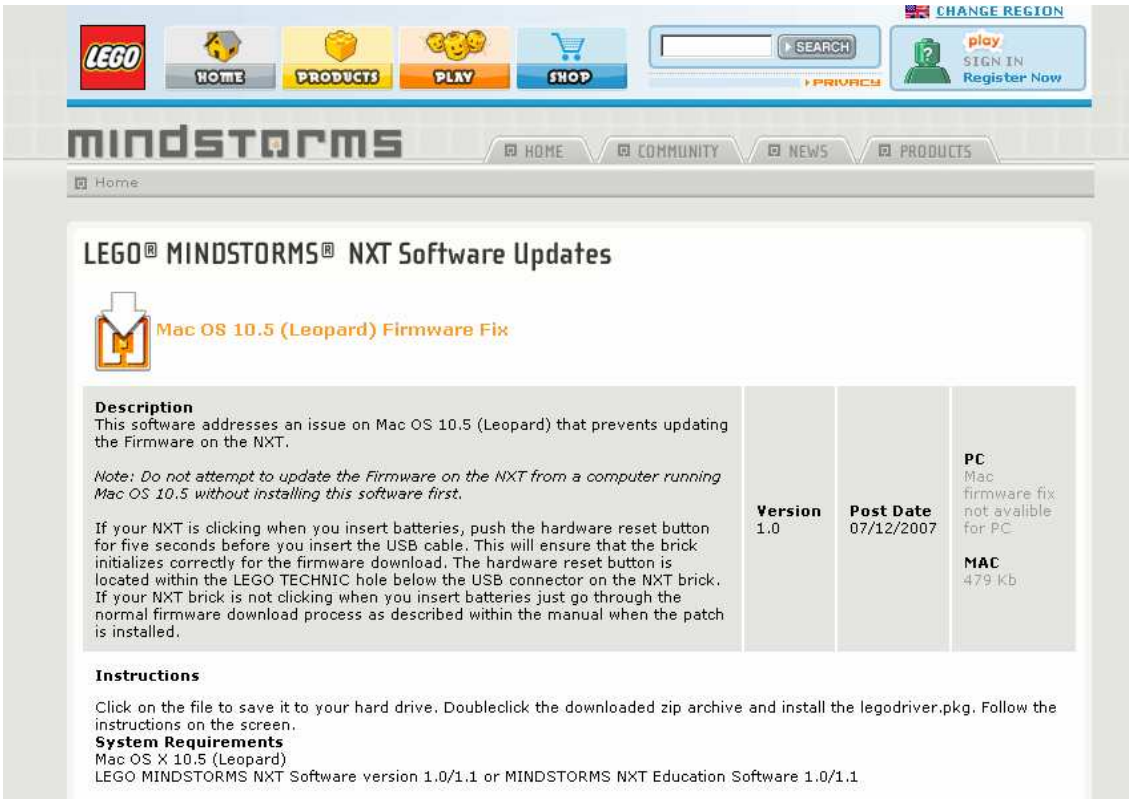
4.1.- How to reinstall Lego Firmware

4.1.1.- Introduction

LeJOS is an excellent platform to develop software for NXT Lego Mindstorm but it is not the unique solutions.
If you read the section NXT Programming Software written by Steve Hassenplug in http://www.teamhassenplug.org/NXT/NXTSoftware.html then you will notice that exist several options to develop software in the NXT brick. If you have installed LeJOS firmware and you decided to reinstall Lego firmware, read this section to know how to do.

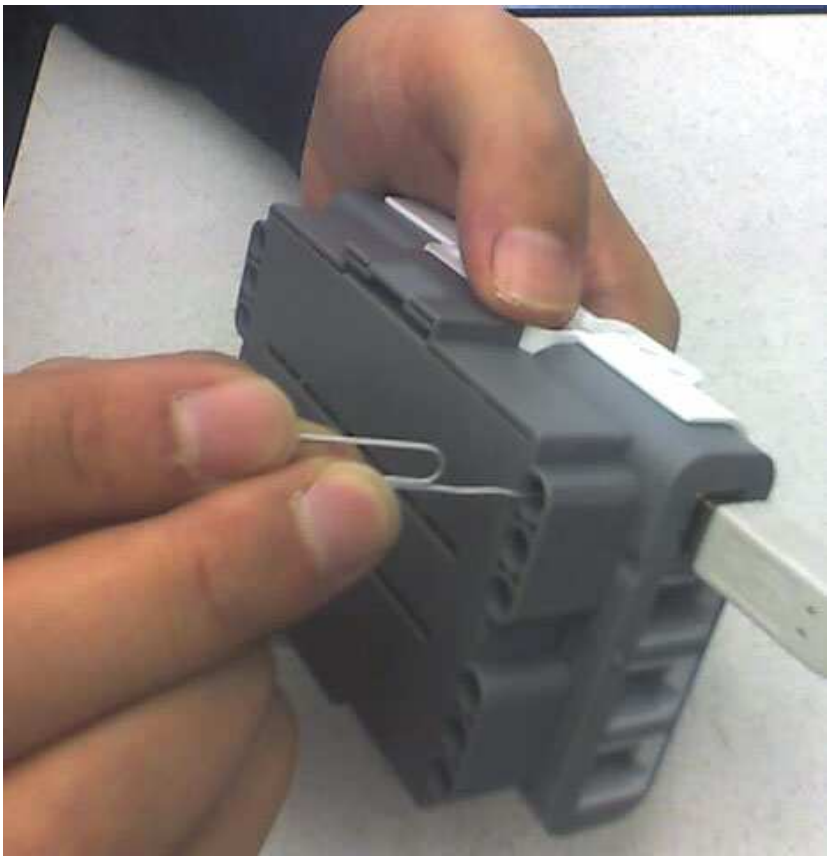
4.1.2.- Download latest Lego firmware

To reinstall Lego Firmware is necessary to have latest firmware in your computer. Visit http://mindstorms.lego.com/support/updates/ to download the firmware.

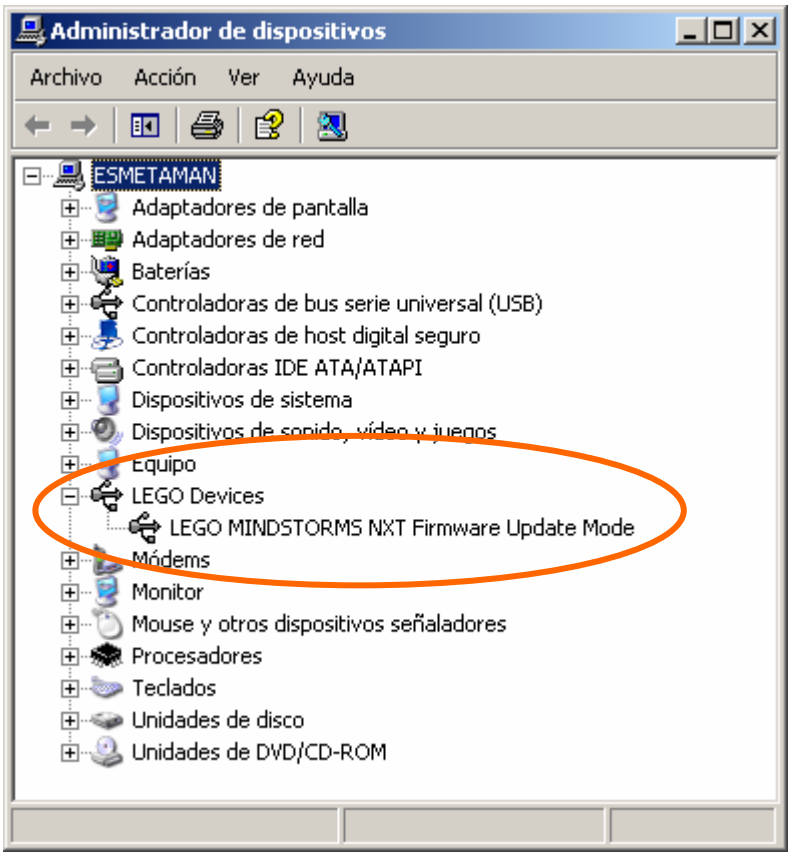


4.1.3.- Set your NXT brick in update mode

Once you have stored latest firmware, it is necessary to set your NXT brick in Update mode. To update the mode in your NXT brick then you have to push reset button. To find that button see at the back of the NXT, upper left corner and push it for more than 5 seconds then you will hear an audibly sound.



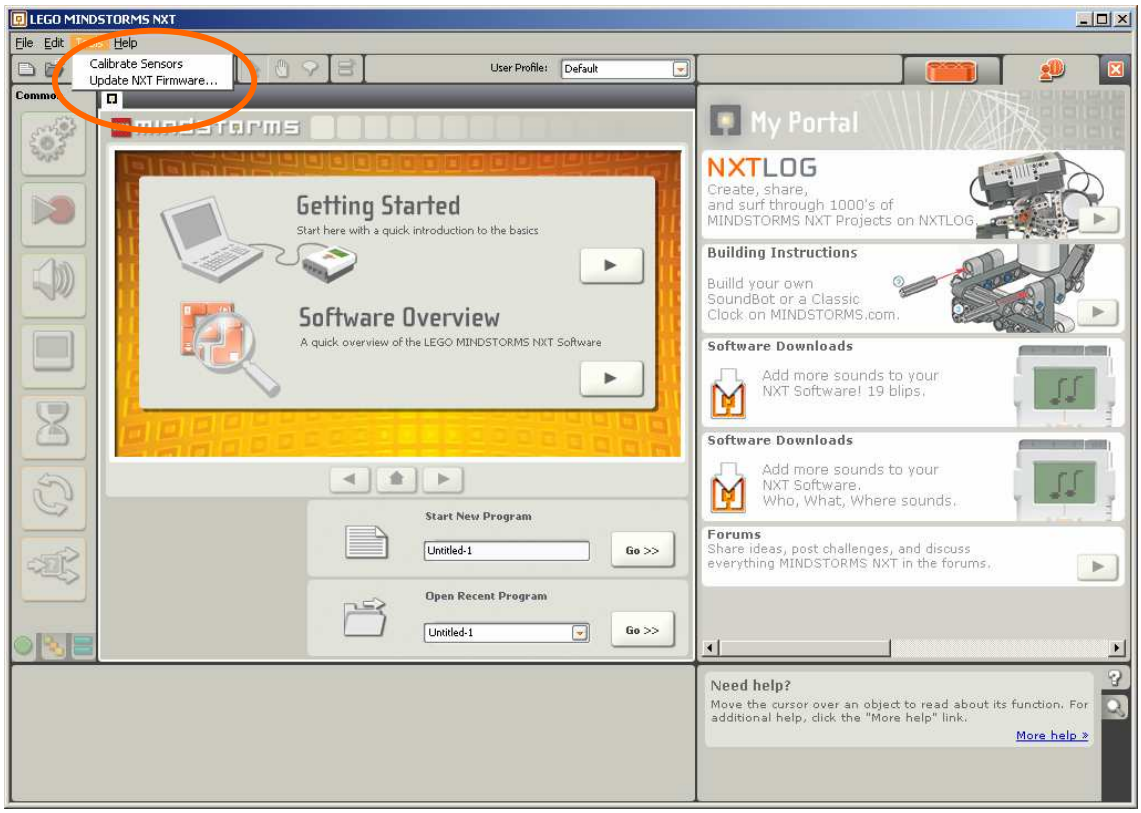
If you want to be sure, check your Lego Devices connected with your computer then you will see:



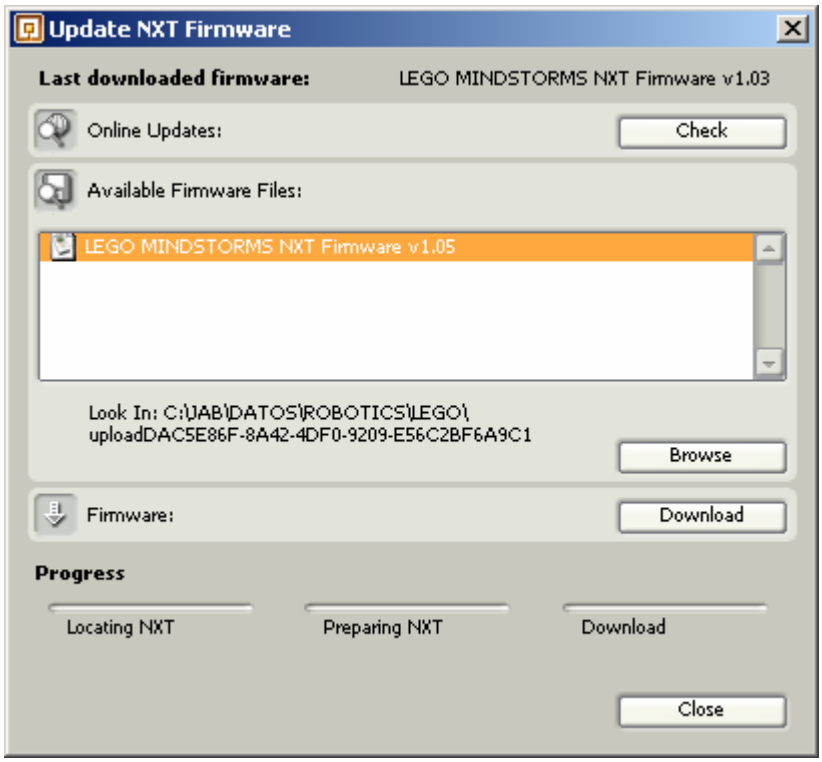
4.1.4.- Reinstall Lego firmware

Use Lego Software that you received with your NXT Kit to download Lego Firmware.

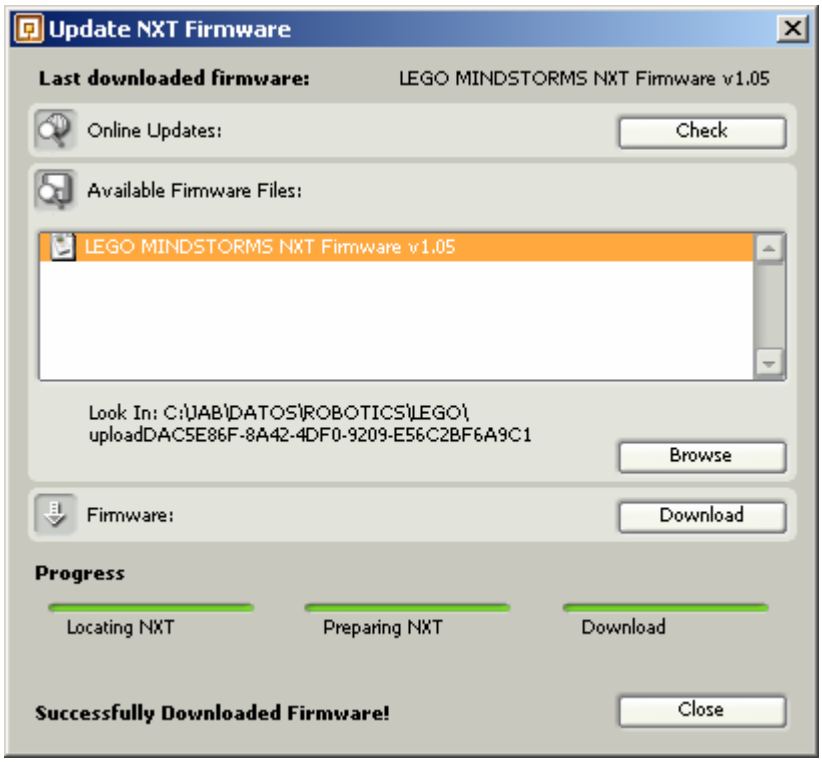
Execute the software and search the option Update NXT Firmware in Tools tab.



When you click in that option then you will see an assistant to download the firmware. Select the firmware that you downloaded and click in download button:



When the process finish then you will see all steps with green color and your NXT brick will have Lego Firmware.



4.2.- Using Tortoise SVN to collaborate in leJOS project

4.2.1.- Introduction

4.2.1.1.- LeJOS Community

LeJOS is a sourceforge project created and maintained to develop a Java Virtual Machine in the Lego Mindstorm NXT Brick. Everybody can participate in any development process with Code and Ideas.

The goals of this document:

1. Learn how LeJOS 's code is organized in Sourceforge servers
2. Learn how to use a Subversion client
3. Learn how to collaborate in LeJOS project with new code

4.2.1.2.- Subversion system

Version control is the art of managing changes to information. It has long been a critical tool for programmers, who typically spend their time making small changes to software and then undoing or checking some of those changes the next day. Imagine a team of such developers working concurrently on the very same files and you can see why a good system is needed to manage the potential chaos.

In LeJOS project we use a subversion system to control the code.

You can see the code's repository here:
http://leJOS .svn.sourceforge.net/viewvc/leJOS /trunk/

4.2.1.3.- Tortoise SVN

TortoiseSVN is a free open-source client for the *Subversion* version control system. That is, TortoiseSVN manages files and directories over time. Files are stored in a central *repository*. The repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to recover older versions of your files and examine the history of how and when your data changed, and who changed it. This is why many people think of Subversion and version control systems in general as a sort of "time machine".

The document explains how to use Tortoise SVN for Windows's operating system. Exist in the market others alternatives:

- Subversive
- Polaris

Further information:

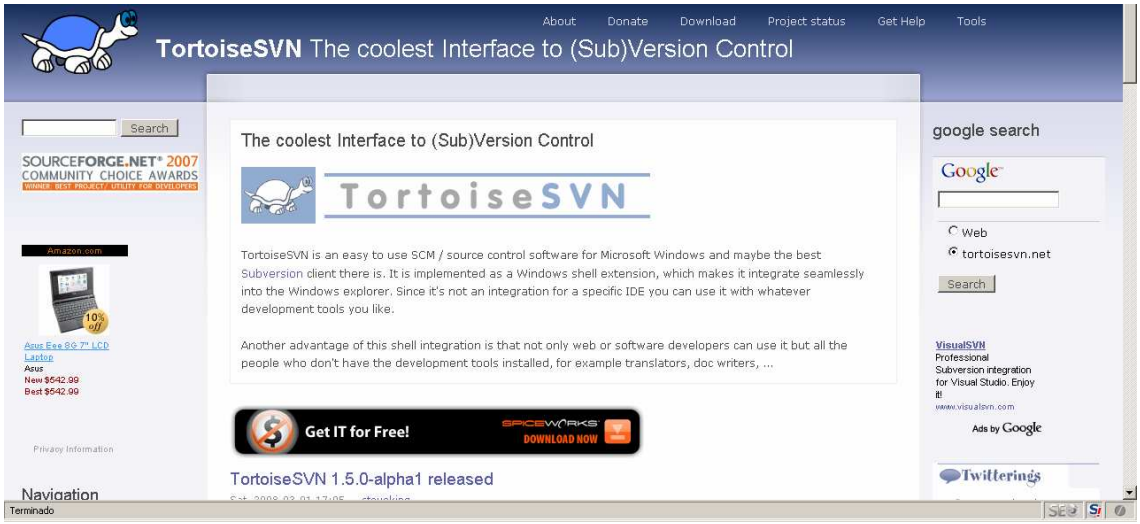
http://subclipse.tigris.org/
http://www.polarion.org/index.php?page=overview&project=subversive

4.2.2.- Installing Tortoise SVN

4.2.2.1.- Download latest version

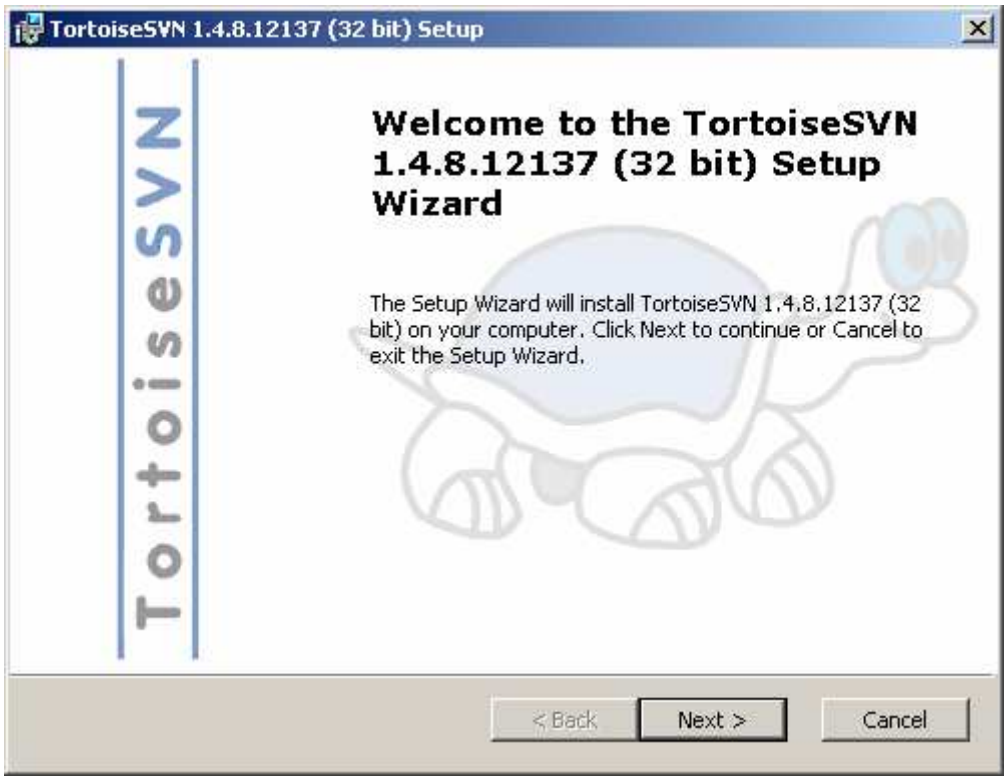
It is necessary to install a Subversion Client in your computer to manage any Subversion system. Download latest version of Tortoise here:

http://www.tortoisesvn.net/

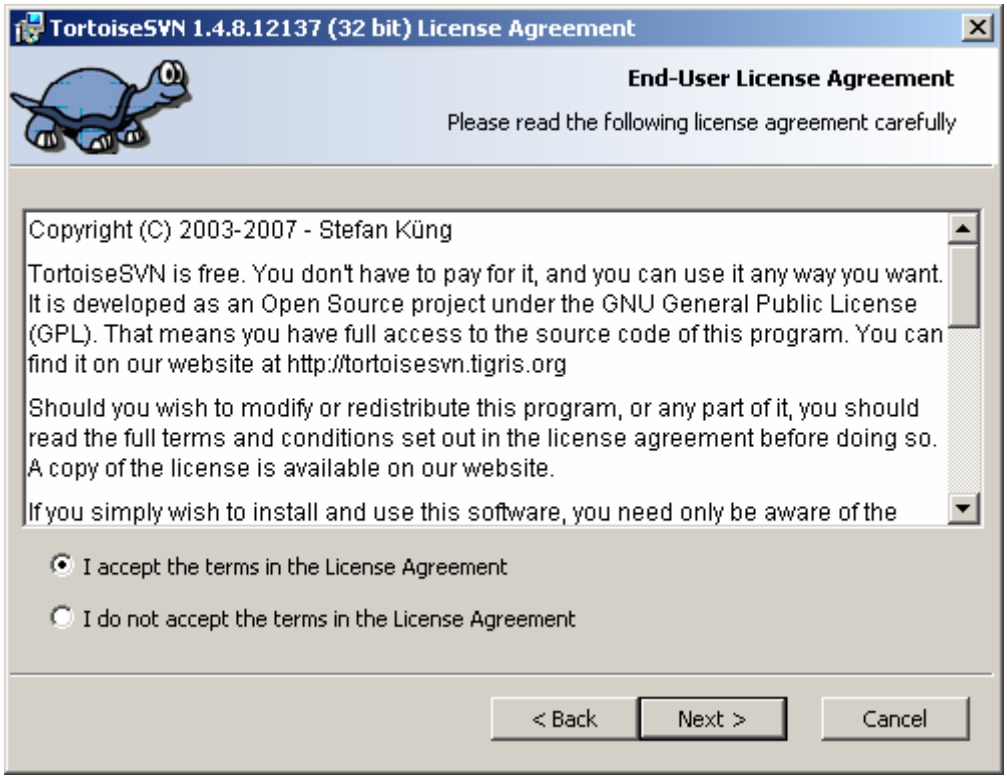


4.2.2.2.- Install Tortoise

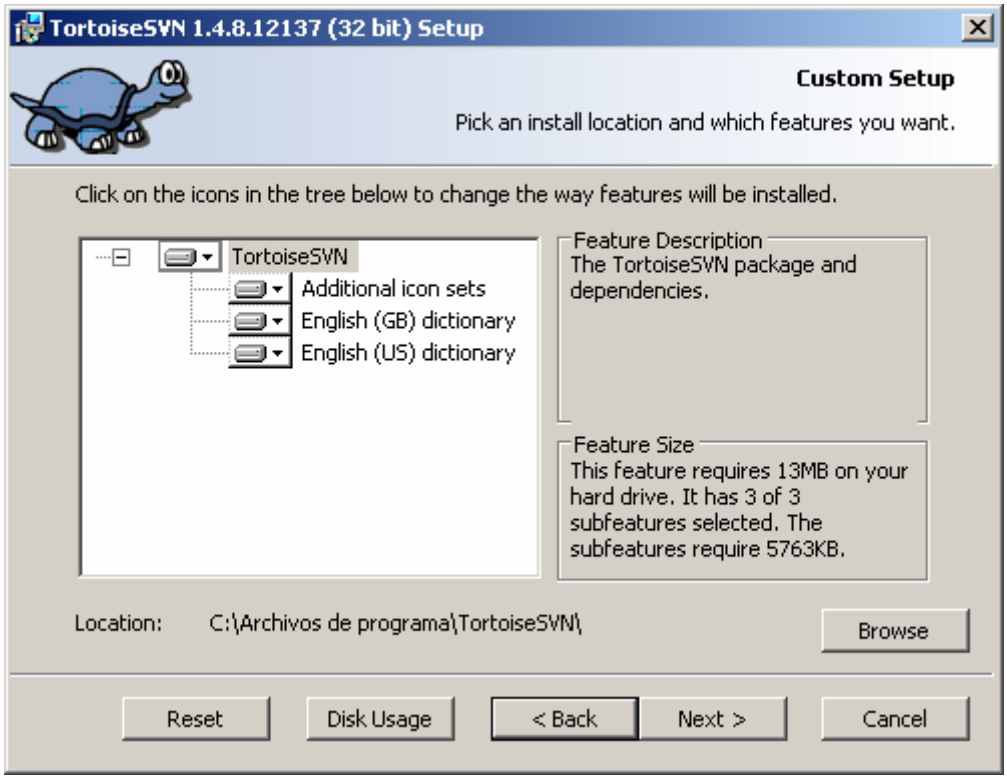
When you have downloaded the client, execute it and follow all steps in the installation.



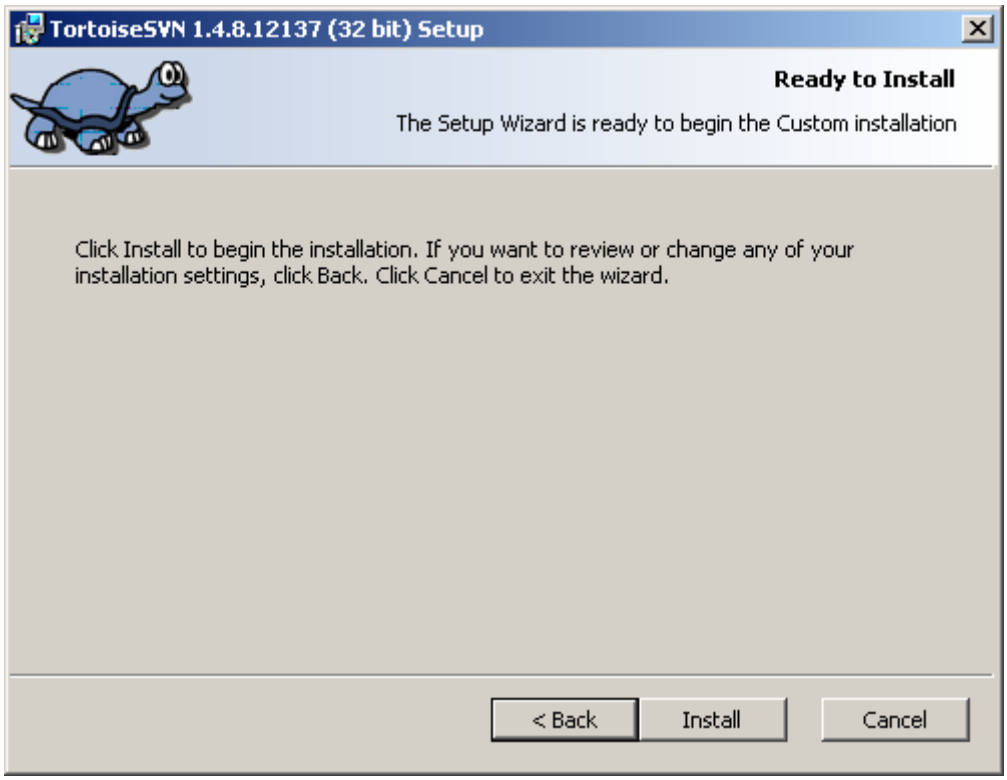
Accept the terms:



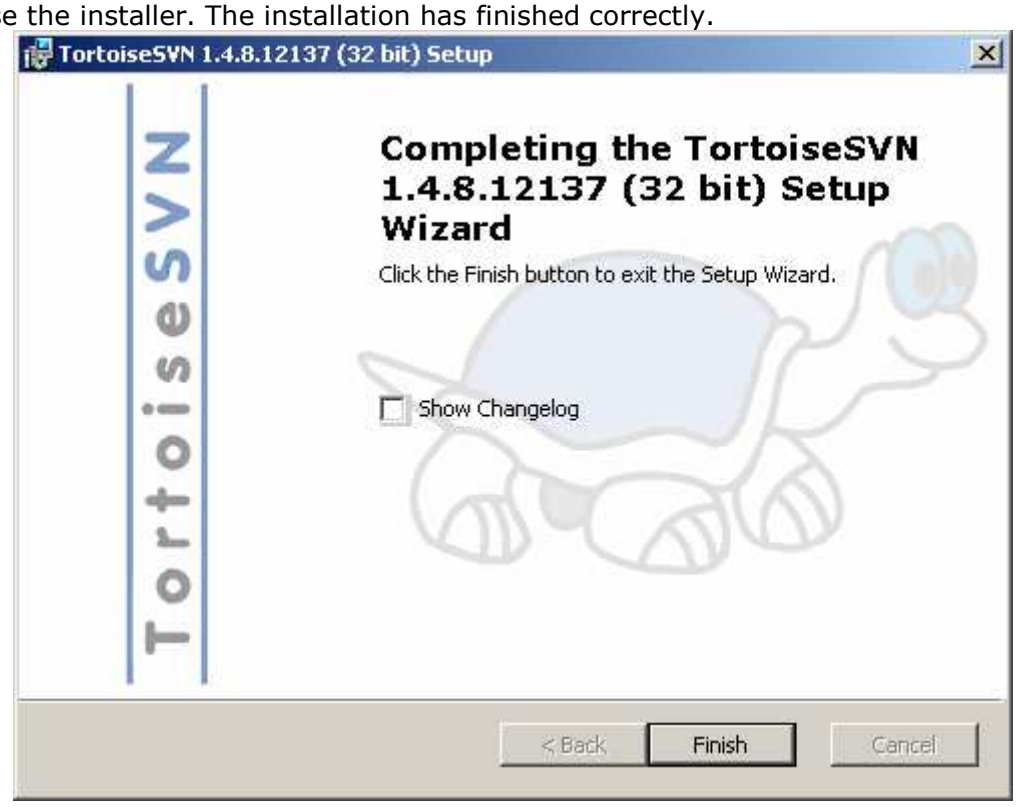
Select all components:



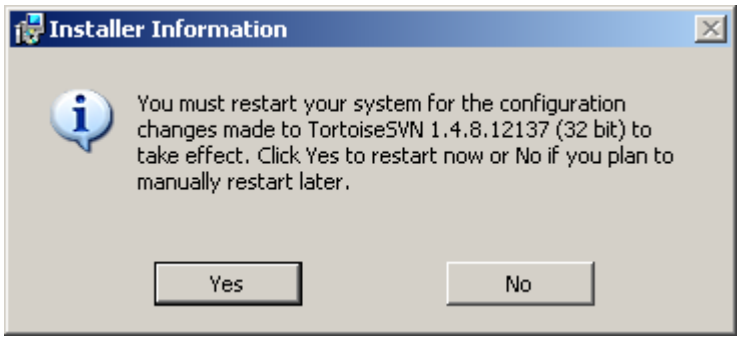
Install the components:



Close the installer. The installation has finished correctly.



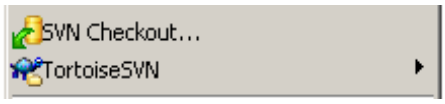
Tortoise needs to reboot your computer to run correctly. Reboot your computer and continue with the following section of this eBook.



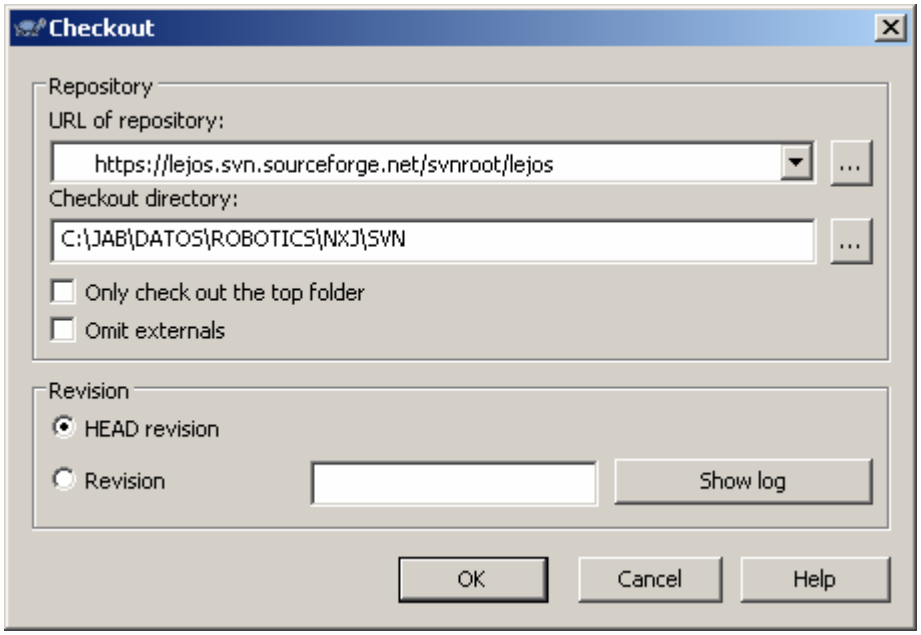
4.2.3.- Downloading LeJOS Repository

4.2.3.1.- First steps with Tortoise SVN

When you have rebooted your machine, if you use contextual menu in windows you can see a new icons in your Windows Explorer:

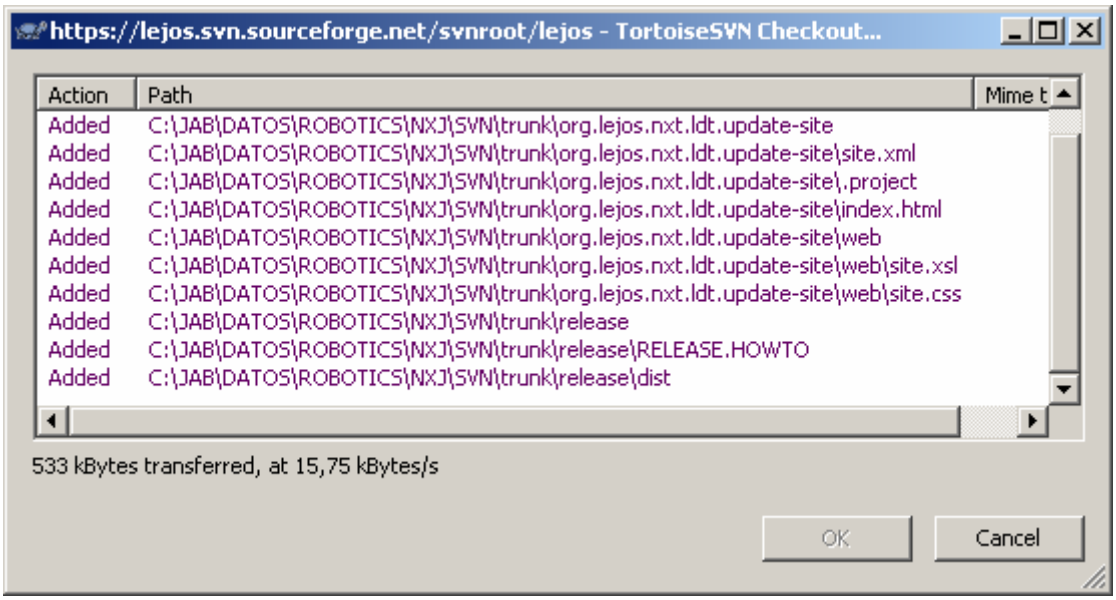


If you want to download all source code, click in the option SVN Checkout:

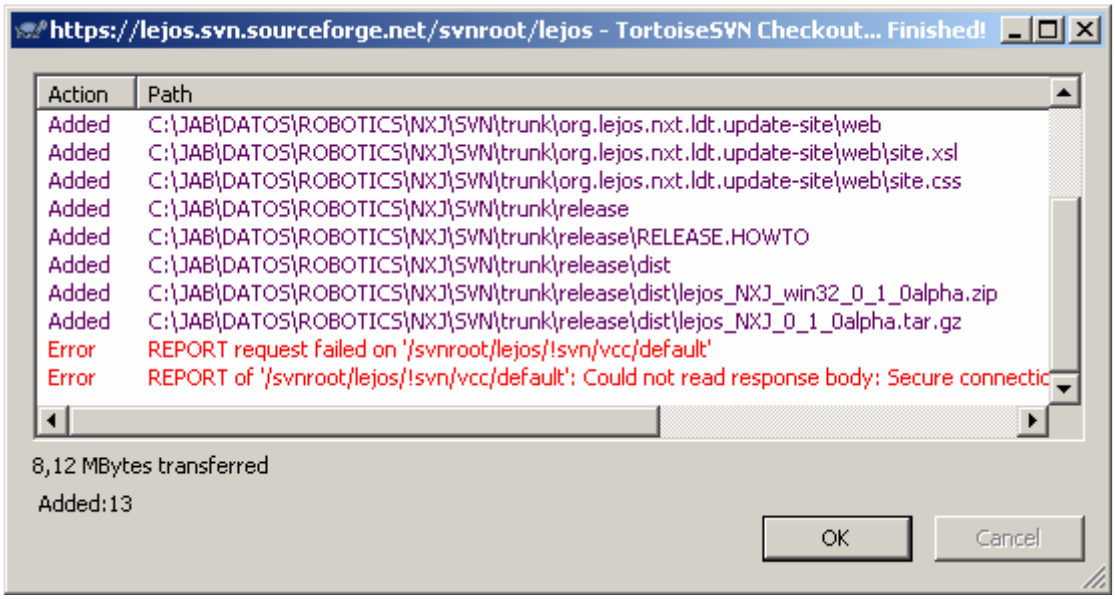


The repository URL is:
https://leJOS .svn.sourceforge.net/svnroot/leJOS

Select the folder where you want to download LeJOS project. Click in Ok Button to start to download then you will see a new window where you can see the files that you are downloading in your computer.

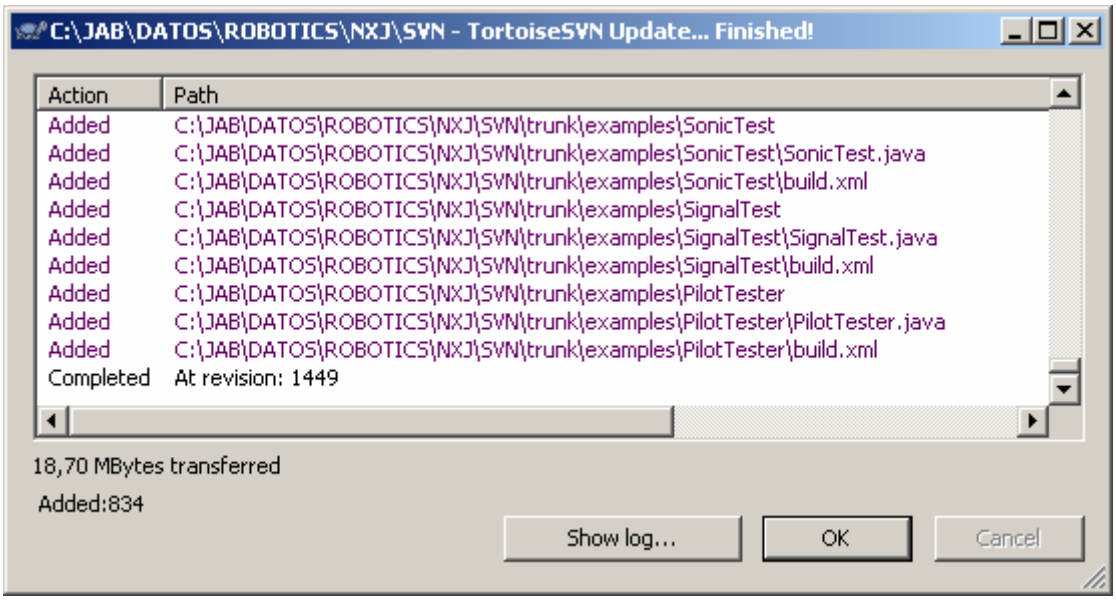


If you lost your internet connection, Tortoise has an option to continue with your download, SVN Update.



Click in SVN Update:

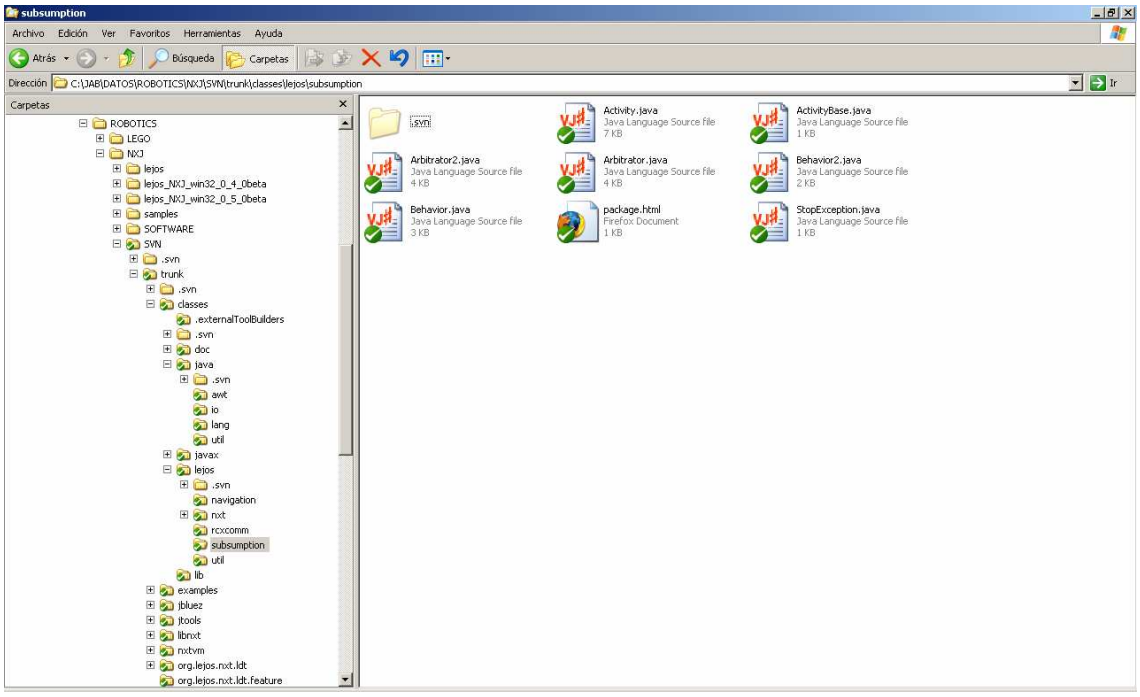
When you finish with your download, you will see this one:



4.2.4.- Using Tortoise in LeJOS

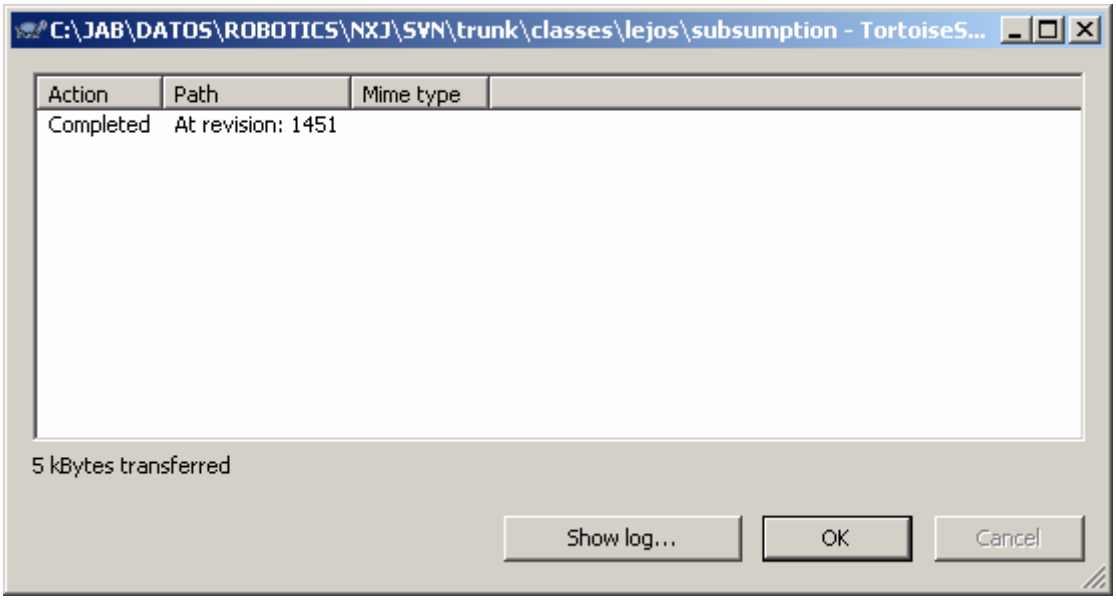
4.2.4.1.- Browsing the repository

When you have downloaded LeJOS project computer then you will see your Windows Explorer with a new icons:

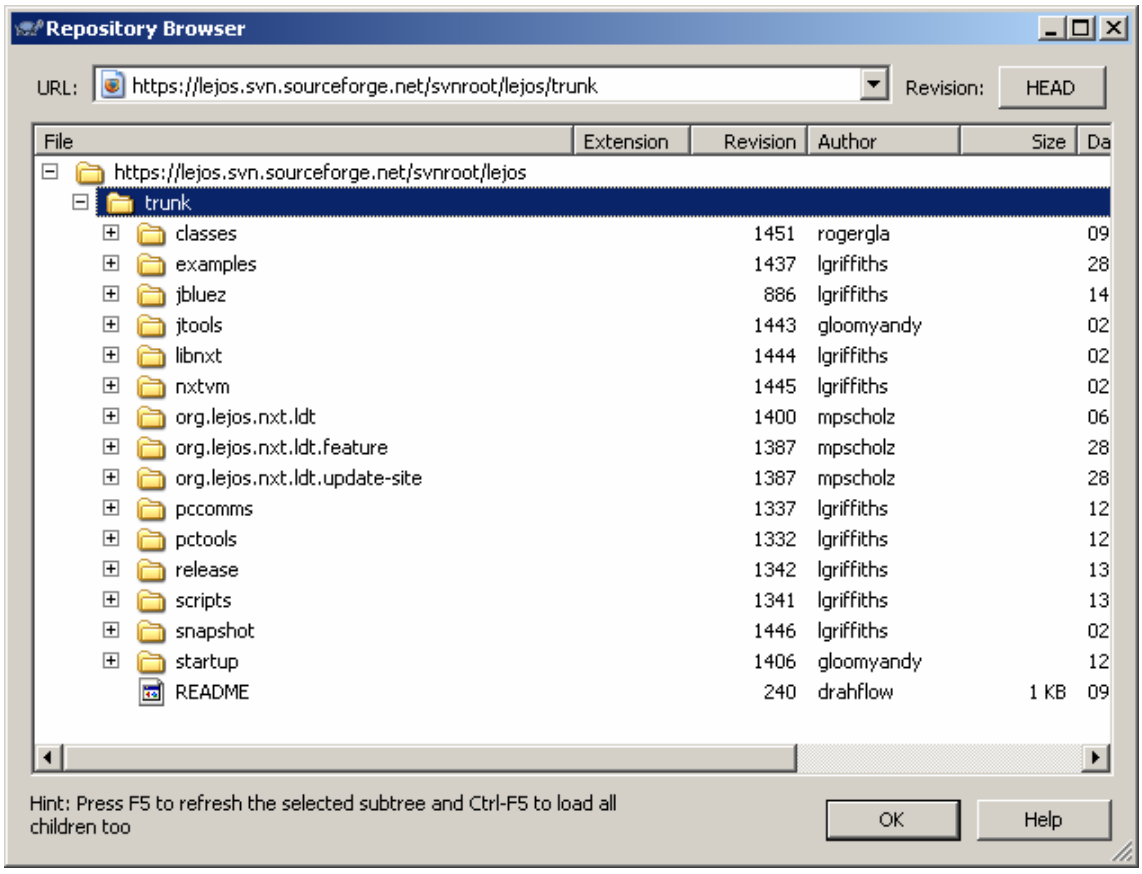


If in any moment you need to update a part of you project, simply select the folder, in this case the folder named "Subsumption" and click in the option "SVN Update"

Then you will update the folder that you have chosen.

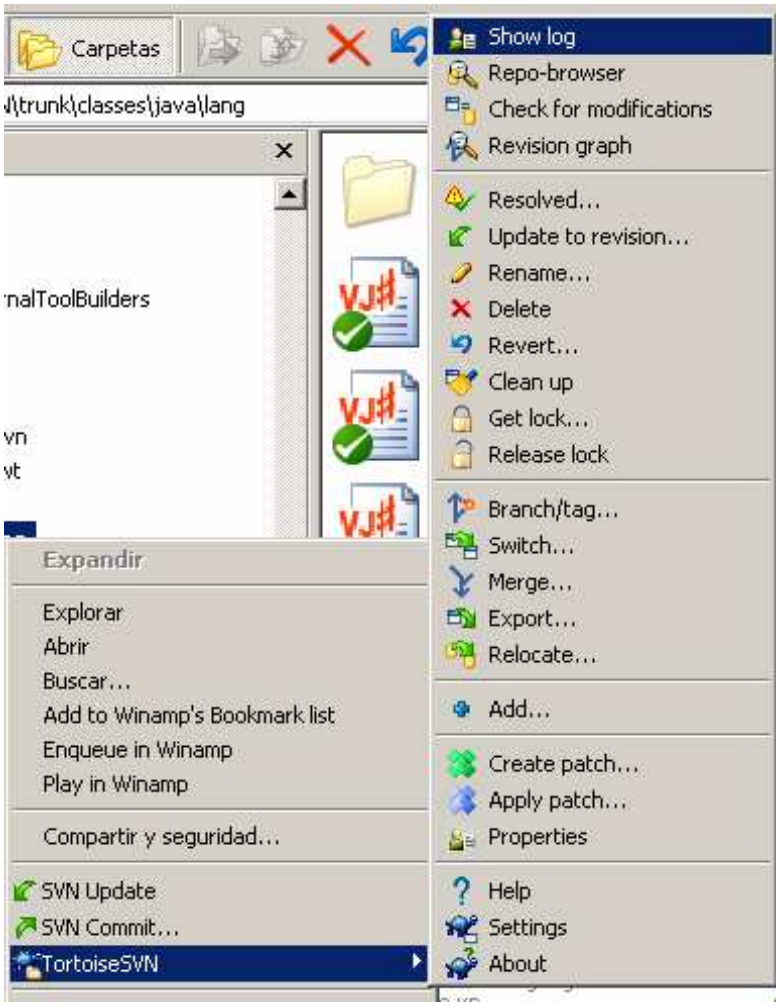


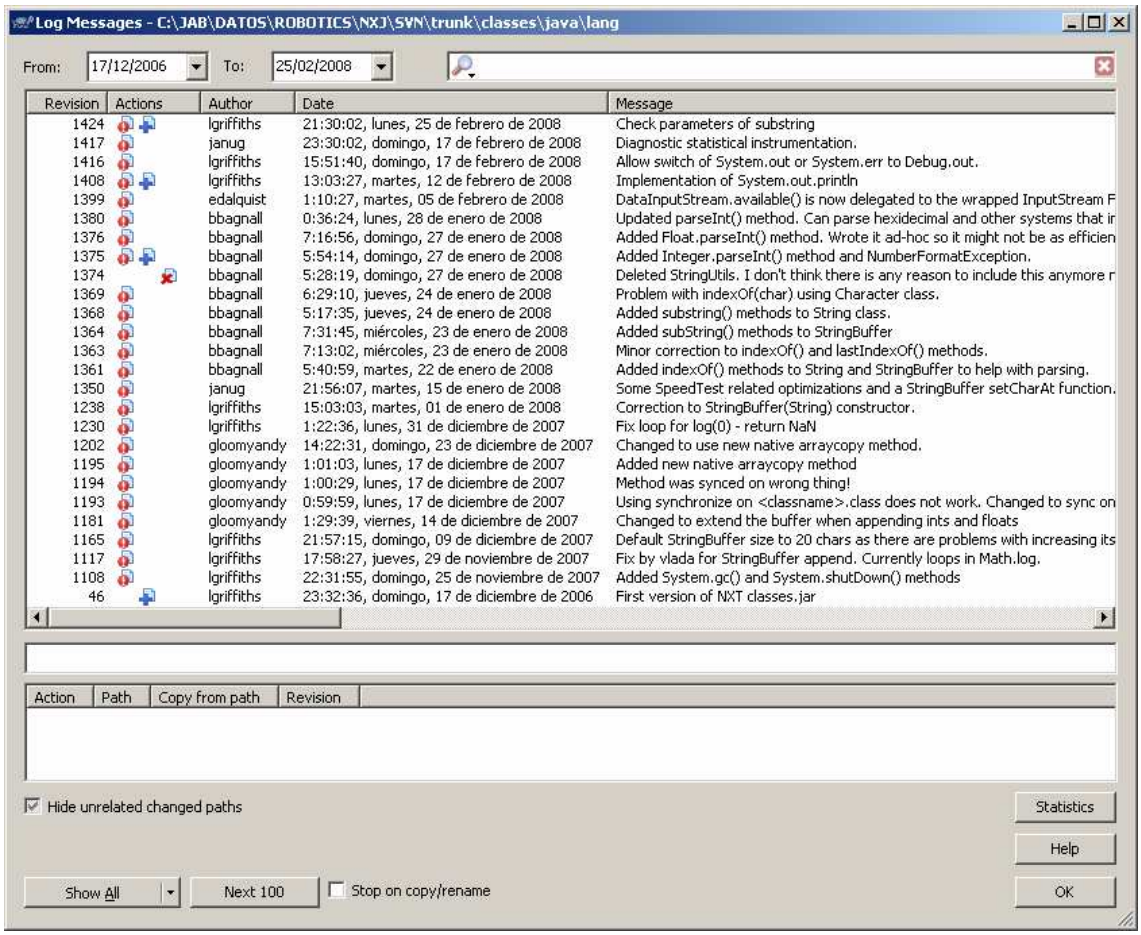
If you need to see the repository in the server, use the option "Repo-browser"



4.2.4.2.- Track the log

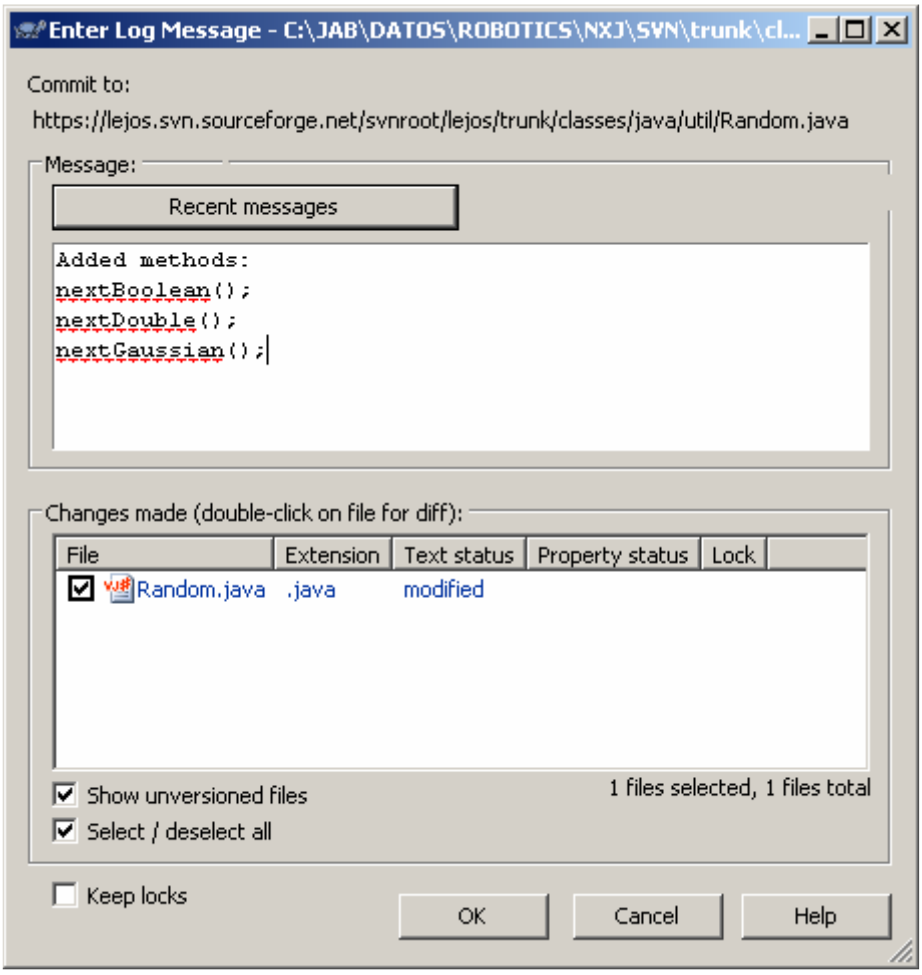
Using subversion technology, it is possible to see the Log. To learn the concept, use a classic package in LeJOS , java.lang. Select the folder lang and click in the option "See log"


Carpetas
\trunk\classes\java\lang
nalToolBuilders
vn
vt
Expandir
Explorar
Abrir
Buscar...
Add to Winamp's Bookmark list
Enqueue in Winamp
Play in Winamp
Compartir y seguridad...
SVN Update
SVN Commit...
TortoiseSVN
Show log
Repo-browser
Check for modifications
Revision graph
Resolved...
Update to revision...
Rename...
Delete
Revert...
Clean up
Get lock...
Release lock
Branch/tag...
Switch...
Merge...
Export...
Relocate...
Add...
Create patch...
Apply patch...
Properties
Help
Settings
About

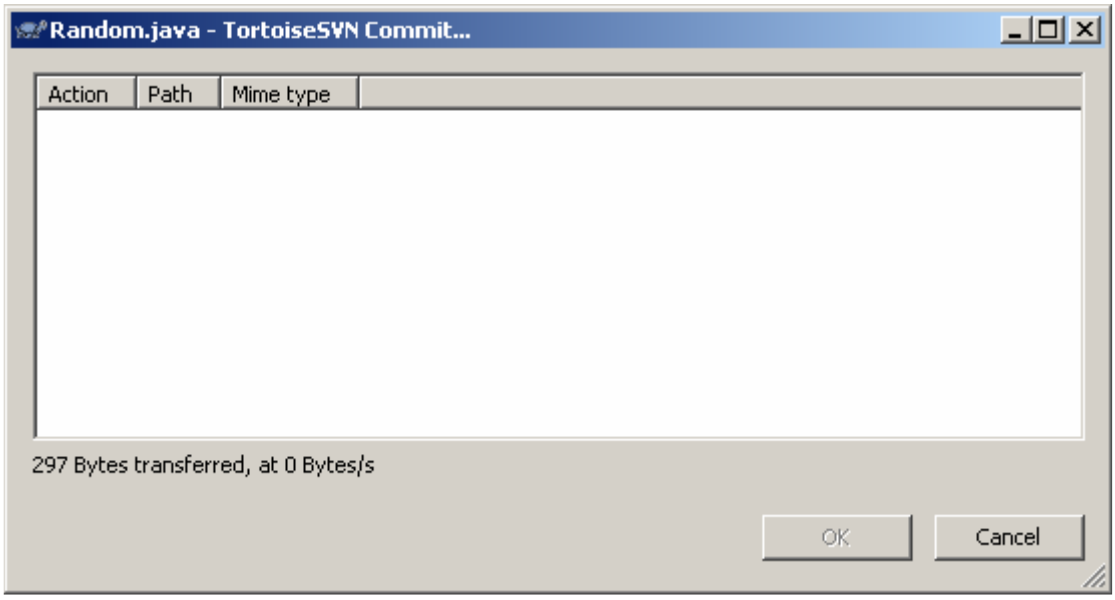


4.2.4.3.- Collaborate with new code

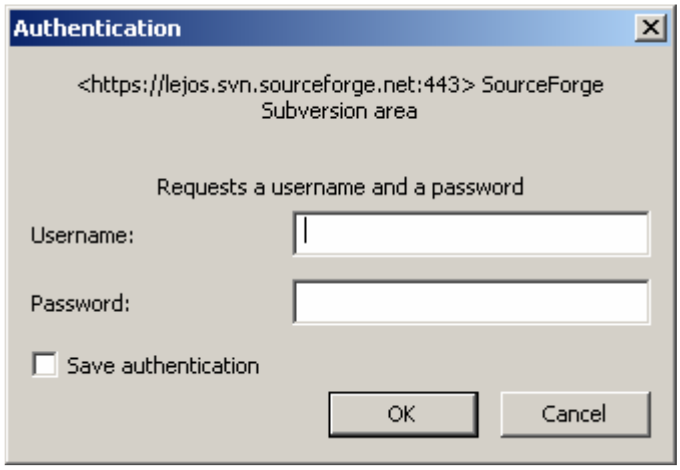
If you want to collaborate with the project then you can update your files in your local computer and click in the option "SVN Commit" then you will see the following window:



In this window, you must to write a text to explain the reasons why you update the code. Click in the Ok button to submit the code and explanation to commit the code.



In this moment, it is necessary to login to finish the process



4.2.5.- How to be a new LeJOS Developer

4.2.5.1.- Request a user

In previous section, we see that if you create new code, it is necessary to authenticate then you need a LeJOS user. Write us to get your LeJOS user to collaborate with us.

4.3.- How to package NXJ programs with Jar

The JavaTM Archive (JAR) file format enables you to bundle multiple files into a single archive file. Typically a JAR file contains the class files and auxiliary resources associated with applets and applications.

The JAR file format provides many benefits:

- **Security:** You can digitally sign the contents of a JAR file. Users who recognize your signature can then optionally grant your software security privileges it wouldn't otherwise have.
- **Compression:** The JAR format allows you to compress your files for efficient storage.
- **Packaging for extensions:** The extensions framework provides a means by which you can add functionality to the Java core platform, and the JAR file format defines the packaging for extensions.
- **Package Sealing:** Packages stored in JAR files can be optionally sealed so that the package can enforce version consistency. Sealing a package within a JAR file means that all classes defined in that package must be found in the same JAR file.
- **Package Versioning:** A JAR file can hold data about the files it contains, such as vendor and version information.
- **Portability:** The mechanism for handling JAR files is a standard part of the Java platform's core API.

JAR files are packaged with the ZIP file format, so you can use them for tasks such as lossless data compression, archiving, decompression, and archive unpacking. These tasks are among the most common uses of JAR files, and you can realize many JAR file benefits using only these basic features.

Even if you want to take advantage of advanced functionality provided by the JAR file format such as electronic signing, you'll first need to become familiar with the fundamental operations.

To perform basic tasks with JAR files, you use the Java™ Archive Tool provided as part of the Java Development Kit. Because the Java Archive tool is invoked by using the `jar` command, this tutorial refers to it as 'the Jar tool'.

As a synopsis and preview of some of the topics to be covered in this section, the following table summarizes common JAR file operations:

Operation	Command
To create a JAR file	`jar cf` *`jar-file input-file(s)`*
To view the contents of a JAR file	`jar tf` *`jar-file`*
To extract the contents of a JAR file	`jar xf` *`jar-file`*
To extract specific files from a JAR file	`jar xf` *`jar-file archived-file(s)`*
To run an application packaged as a JAR file (requires the `Main-class` manifest header)	`java -jar` *`app.jar`*

5.- Links

5.1.- LeJOS links

Official LeJOS Web site
http://www.leJOS .org/
http://leJOS .sourceforge.net/p_technologies/nxt/nxj/api/index.html
http://leJOS .sourceforge.net/forum/
http://sourceforge.net/project/showfiles.php?group_id=9339&package_id=217619
http://sourceforge.net/project/showfiles.php?group_id=178176

Windows LibUSB Project
http://libusb-win32.sourceforge.net/

LibUSB downloads
http://sourceforge.net/project/showfiles.php?group_id=78138&package_id=79216

Java Developer Kit
http://java.sun.com/
http://java.sun.com/javase/downloads/index.jsp

NXJ Plug-in for Eclipse
http://mynxt.matthiaspaulscholz.eu/leJOS /eclipse/

Brian Bagnall's NXT Book
http://www.variantpress.com/books/maximum-lego-nxt

NXTCam Software
http://nxtcamview.sourceforge.net/
http://sourceforge.net/project/showfiles.php?group_id=203058

LeJOS statemachine developer toolkit
http://fermat.nordakademie.de/update/Beschreibung.pdf

5.2.- Java links

Jar tool
http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/jar.html
http://java.sun.com/docs/books/tutorial/deployment/jar/

Java™ Robotics Tutorials
http://www.ridgesoft.com/tutorials.htm
http://www.ridgesoft.com/articles/education/ExploringRoboticsRevision2.pdf

JDK
http://openjdk.java.net/
https://openjdk.dev.java.net/source/browse/openjdk/jdk/trunk/jdk/src/share/classes/java/

5.3.- Lego links

TinyVM Project
http://tinyvm.sourceforge.net/

Lego Mindstorm in Wikipedia:
http://en.wikipedia.org/wiki/Lego_Mindstorms

Lego Mindstorm Google Custom Search Engine
http://www.google.com/coop/cse?cx=009272880476059840496%3Abhbv0xwyrlk

Official Lego Mindstorm Website
http://mindstorms.lego.com/
http://mindstorms.lego.com/support/updates/

NXT Sensor providers
http://www.mindsensors.com
http://www.hitechnic.com/
http://www.codatex.com/

NXT Motor comparative
http://www.philohome.com/motors/motorcomp.htm
http://www.philohome.com/nxtmotor/nxtmotor.htm

5.4.- Robotics links

Rodney Brooks
http://people.csail.mit.edu/brooks/
http://people.csail.mit.edu/brooks/papers/AIM-864.pdf
http://www.cs.brown.edu/people/tld/courses/cs148/02/architectures.html
http://dprg.org/articles/2007-03a/

Subsumption architecture
http://en.wikipedia.org/wiki/Subsumption_architecture
http://people.csail.mit.edu/brooks/papers/representation.pdf
http://geology.heroy.smu.edu/~dpa-www/robo/subsumption/subsumption.pdf
http://geology.heroy.smu.edu/~dpa-www/robo/subsumption/

Steering Behaviors for Autonomous Characters
http://www.red3d.com/cwr/steer/
http://www.steeringbehaviors.de/
http://www.galaxygoo.org/blogs/2006/07/steering_behavior_hunting_and_1.html
http://www.quasimondo.com/archives/000027.php

Collision detection
http://www.cs.unc.edu/~geom/collide/index.shtml

Maja J. Mataric
http://www-robotics.usc.edu/~maja/publications/ieeetra92.pdf

RC Servos and I2C
http://www.min.at/prinz/oe1rib/I2C-Servo/

Bioloid
http://www.bioloid.info/
http://www.robotservicesgroup.com/DetailsPage2.html

RC Servos
http://en.wikipedia.org/wiki/Pulse-width_modulation

5.5.- Technology links

GPS
http://www.informatik.uni-ulm.de/rs/mitarbeiter/hmp/gps/gps.pdf
http://automation.tkk.fi/attach/AS-84-3145/robotpositioning.pdf

Odometry
http://www-personal.umich.edu/~johannb/umbmark.htm
http://geology.heroy.smu.edu/~dpa-www/robo/Encoder/imu_odo/

5.6.- Software links

Eclipse IDE
http://www.eclipse.org/downloads/

Notepad++
http://notepad-plus.sourceforge.net/uk/site.htm

6.- LeJOS Books

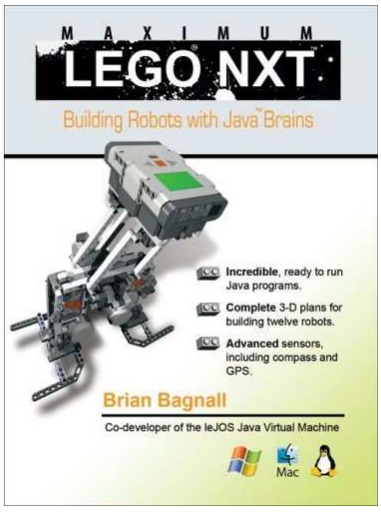
6.1.- Maximum Lego NXT: Building Robots with Java Brains

Maximum Lego NXT introduces a diverse set of projects, building tips, programming code, complete 3D rendered building instructions and hundreds of illustrations to help you realize your robotic dreams.

Using Java™, the most popular and easy to use programming language available, this book will give you endless entertainment and exploration. It introduces the new LEGO® NXT kit, including the NXT intelligent brick and Bluetooth™.

Maximum NXT includes:

Easy to follow instructions by the author of Core LEGO® MINDSTORMS™ Programming
Explanations for all available sensors and expansion products available for the NXT kit, including unique projects interfacing a video camera, cell phone, GPS, data gloves, and many more
An exciting collection of 14 robots, including a chess playing robot, an exoskeleton for your hand, a Mars Rover, a robotic arm you can control through the Internet, a 3D object scanner, soccer robots, and many more
Introduces over two dozen in-depth programming projects including navigation, mapping, precise robotic arm control, voice control and global localization
Artificial Intelligence concepts including Vision analysis, Rodney Brooks' Subsumption Architecture, and Reinforcement Learning
Exciting projects that use third-party sensors like compass tilt sensor, and port expanders
A full chapter on building with the new LEGO stud-less brick paradigm
A complete tutorial on programming Java™
How to install a free development environment for leJOS NXJ, the Java™ Virtual Machine for the NXT



6.2.- Core LEGO MINDSTORMS Programming

From the Back Cover

- Complete 3-D plans for building five unique robots
- Advanced control techniques—including distance and compass sensors
- Behavior control programming, the breakthrough methodology invented at MIT
- For LEGO Mindstorms Robotics Invention System, versions 1.0, 1.5, and 2.0

Your LEGO Mindstorms robots can do more than you ever imagined. The secret: go beyond the built-in tools, and leverage the power of the Java platform—the world's hottest programming technology.

Core LEGO Mindstorms shows you how, step by step. Working from beautifully rendered 3-D plans, you'll construct five unique robots-each capable of increasingly powerful navigation. You'll build and program two powerful custom sensors-an accurate distance sensor and the "Holy Grail" of navigation sensors: the compass sensor.

Brian Bagnall, co-creator of the leJOS Java Virtual Machine for LEGO Mindstorms, starts with the absolute basics then teaches you sophisticated, never-before-published techniques for controlling LEGO Mindstorms robots. No matter what version of LEGO Mindstorms you own, this book will teach you how to build robots with remarkable intelligence and amazing power.

- Installing leJOS , the Java Virtual Machine designed for the LEGO Mindstorms RCX programmable controller
- Setting up your Java platform development environment for LEGO Mindstorms
- Programming rotation and custom sensors
- Using behavior control programming, the biologically inspired MIT breakthrough in robot control
- New techniques for improving navigation accuracy
- Gathering map data and transmitting it back to your computer
- For LEGO Mindstorms RIS versions 1.0, 1.5, and 2.0—including USB versions