

Develop leJOS Programs Step by Step

Version 0.6

Juan Antonio Breña Moral

12-Apr-09

Index

I.- Preface.....	9
I.1.- Introduction.....	9
I.2.- Audience.....	9
I.3.- Organization	9
I.4.- Comments & Questions	10
I.5.- Acknowledgments	10
I.6.- Ebook requirements.....	11
I.7.- About the author	11
I.8.- About the collaborators	11
1.- Introduction	13
1.1.- Lego Mindstorms	13
1.1.1.- History	13
1.1.2.- NXT Brick	15
1.1.3.- Alternatives for Lego Mindstorms NXT	16
1.2.- LeJOS Project	17
1.2.1.- LeJOS solutions.....	18
1.2.2.- The project in numbers	19
1.2.3.- LeJOS Packages	19
1.2.4.- LeJOS documentation.....	21
1.2.5.- Alternatives for leJOS project	21
1.2.6.- To do list for leJOS project.....	21
1.3.- Summary.....	22
2.- Getting started with leJOS.....	23
2.1.- Introduction.....	23
2.2.- LeJOS Installation with leJOS Installer	23
2.2.1.- Introduction	23
2.2.2.- Java JDK Installation.....	23
2.2.3.- Checking your J2SE Installation	26
2.2.4.- Installing leJOS project with leJOS installer	30
2.3.- Manual Installation.....	34
2.3.1.- Prerequisites	34
2.3.2.- Lego Mindstorm NXT Software.....	34
2.3.3.- Java Developer Kit	38
2.3.4.- LibUSB Filter Driver for Microsoft Windows	42
2.3.5.- LeJOS NXJ.....	46
2.4.- Install leJOS firmware into your NXT brick	49
2.4.1.- Introduction	49
2.4.2.- Install leJOS firmware using a GUI.....	49
2.4.3.- Install leJOS firmware using a shell console	52
2.5.- Eclipse IDE and Eclipse plugin for leJOS.....	53
2.5.1.- Introduction	53
2.5.2.- Installing Eclipse	54
2.5.3.- Installing Eclipse plugin for leJOS	54
2.5.4.- Configuring Eclipse plugin for leJOS.....	60
2.5.5.- Creating a new project in Eclipse IDE.....	61

2.6.-	Developing your first program with NXJ	68
2.7.-	Summary.....	69
3.-	<i>Basic concepts about Java.....</i>	71
3.1.-	Introduction.....	71
3.2.-	Learning the example HelloWorld.java	71
3.3.-	Discovering the sections in any Java class	71
3.3.1.-	The Import Area:	71
3.3.2.-	Class encapsulation:	71
3.3.3.-	The main method:.....	72
3.4.-	Summary.....	72
4.-	<i>Sensors</i>	73
4.1.-	Introduction.....	73
4.2.-	Ultrasonic Sensor	73
4.3.-	Compass sensor	74
4.4.-	GPS.....	75
4.4.1.-	Using the package javax.microedition.location	76
4.4.2.-	Using the package lejos.gps.....	76
4.5.-	NXTCam.....	86
4.5.1.-	Introduction	86
4.5.2.-	Install NXTCam driver.....	87
4.5.3.-	Install NXTCamView	93
4.5.4.-	Using NXTCamView to Train NXTCam.....	96
4.5.5.-	NXTCam NXJ API.....	101
4.6.-	NXTLine	104
4.7.-	Touch sensor	104
4.8.-	Sound sensor	106
4.9.-	Summary.....	108
5.-	<i>Actuators</i>	109
5.1.-	Introduction.....	109
5.2.-	NXT Motors.....	109
5.3.-	PF Motors	111
5.4.-	RC Servos and DC Motors with NXTe/LSC	113
5.4.1.-	Introduction	113
5.4.2.-	NXTe architecture	113
5.4.3.-	Servos	115
5.4.4.-	LeJOS and NXTe/LSC	116
5.5.-	RC Servos with NXTServo.....	118
5.6.-	RC Servos systems with MRS H01.....	120
5.6.1.-	Using MRS H01 with Bluetooth.....	120
5.6.2.-	Using MRS H01 with I2C	120
5.7.-	Summary.....	121
6.-	<i>Graphical user interfaces with leJOS.....</i>	122
6.1.-	Introduccion	122

6.2.-	LCD	122
6.3.-	System.out.....	122
6.4.-	RConsole	122
6.5.-	Learning javax.microedition.Lcdui	123
6.5.1.-	Using the class Graphics.....	124
6.6.-	Creating Textmenu for your GUI.....	124
6.7.-	Summary.....	125
7.-	<i>Communications</i>	126
7.1.-	Introduction.....	126
7.1.1.-	Communications for System integrations.....	126
7.1.2.-	Communications for NXT integrations	126
7.2.-	Strategies.....	126
7.3.-	Summary.....	126
8.-	<i>Communications I: Bluetooth</i>	127
8.1.-	Introduction.....	127
8.2.-	History.....	127
8.3.-	Bluetooth Architecture	127
8.4.-	Bluetooth Protocols	128
8.5.-	Bluetooth profiles	129
8.6.-	Bluetooth Networks.....	130
8.6.1.-	Piconet	130
8.6.2.-	Scatternet	131
8.7.-	Bluetooth Connections.....	131
8.8.-	How to use Bluetooth with leJOS	132
8.8.1.-	Discovery Bluetooth Device.....	132
8.8.2.-	Connect with a Bluetooth device.....	133
8.8.3.-	Exchange Data between a NXT Brick with another one or a Bluetooth device	134
8.8.4.-	Listen a Bluetooth connection	136
8.9.-	LeJOS examples using Bluetooth	137
8.10.-	Summary.....	137
9.-	<i>Communications II: USB</i>	138
9.1.-	Introduction.....	138
9.2.-	Send data from PC to NXT	138
9.3.-	Receive data from PC	139
9.4.-	Summary.....	140
10.-	<i>Communications III: RS485</i>	141
10.1.-	Introduction.....	141
10.2.-	Send data with RS485	141
10.3.-	Receive data with RS485	142
10.4.-	Summary.....	144
11.-	<i>Communications IV: I2C</i>	145

11.1.-	Introduction.....	145
11.2.-	I2C Bus terminology	146
11.3.-	Terminology for bus transfer.....	146
11.4.-	LeJOS API.....	147
11.5.-	I2C Examples with leJOS.....	148
11.6.-	Migrating code I2C from others platforms.....	149
11.6.1.-	Migrating I2C Code from RobotC to Java leJOS	149
11.6.2.-	Migrating I2C Code from NXJ to Java leJOS	150
11.7.-	Summary.....	153
12.-	<i>Subsumption architecture</i>	<i>154</i>
12.1.-	Introduction.....	154
12.2.-	Subsumption package in NXJ	154
12.3.-	Subsumption example.....	154
13.-	<i>Multithreading with Java leJOS.....</i>	<i>157</i>
13.1.-	Introduction.....	157
13.2.-	The Thread concept	157
13.3.-	The Thread life cycle.....	159
13.4.-	Using the class Thread.....	159
13.4.1.-	Method start.....	160
13.4.2.-	Method isAlive	160
13.4.3.-	Method sleep.....	161
13.5.-	Examples.....	161
13.5.1.-	Example1: LineFollower	161
13.5.2.-	Example2: Stereo US.....	165
14.-	<i>LeJOS and mobile phones</i>	<i>169</i>
14.1.-	Introduction.....	169
14.2.-	Install SDKs.....	169
14.2.1.-	Installing J2SE SDK.....	169
14.2.2.-	Installing Java ME Wireless Toolkit	169
14.3.-	Install IDE.....	173
14.3.1.-	Installing Eclipse IDE.....	173
14.3.2.-	Installing Eclipse ME Plugin	175
14.4.-	Install Optional Software	178
14.4.1.-	Proguard	178
14.4.2.-	Antenna.....	179
14.5.-	Creating your first Java ME project.....	180
14.6.-	Creating a new Java ME Project.....	180
14.7.-	Java ME Settings.....	183
14.8.-	Add a HelloWorld Middlet.....	185
14.9.-	Test your Middlet with a simulator	187
14.10.-	Package your application.....	189
14.11.-	Deliver your Middlet into a Mobile Phone.....	190
14.11.1.-	Deliver Java ME with Nokia PC Suite	190

15.-	<i>LeJOS Tools</i>	193
15.1.-	Introduction	193
15.2.-	NXJFlashg	193
15.3.-	NXJBrowse	194
15.4.-	NXJConsole	196
15.5.-	NXJMonitor	197
15.6.-	LeJOS statemachine developer toolkit	199
15.6.1.-	Introduction	199
15.6.2.-	Installing LeJOS statemachine developer toolkit.....	200
15.6.3.-	Creating the first project with the toolkit.....	209
15.6.4.-	Videos.....	212
15.7.-	Summary	212
16.-	<i>Robotics projects</i>	213
16.1.-	Introduction	213
16.2.-	Steering behaviors	213
16.2.1.-	Introduction	213
16.2.2.-	Behaviors.....	213
16.2.3.-	Obstacle Avoidance	214
16.3.-	Parallel Architectures	218
16.3.1.-	Introduction	218
16.3.2.-	JCSP re	218
16.3.3.-	JCSP re Stack.....	219
16.3.4.-	The Process Orientated Design Pattern	220
16.3.5.-	LeJOS & JCSP re for building robot controllers	220
16.3.6.-	Line follower robot with JCSP re & leJOS.....	221
17.-	<i>FAQ</i>	230
17.1.-	How to reinstall Lego Firmware	230
17.1.1.-	Introduction	230
17.1.2.-	Download latest Lego firmware	230
17.1.3.-	Set your NXT brick in update mode.....	230
17.1.4.-	Reinstall Lego firmware	231
17.2.-	Using Tortoise SVN to collaborate in leJOS project	234
17.2.1.-	Introduction	234
17.2.2.-	Installing Tortoise SVN.....	235
17.2.3.-	Downloading LeJOS Repository	238
17.2.4.-	Using Tortoise in LeJOS	240
17.2.5.-	How to be a new LeJOS Developer.....	245
17.3.-	How to use beta software from leJOS SVN?	245
17.3.1.-	Make a checkout from leJOS SVN.....	245
17.3.2.-	Copy snapshot to your leJOS installation	246
17.4.-	How to package NXJ programs with Jar	247
18.-	<i>Links</i>	248
18.1.-	LeJOS links	248
18.2.-	Java links	248
18.3.-	Lego links	249
18.4.-	Robotics links	249

18.5.- Technology links.....	250
18.6.- Software links	250
18.7.- Multithreading Links.....	250

Revision History

Name	Date	Reason For Changes	Version
Juan Antonio Breña Moral	12/01/2008	Initial release	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
Juan Antonio Breña Moral	20/03/2009	Book reorganization	0.5
Juan Antonio Breña Moral	10/04/2009	Add RS485 Chapter	0.6

I.- Preface

I.1.- Introduction

In next 10 years, Robotics will become in one of the most helpfully technology for the society. Currently, robotics field is not in a mature phase and it needs new ideas to evolve but this goal is not easy because robotics is a complex science and it has several research lineas as as Localization, Computer vision & Neural Networks for example.

In the market exists many products to learn basic concepts and techniques about robotics and Artificial Intelligence but in my personal opinion, Lego Mindstorms NXT is the best platform to be used in robotics courses at secondary school, university bachelors, and postgraduate programs / Phd.

Lego Midstorms NXT has many ways to develop software for robots but this ebook only offer support for leJOS project which offers the possibility to develop with Java.

This ebook is a project to spread the knowledge about leJOS project and Java techniques to develop software for robots. This ebook is live and every 3-6 months, I will try to update with new ideas and techniques from the projects and the readers.

Enjoy, Learn, **Contact with me** to improve the eBook and share ideas.

Juan Antonio Breña Moral.

www.juanantonio.info

I.2.- Audience

The ebook has been written to be read by the following kind of users:

- Lego Mindstorms users
- LeJOS Developers
- Java Developers
- Teachers who teach robotics courses
- Students in Secondary School
- Students in University
- Students in Postgraduate programs / Phd
- Scientifics
- Engineers

I.3.- Organization

The ebook has been organized in the following chapters:

Chapter 1: Introduction

This chapter explains what Lego Mindstors NXT is and the context in the market. The chapter explains the origins, history and milestones with the product Lego Mindstorms NXT.

Chapter 2: LeJOS project

This chapter explains the LeJOS Project, API, Tools, Project structure, etc.

Chapter 3: Getting started with leJOS project

This chapter explains how to install LeJOS Project to execute the , API, Tools, Project structure, etc.

Chapter 4: Basic concepts about Java

This chapter explains basic concepts about Java.

Chapter 5: Sensors

This chapter explains how to use sensors from NXT Kit or sensors from NXT providers as Mindsensors, Hitechnic, CANCAN and others.

Chapter 6: Actuators

This chapter explains how to use actuators. This chapter includes NXT Motors, PF Motors, Servos, DC Motors and RCX Legacy Motors.

Chapter 6: GUI

This chapter explains how to use LCD in NXT brick

Chapter 7-11: Communications

These sets of chapters explain how to use Bluetooth, USB, RS485 & I2C Protocols.

Chapter 12: Sumsubption architecture

This chapter explains how to use sumsubption architecture

Chapter 13: Multithreading

This chapter explains how to manage a java feature which allow your robot manage in parallel multiple tasks.

Chapter 14: LeJOS and mobile phones

This chapter explains how to use some leJOS with mobile phones.

Chapter 15: LeJOS Tools

This chapter explains how to use some tools which are included in every leJOS release and others from leJOS community.

I.4.- Comments & Questions

Please, I would like to receive your feedback about the book to improve it.

I.5.- Acknowledgments

This Project has been posible with the help of my family and friends as Juan Diego Avendaño, Antonio Tejero, Bruno Piñeiro, Isaac Olmos & Marina Perez. I have to congratulate to Brian Bagnall because in the past, he gave the opportunity to join to the leJOS developer team. Besides I give my sincere thanks to my colleagues in the project as Lawrie Griffiths, Andy Shaw, Roger Glassey & Matthias Paul Scholz,

they have nice ideas about the future of the project and they have strong experience with Java. Everyday, I learn new things with them.

Finally I have to give many thanks for my readers in special, Yu Yang, Deepak Patik, Dhinakar Radhakrishnan, Takashi Chikamasa, Koldo, Craig Reynolds, Matt Denton and Jose Maria Plaza.

Sorry If I forgot some name.

I.6.- Ebook requirements

This ebook needs the following requirements to use correctly

1. Lego Mindstorms NXT Kit
2. Computer with your favorite OS (Windows, Linux or Mac OS)

Note:

For the moment, this ebook only offer support for Windows OS but I hope to expand the support for Linux too in 2009

I.7.- 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. Currently, he teaches NXT courses and study Phd about Robotics and Artificial Intelligence in URJC.

Further information:

www.juanantonio.info

www.lejostraining.com

I.8.- 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.patricklismore.com>

1.- Introduction

1.1.- Lego Mindstorms

Lego Mindstorms is an educational product designed by Lego to build easy robots. The product has evolved with the decades and currently Lego Mindstorms NXT is the third generation of the same product.

1.1.1.- History

The history of the product Lego Mindstorms is the following:

1988:

Collaboration between the LEGO Group and Massachusetts Institute of Technology (MIT) begins on development of an "intelligent brick" that will bring LEGO creations to life via computer programming.



1989:

Dr. Seymour Papert, of Massachusetts Institute of Technology's Development Laboratory of Computer Learning becomes "LEGO Professor of Learning Research."

1998:

LEGO MINDSTORMS and the Robotics Invention System are unveiled to the public at Toy Fairs in Nürnberg, London and New York.

1998:

RoboTour™ '98 launches from the Chicago Museum of Science and Industry, kicking-off a two-month, 30-city odyssey across America in search of learning about and seeing everything robotic.



1999:

The Robotics Discovery Set™, a derivative of the Robotics Invention System allowing users to program right on the smart brick instead of through the computer, and the Droid Developer Kit™, a pre-programmed, remote controlled constructible robot kit, are unveiled at the International Toy Fair in New York.

The Robotics Discovery Set, Ultimate Accessory Set, Droid Developer Kit and the Robotics Invention System 1.5 are released in the United States. The Droid Developer Kit and the Robotics Invention System 1.5 are released in Europe and Asia, and The Robotics Discovery Set and the Robotics Invention System 1.5 are launched in the United Kingdom.

2000:

The Robotics Invention System 2.0, Dark Side Developer Kit™ (a preprogrammed, remote controlled constructible robot), Vision Command System™ (a PC camera expansion kit for the RIS) and Exploration Mars™ (themed robot challenges, building instructions and games for the RIS) expansion set are unveiled at the International Toy Fair in New York.

The Robotics Invention System 2.0, Dark Side Developer Kit and Vision Command System are released in the United States.

2001:

SpyBotics, a spy gaming-oriented series of remote controlled and programmable robots, are unveiled at the International Toy Fair in New York.

2006:

The next generation of LEGO MINDSTORMS robotics is unveiled at the International Consumer Electronics Show.



2009:

LEGO launches LEGO Mindstorms NXT 2.0 with a better commercial kit, but the hardware is the same.

1.1.2.- NXT Brick

The NXT brick is the brain of any LEGO Mindstorm robot. It's an intelligent, computer-controlled LEGO brick that lets a 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

1.1.3.- Alternatives for Lego Mindstorms NXT

In the market exist others alternatives for Lego Mindstorms NXT:

- High-Level Plug'n'Play Architectures
 - Phidgets
 - Teleo by MakingThings
 - ActiveWire
 - NIQ EZIO
- Hybrid MC/PC systems
 - Arduino
 - Sun spot
 - Jstamp
 - BasicStamp
 - AVRmini
 - OOPIC
 - Handyboard
 - BrainStem
 - iCube
- Low-level MC
 - Lego Mindstorm RCX
 - MIT Programmable Bricks / Crickets
- Others:

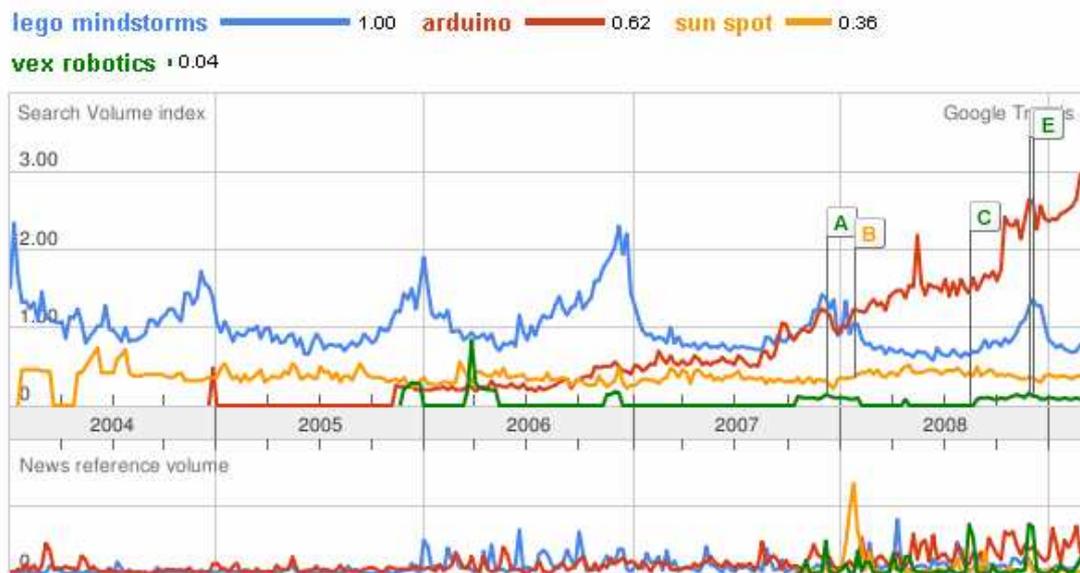
- Intel Motes
- iStuff

But if you study the market in detail, you will notice that the only serious competitor for the product Lego Mindstorms NXT is the project Arduino.

1.1.3.1.- Arduino platform

Arduino is a great project and it has nice features as:

- IDE to develop software for Arduino
- A functional programming language to develop software. Similar to C.
- The possibility to expand the hardware features with homebrew electronics
- A good price
- Nice community
- Multi purpose platform



In my opinion Arduino is good generic platform for guys who like electronics and it can be used in several projects, but Lego Mindstorms NXT is oriented for robotics and software.

1.2.- LeJOS Project

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

LeJOS project offers support for Lego Mindstorms NXT and previous product, Lego Mindstorms RCX.

LeJOS project deliver the following solutions:

- Lego Mindstorms NXT
 - JVM for NXT Brick
 - LeJOS API for NXT brick
 - LeJOS PC Comms
 - LeJOS JavaME Comms
 - LeJOS Tools
- Lego Mindstorms RCX

- JVM for RCX Brick
- LeJOS API for NXT brick
- LeJOS PC Comms
- LeJOS Tools

1.2.1.- LeJOS solutions

With leJOS project, you can develop software for robots with 2 philosophies:

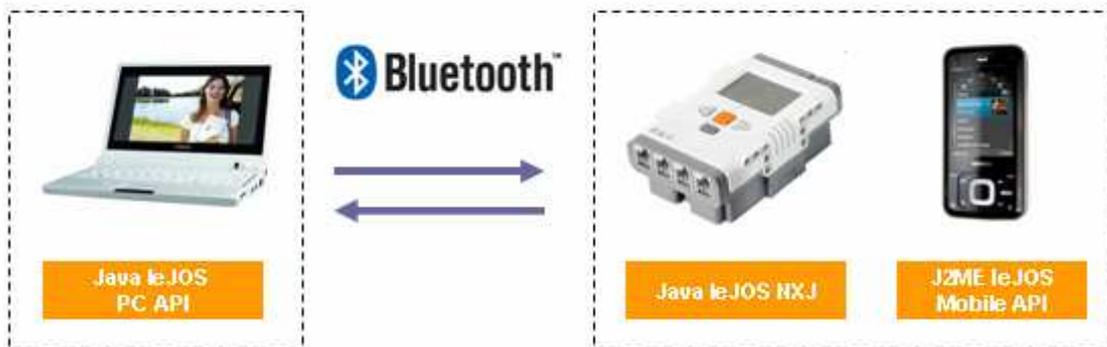
1. Distributed solution
2. Non distributed solution

A distributed solution

A distributed solution is a solution which you add a certain kind of logic in very part of your project. Since 2009, you can distribute your logic in 2 parts:

1. Side A:
 - a. PC/Notebook/Netbook (J2SE)
 - b. Mobile phone (JavaME)
2. Side B:
 - a. NXT brick (Java leJOS, NXJ)

#1. Distributed architecture

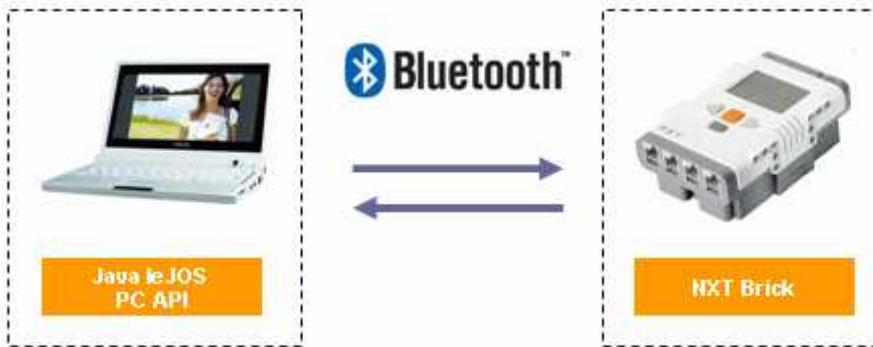


With this kind of solution you can develop your control system in a machine with better resources, for example a PC and send/receive information from your NXT brick but in case of your communication link is broken, the sides, PC and NXT can execute a reconnection routines to stablish the communication again. This idea is really important for complex project in robotics because imagine if you don't add logic in a mars robot for example.

A non distributed solution

The another way to develop your project is adding the logic in your PC only so NXT brick is controlled remotely by the PC.

#2. Non distributed architecture



1.2.2.- The project in numbers

In 2009, leJOS project has website traffic with more than 700 visits everyday from 100 countries.

Panel



Besides el official forum has more than 500 registered users in 2009

1.2.3.- LeJOS Packages

Currently, the official packages which are available to use are the following:

Package	Description
java.awt	Minimal AWT package for Rectangle class
java.io	Input/Output support
java.lang	Core Java classes
java.util	Utilities
javax.bluetooth	Standard Bluetooth classes
javax.microedition.io	J2ME I/O
javax.microedition.lcdui	J2ME LCD User Interface classes.
javax.microedition.location	J2ME Support for location services
lejos.gps	GPS Parsing
lejos.keyboard	Support for SPP keyboards
lejos.localization	Localization support
lejos.math	Extra Math classes for leJOS
lejos.navigation	Navigation classes
lejos.nxt	Access to NXT sensors, motors, etc.
lejos.nxt.addon	Access to third party and legacy RCX sensors, motors and other hardware not included in the Lego NXT kit
lejos.nxt.comm	NXT communication classes
lejos.nxt.debug	Debugging thread classes
lejos.nxt.remote	Remote NXT access over Bluetooth

lejos.nxt.socket	Support for sockets via PC SocketProxy
lejos.rcxcomm	Emulation of RCX communication classes
lejos.subsumption	Support for Subsumption architecture
lejos.util	More utility classes

Core java packages:

These packages are subset of Java core classes, but they are sufficient to develop any java program.

Package	Description
java.awt	Minimal AWT package for Rectangle class
java.lang	Core Java classes
java.util	Utilities

Communication packages:

One the best features in leJOS project is the communication support. LeJOS offers support to establish a communication link using the following protocols:

- Bluetooth
- USB
- RS485
- I2C

Package	Description
java.io	Input/Output support
javax.bluetooth	Standard Bluetooth classes
javax.microedition.io	J2ME I/O
lejos.nxt.comm	NXT communication classes
lejos.nxt.remote	Remote NXT access over Bluetooth
lejos.nxt.socket	Support for sockets via PC SocketProxy
lejos.rcxcomm	Emulation of RCX communication classes

NXT packages:

These packages allow you to manage NXT brick, sensors and actuators.

Package	Description
javax.microedition.lcdui	J2ME LCD User Interface classes.
javax.microedition.location	J2ME Support for location services
lejos.gps	GPS Parsing
lejos.keyboard	Support for SPP keyboards
lejos.nxt	Access to NXT sensors, motors, etc.
lejos.nxt.addon	Access to third party and legacy RCX sensors, motors and other hardware not included in the Lego NXT kit
lejos.nxt.debug	Debugging thread classes
lejos.util	More utility classes

Robotics / AI packages:

These packages offer support for some robotics problems as localizacion and navigation.

Package	Description
lejos.localization	Localization support
lejos.math	Extra Math classes for leJOS
lejos.navigation	Navigation classes
lejos.subsumption	Support for Subsumption architecture

1.2.4.- LeJOS documentation

Every leJOS release has documentation about the packages in Javadoc format. You can take a look about the documentation in the following URL:

<http://leJOS.sourceforge.net/nxt/nxj/api/index.html>

Packages	
java.awt	Minimal AWT package for Rectangle class
java.io	Input/Output support
java.lang	Core Java classes
java.util	Utilities
javax.bluetooth	Standard Bluetooth classes
javax.microedition.io	J2ME I/O.
javax.microedition.lcdui	J2ME LCD User Interface classes.
javax.microedition.location	Location API
lejos.gps	GPS Parsing
lejos.keyboard	Support for SPP keyboards
lejos.localization	Localization support
lejos.math	Extra Math classes for leJOS
lejos.navigation	Navigation classes.
lejos.nxt	Access to NXT sensors, motors, etc.
lejos.nxt.addon	Access to third party and legacy RCX sensors, motors and other hardware not included in the Lego NXT kit
lejos.nxt.comun	NXT communication classes
lejos.nxt.debug	Debugging thread classes

1.2.5.- Alternatives for leJOS project

leJOS project is one of the options to develop software for robots which use Lego Mindstorms NXT as hardware platform but exists others alternatives.

Alternatives for leJOS project:

1. NXT-G
2. NXC/NBC
3. RobotC
4. Plua
5. NXTOSek

1.2.6.- To do list for leJOS project

If you like leJOS project, you could collaborate with the developer team in the following areas:

1. Artificial Intelligence
 - a. Neural Networks
 - b. Qlearning
2. Control systems
3. Fuzzy logic
4. Mobile services (Add new features to a NXT brick from a mobile phone)
5. Energy projects
 - a. Add power from a PC to NXT
 - b. Add power from a NXT brick to others devices
6. Graphical environment (A similar concept in relation to NXT-G)
7. leJOS simulator
8. Integration technologies

- a. ICE
- 9. XML Support for leJOS
- 10. Finish Junit support for leJOS
- 11. Zigbee support for leJOS
- 12. Better Date support

**I think that existe many areas amazing technologies to explore and enjoy.
If you collaborate, you will improve the whole solution.**

1.3.- Summary

In this chapter I tried to show the following concepts:

- Lego Mindstorms NXT is a platform to learn basic concepts about robotics and artificial intelligence.
- LeJOS project is a platform to develop software for Lego Mindstorms NXT using Java.
- LeJOS project has excellent documentation about the leJOS packages.
- LeJOS project can be used in any project which uses a NXT brick.
- LeJOS project is the unique platform for Lego Mindstorms NXT which is able to manage software in a PC and NXT with the same programming language.
- Every hour with leJOS is a hour with Java. If you learn Java so you can learn OOP techniques and you could develop software for Servers, Desktop, Mobile phones and new devices which include a Java Virtual Machine.

2.- Getting started with leJOS

2.1.- Introduction

LeJOS project can be installed in the most important operating systems as Windows (XP, Vista), Linux (Ubuntu) and Mac OS

In this chapter you will learn:

1. How to install leJOS software into your windows operating system
 - a. With leJOS installer
 - b. Manual leJOS installation
2. How to install leJOS firmware into your NXT brick
3. How to install and use leJOS plugin for eclipse IDE.

2.2.- LeJOS Installation with leJOS Installer

2.2.1.- Introduction

If you have chosen the option to install leJOS project into your system with Windows OS then you have to download the following software:

1. Java JDK
2. Latest leJOS release

2.2.2.- Java JDK Installation

LeJOS project is based on Java technology so you need to be installed the latest Java JDK's release.

Use the following URL to download the software:

<http://java.sun.com/javase/downloads/index.jsp>

Note:

This ebook has used the following Java JDK: jdk-6u3-windows-i586-p.exe

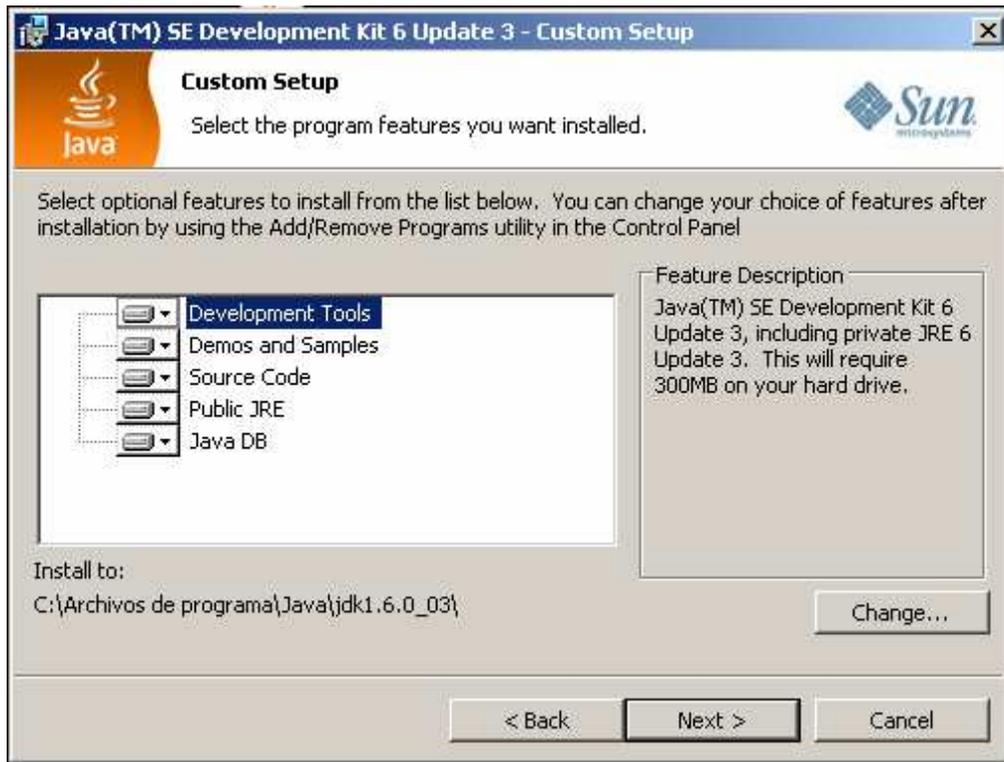
Step 1: Accept the licence



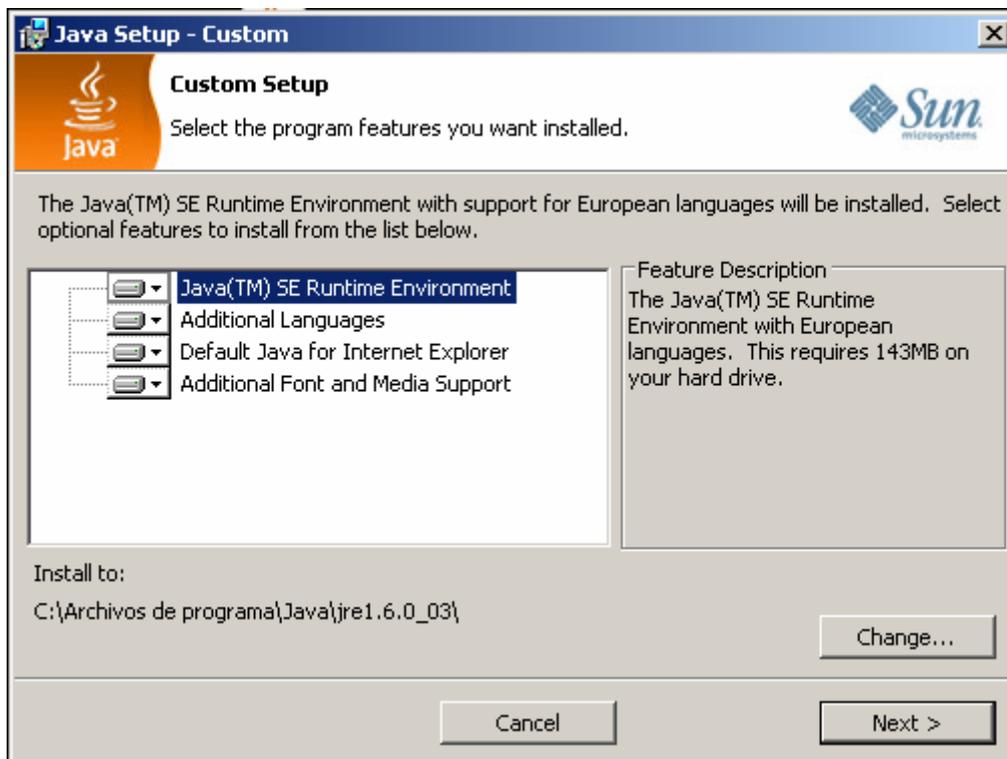
Click in next button show the text about licence agreement and click in Accept:



Paso 2: Choose the components to install
 Choose all components and click in next button. In this moment, it will install JDK components that you choose.



Once the installer finished that process, you will see another screen where you will see others components to install. Choose all options and click in next button:



When the installer finish the second installation process then you have finish the JDK Installation on your computer.



2.2.3.- Checking your J2SE Installation

Once you have installed J2SE SDK on your computer, it is necessary to check that you can compile and execute any java program.

Open a Shell console on your computer and type the command Java:

- Java: Java command used to execute java programs.
- Javac: Java command used to compile a Java program

2.2.3.1.- Test1: Command java

The reason to make the first test is due to you need to check that your operating system recognize the command java which is used to execute java programs. If the shell console returns the options to use the command then the test is a success.

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\juanantonio.breña>java
Usage: java [-options] class [args...]
           (to execute a class)
 or java [-options] -jar jarfile [args...]
           (to execute a jar file)

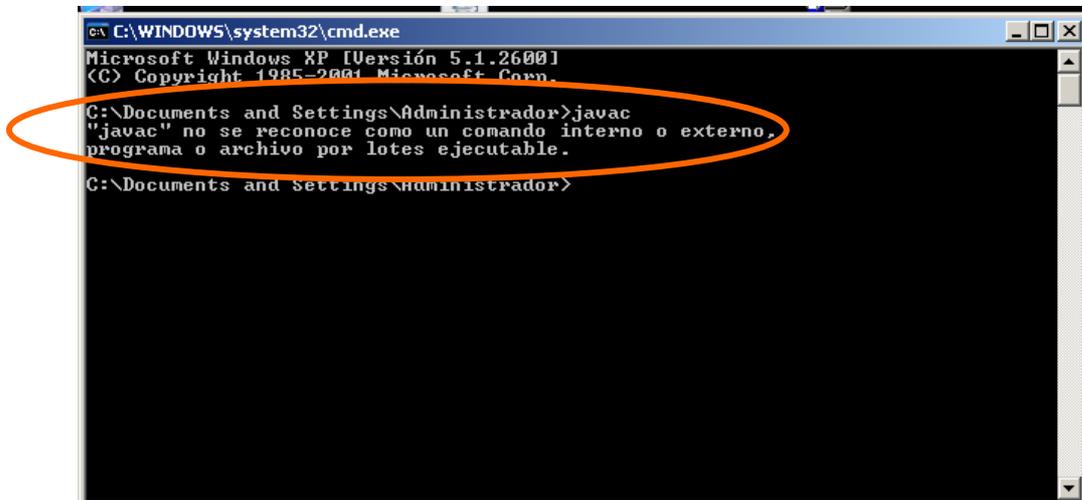
where options include:
  -client          to select the "client" VM
  -server          to select the "server" VM
  -hotspot         is a synonym for the "client" VM [deprecated]
                   The default VM is client.

  -cp <class search path of directories and zip/jar files>
  -classpath <class search path of directories and zip/jar files>
                   A ; separated list of directories, JAR archives,
                   and ZIP archives to search for class files.
  -D<name>=<value>
                   set a system property
  -verbose[:class[:gc[:jni]]
                   enable verbose output
  -version         print product version and exit
  -version:<value>

```

2.2.3.2.- Test2: Command javac

The second test is necessary to know if your operating system recognizes the command javac which is used to compile your programs. Type javac on your console and see the message.



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrador>javac
"javac" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Documents and Settings\Administrador>

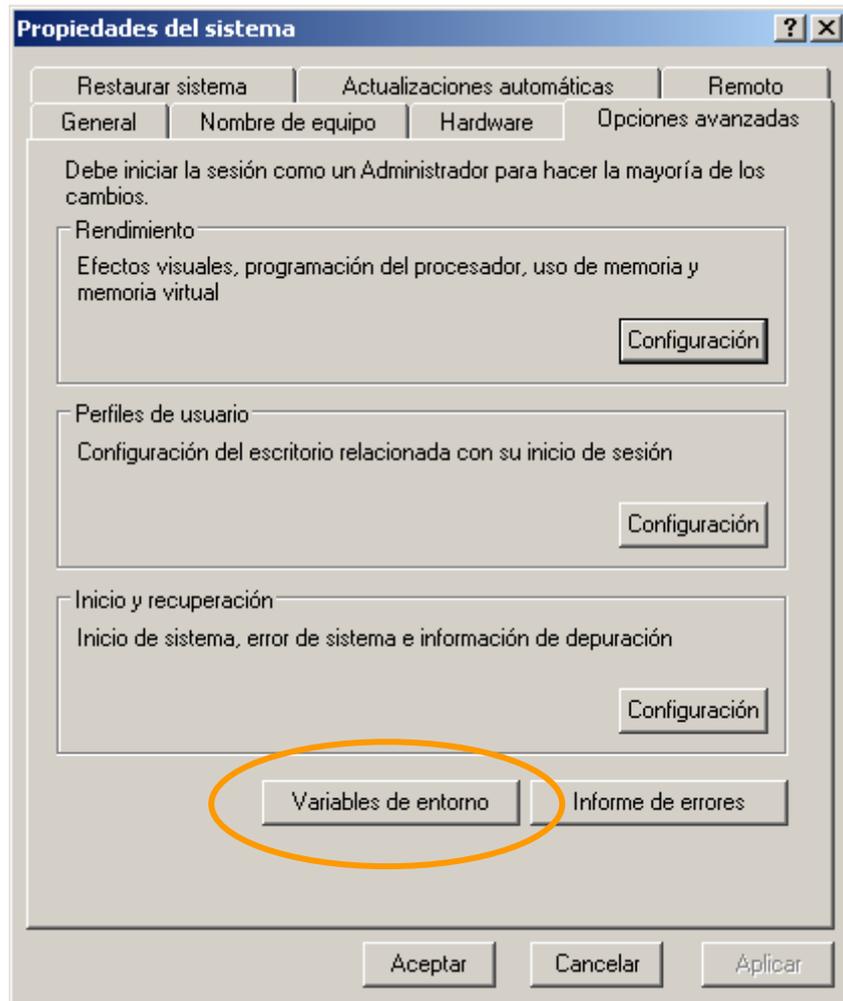
```

If your system says that your command doesn't understand the command, then you have to update environment variables in your system.

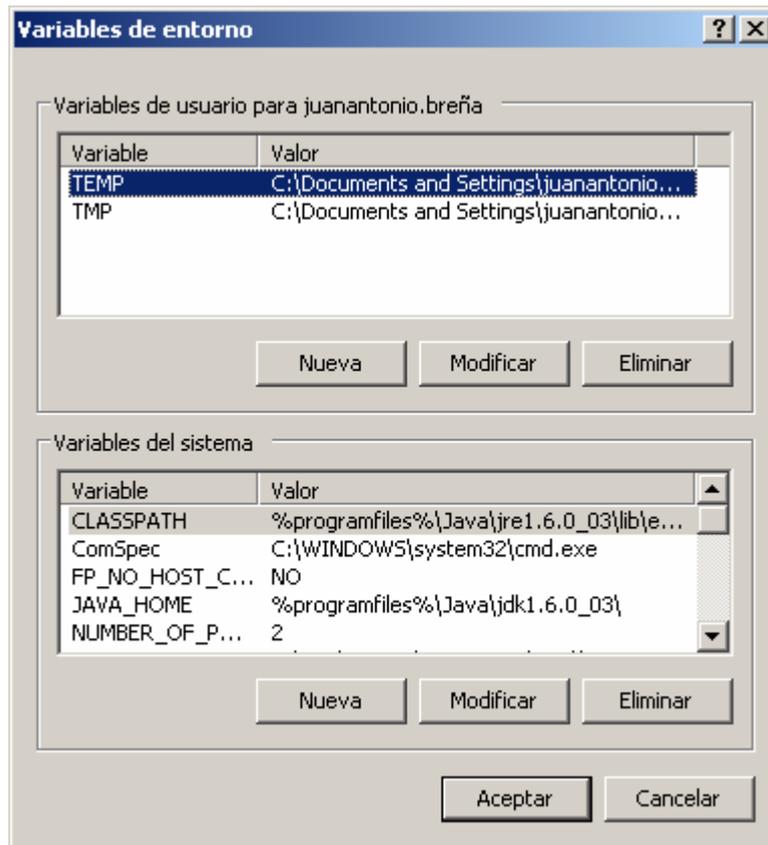
Right click in your PC Icon and select properties. Click in the tab "advanced options"



On the tab "Advanced options" click in the button "Environment variables"



When you click on this window you will see a new window where you will have to update the variable path. This path is really critic because you say windows what command you could execute directly from a Shell console.

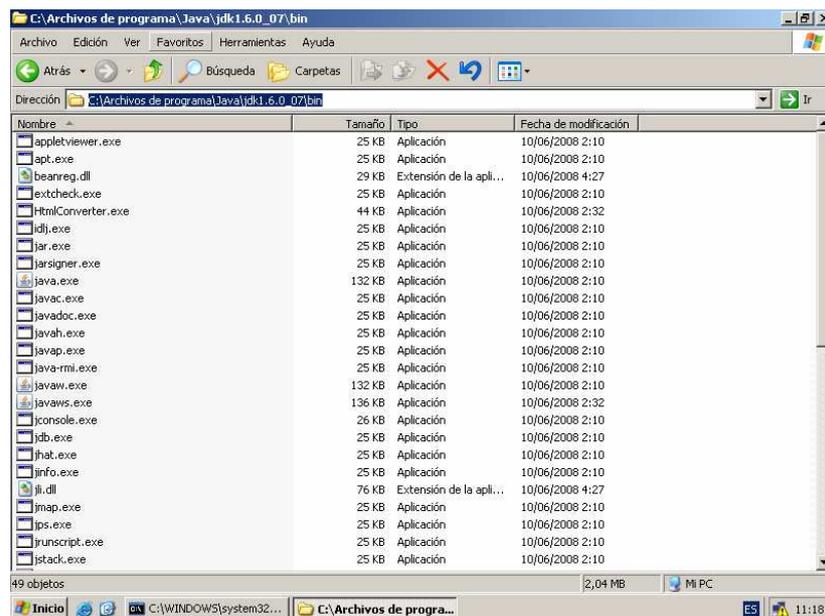


Path variable is located on System variables area. Find the variable path and click and update button. Path variable could have many statements due to it is used by several applications. In a new clean system, you should have the following content:

path:

%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem

To update path variable, find on your computer where is located J2SE SDK.



In this case, the path is:

C:\Archivos de programa\Java\jdk1.6.0_07\bin\;

Once you know the path, add the path at the end of the content of the system variable path:

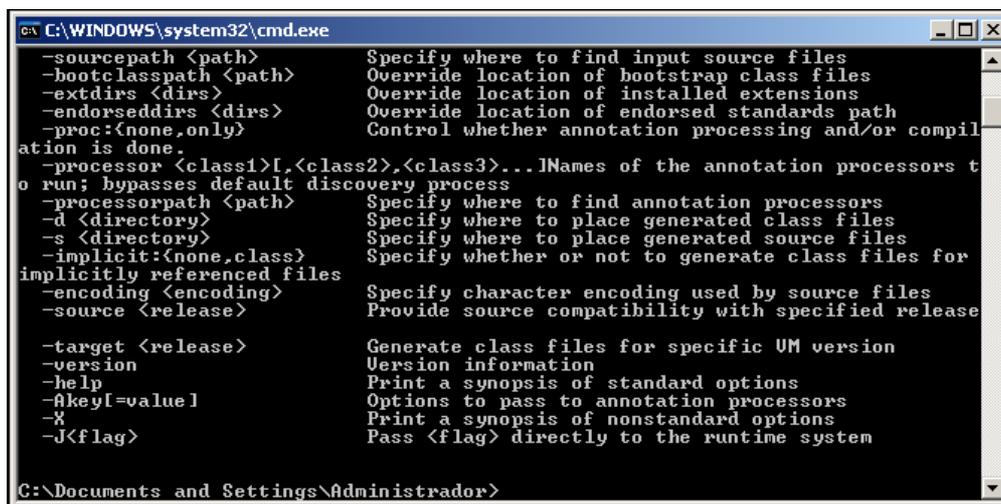
%programfiles%\Java\jdk1.6.0_07\bin\;

And the result is:

New path:

%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;
%programfiles%\Java\jdk1.6.0_07\bin\;

Once you have made the changes, **reboot** the system and check again the command javac.



```

C:\WINDOWS\system32\cmd.exe
-sourcepath <path>          Specify where to find input source files
-bootclasspath <path>      Override location of bootstrap class files
-extdirs <dirs>            Override location of installed extensions
-endorseddirs <dirs>      Override location of endorsed standards path
-processor <none,only>     Control whether annotation processing and/or compilation is done.
-processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
-processorpath <path>      Specify where to find annotation processors
-d <directory>            Specify where to place generated class files
-s <directory>            Specify where to place generated source files
-implicit:<none,class>    Specify whether or not to generate class files for implicitly referenced files
-encoding <encoding>      Specify character encoding used by source files
-source <release>         Provide source compatibility with specified release
-target <release>         Generate class files for specific VM version
-version                   Version information
-help                       Print a synopsis of standard options
-Akey[=value]             Options to pass to annotation processors
-X                         Print a synopsis of nonstandard options
-J<flag>                  Pass <flag> directly to the runtime system

C:\Documents and Settings\Administrador>

```

If you notice that you see the options for the command javac, then the test was a success and you have finished the installation. Now your computer is able to develop Java Software.

In this point of the installation, you have the basic java tools installed and configured. This is the moment to install leJOS software.

2.2.4.- Installing leJOS project with leJOS installer

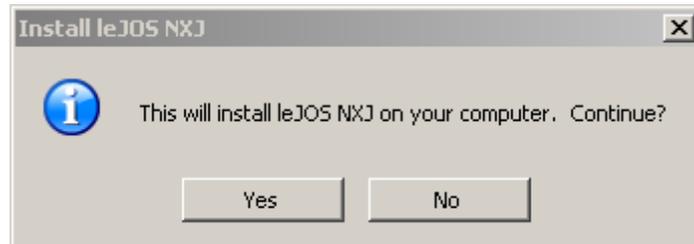
LeJOS installer is a great advance in the Project because it reduces installation's time.

The installer does the following tasks:

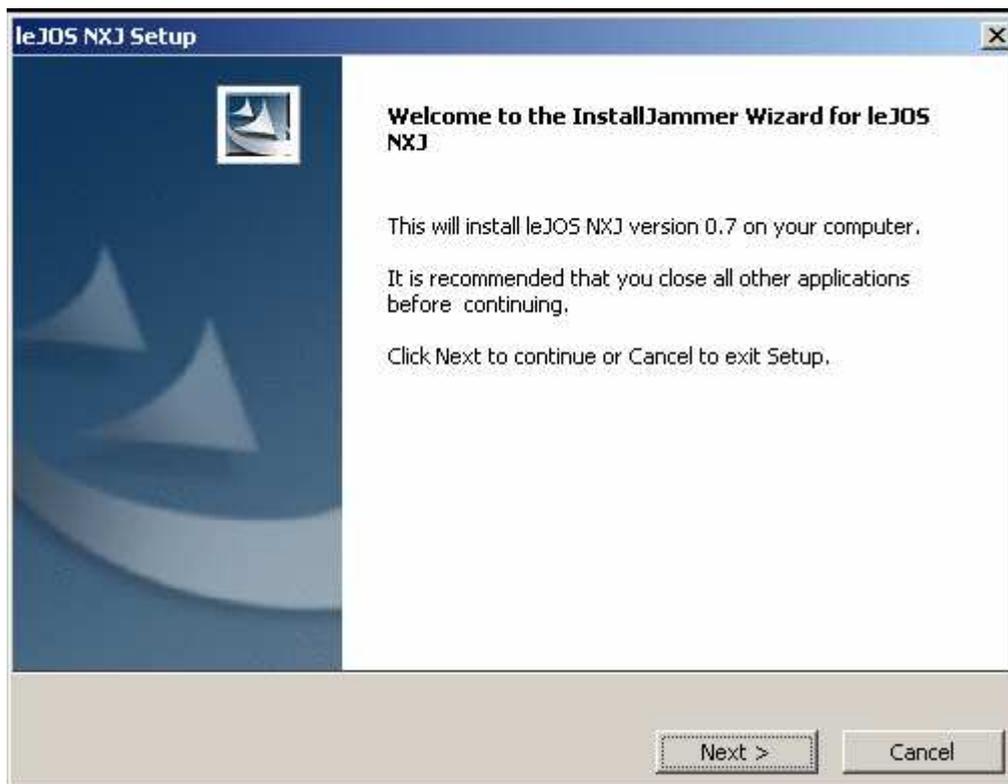
1. Install NXT driver. This software allows windows OS recognize a NXT brick connected to a PC.
2. LibUSB Installation. This software allows connect a NXT brick with a PC using USB connection.
3. LeJOS library installation. This task has the mission to install libraries, shell commands, utilities and examples about leJOS project.

Besides the installer do others tasks as add variables to the operating systems.

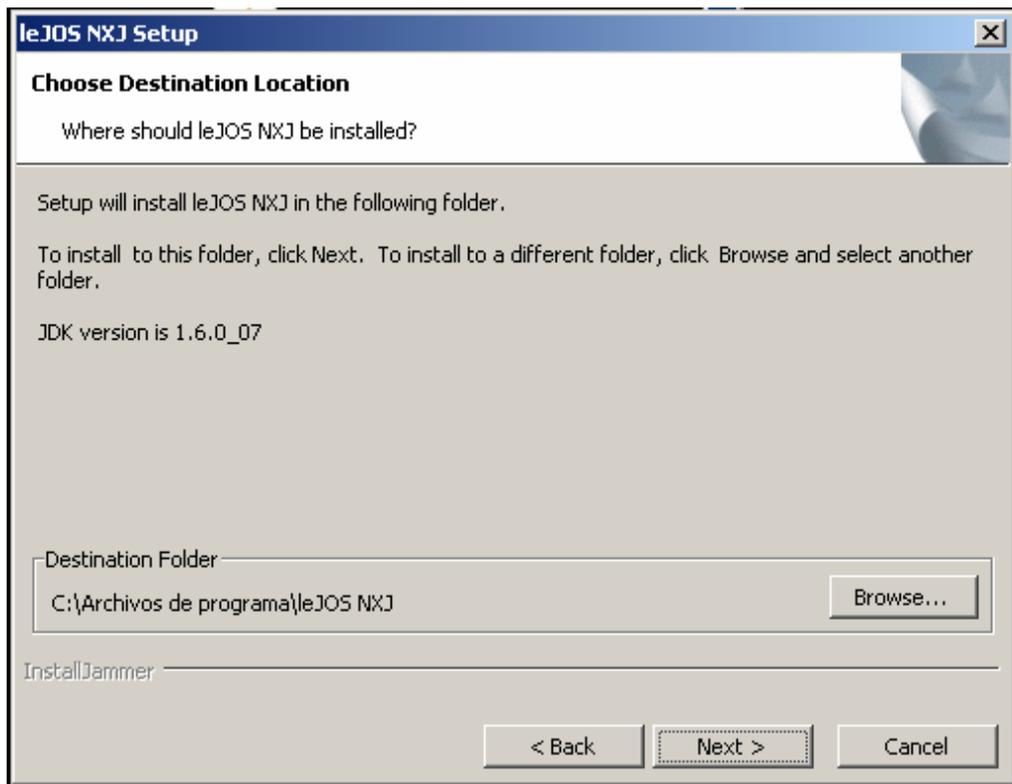
To use the installer, download latest leJOS release from leJOS website:
<http://www.lejos.org>
Once you have downloaded the installer, execute it.



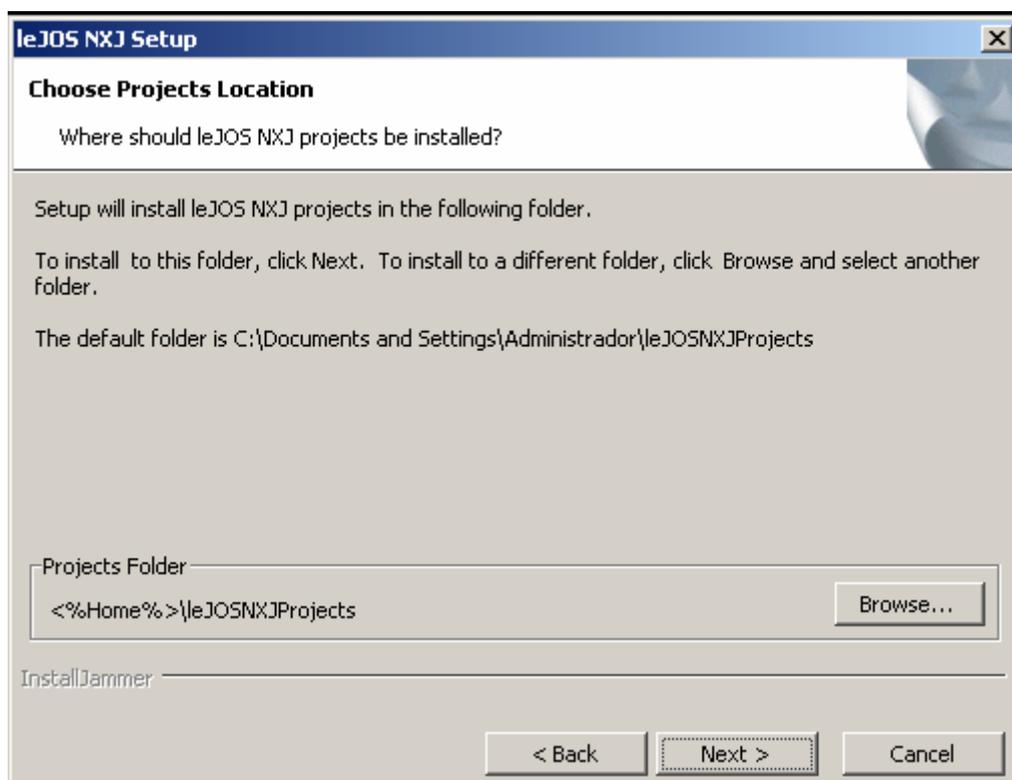
Click in the button "Yes" to begin the process:



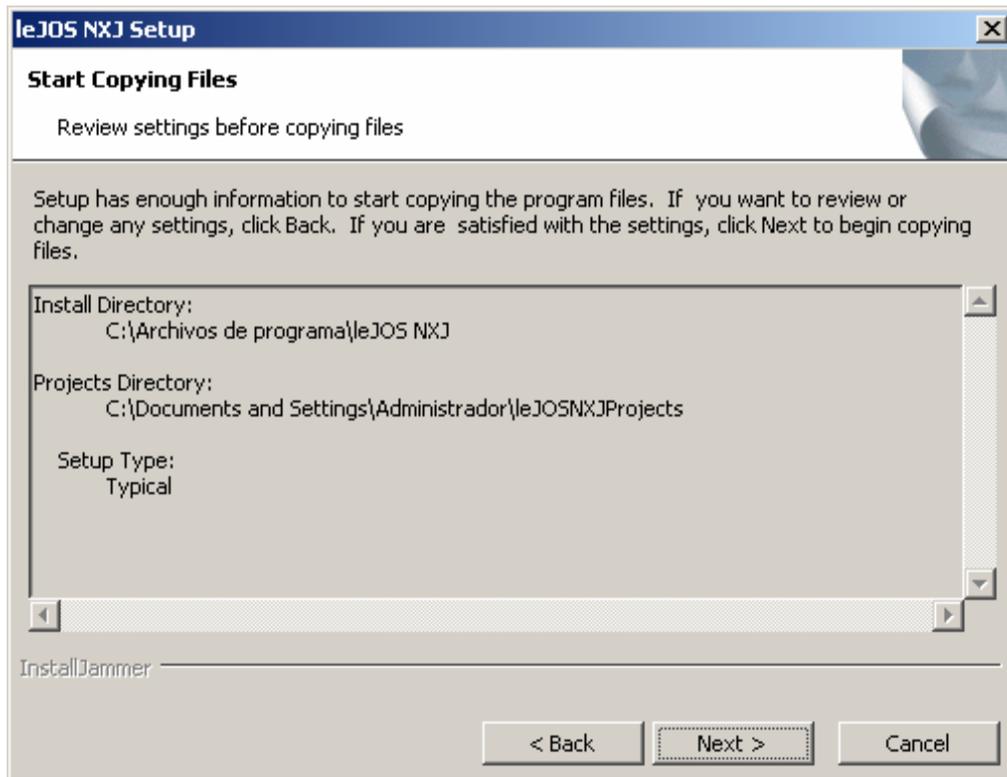
Click in the button "Next" to continue



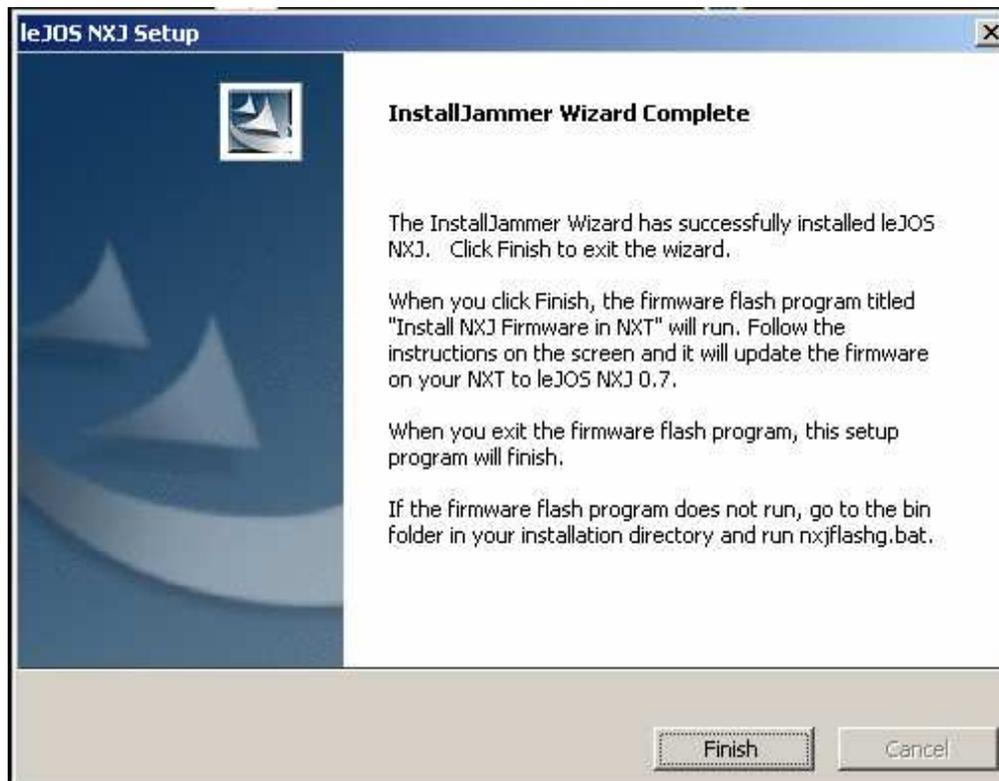
System detect latest JDK installed into the system. Click in the button "Next" to continue.



Installer ask you a path where you will add your leJOS projects. Click in the button "Next" to continue



Verify if all is correct and click in the button "Next" to install leJOS software.



2.3.- Manual Installation

2.3.1.- 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

2.3.2.- 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:

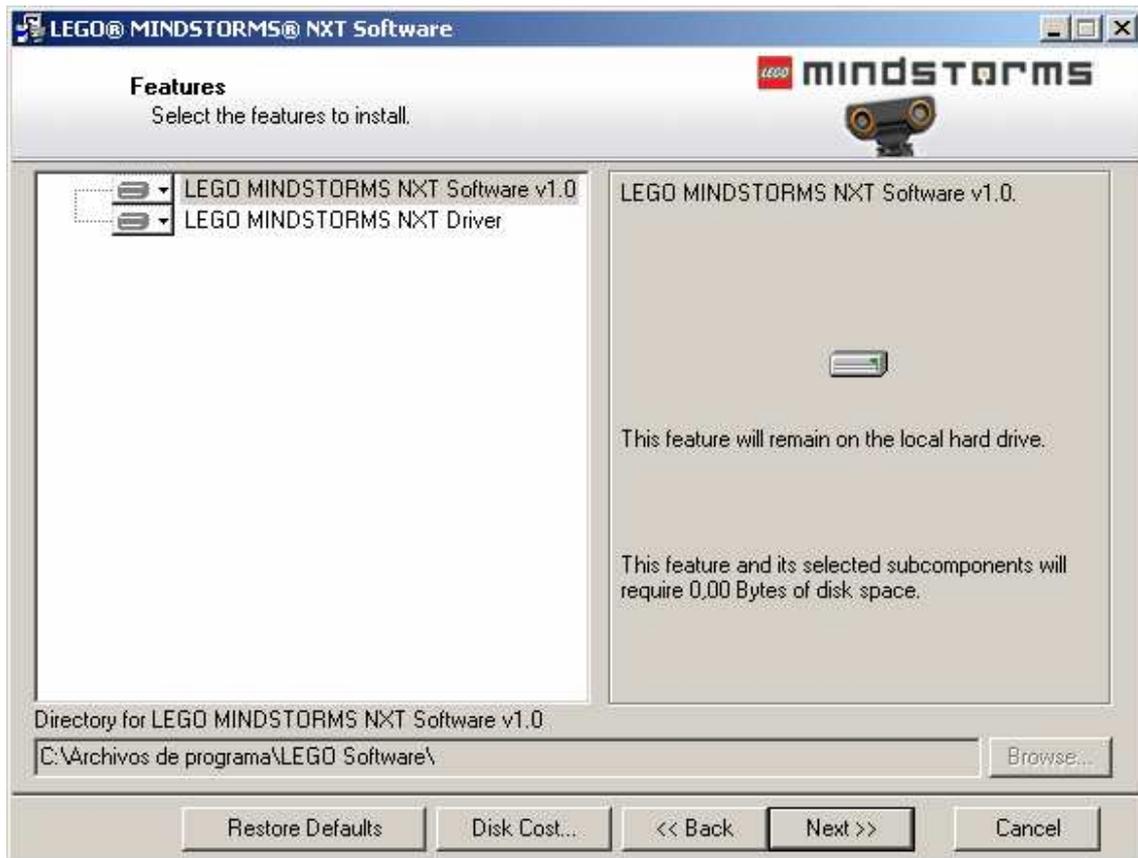
First Step: Execute your CD in your Desktop PC or Laptop with Windows Operating System. The installer will make a question about the language.



Step2: Select the components to install.

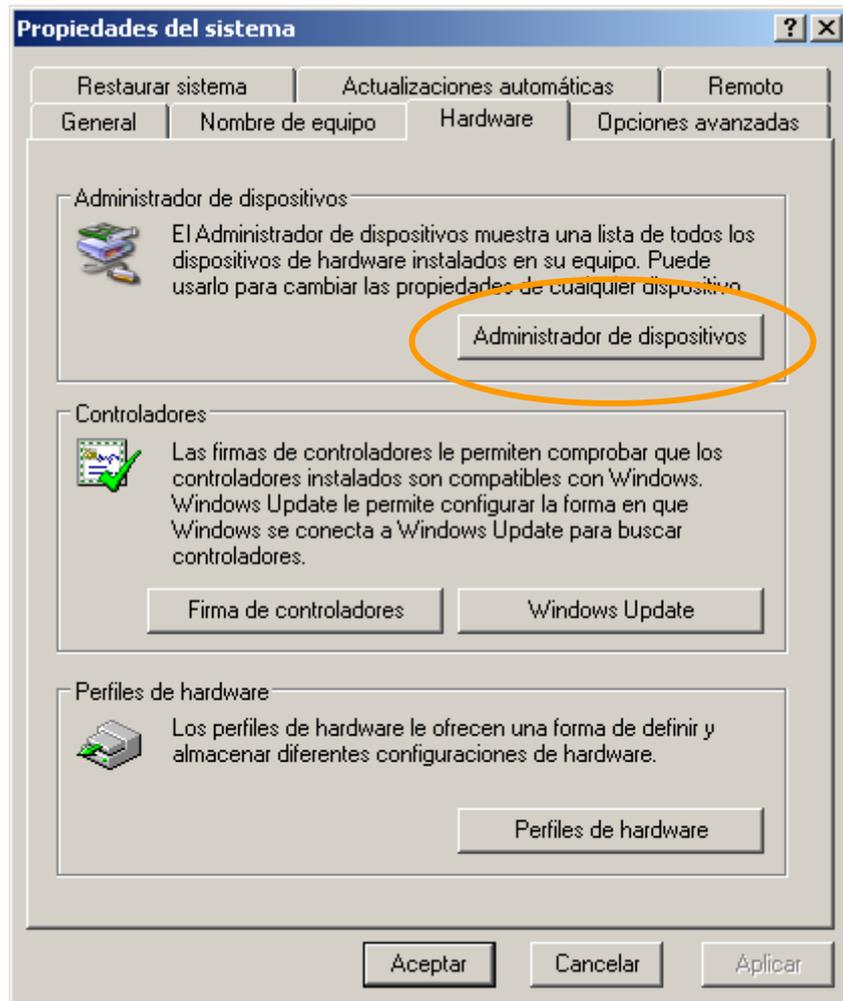


Select all Components.

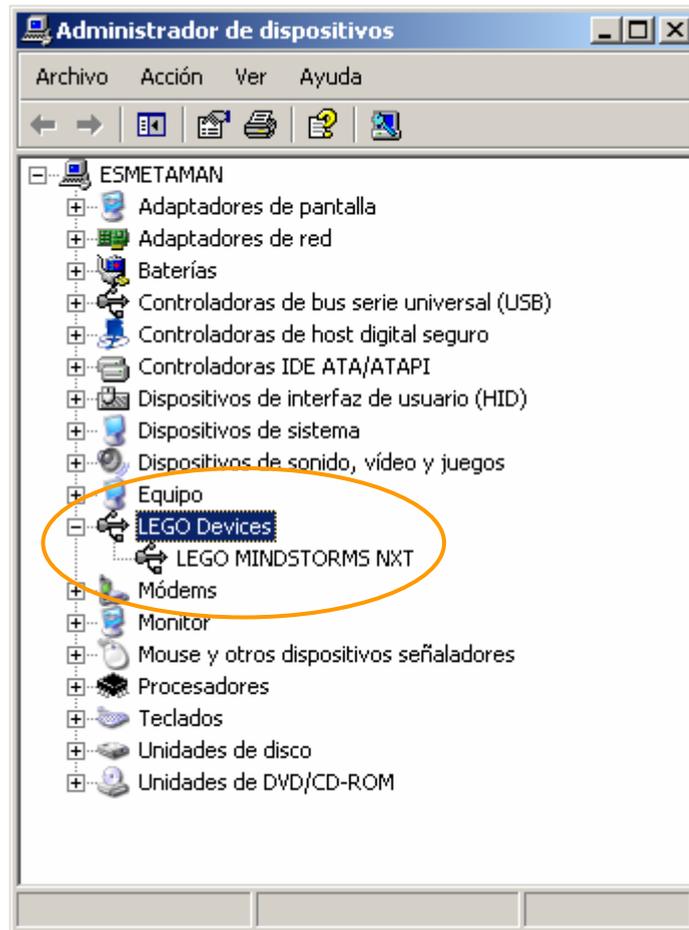


The first time when you connect your NXT Brick with your computer, Windows doesn't recognize your Brick, Windows will think that NXT brick is a USB device. In this moment update the driver. If you update the driver, Windows recognize correctly that you have connected a NXT Brick.

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.

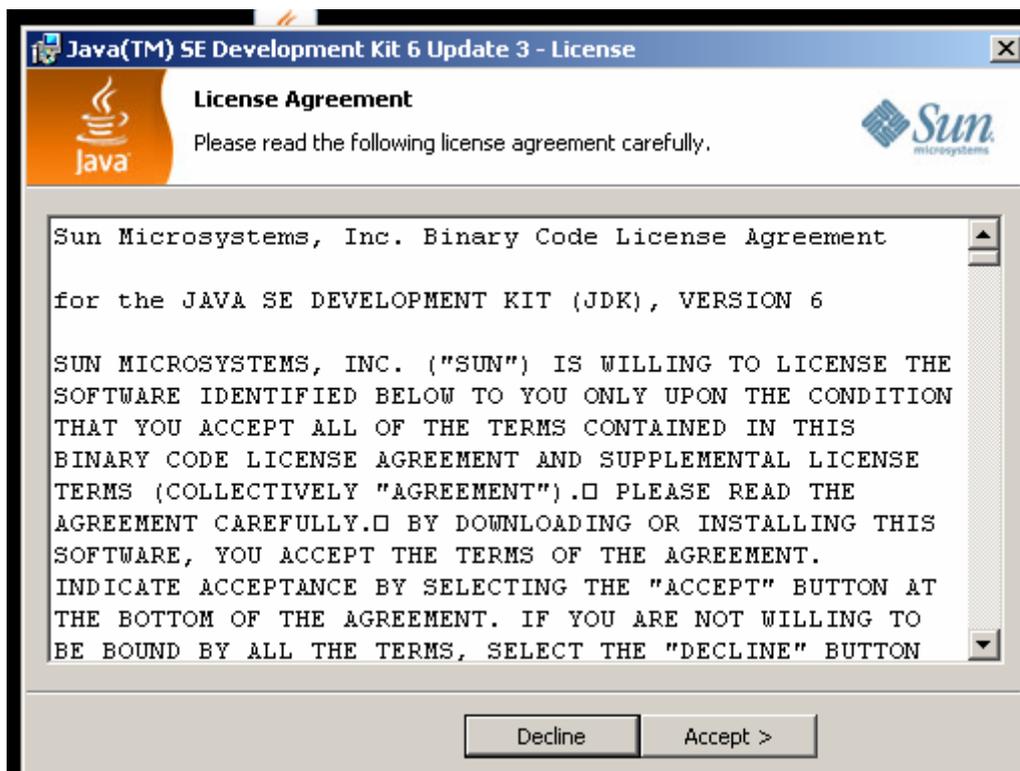
2.3.3.- Java Developer Kit

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.

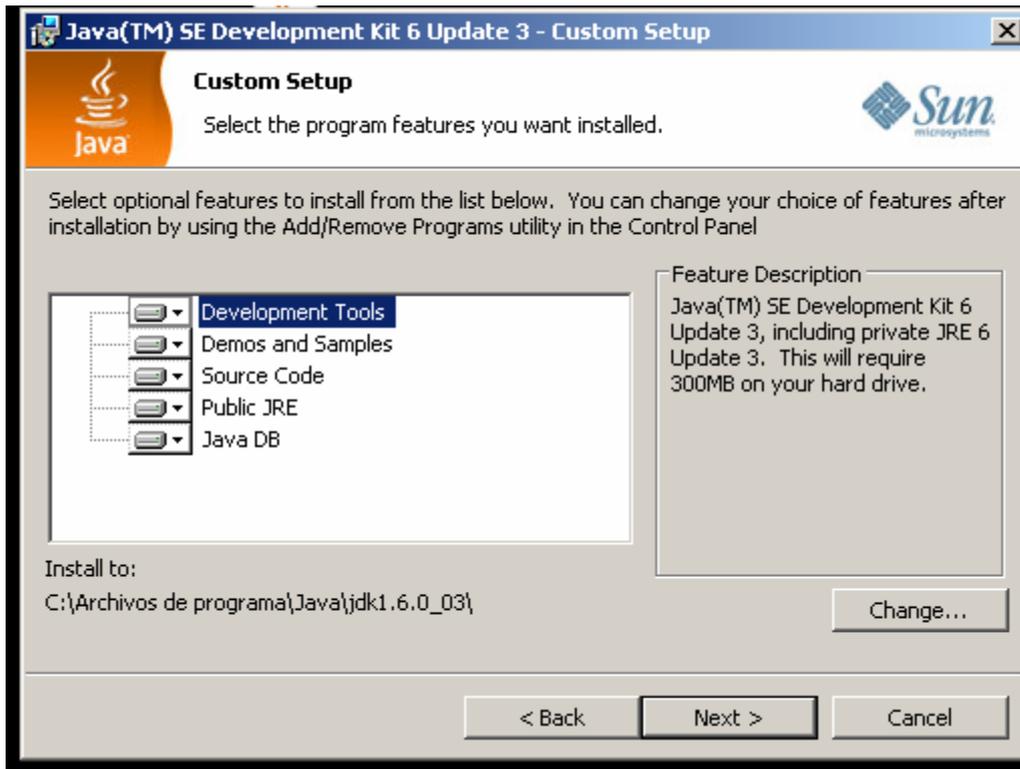
NOTE: To create this document, the installer used is: jdk-6u3-windows-i586-p.exe

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

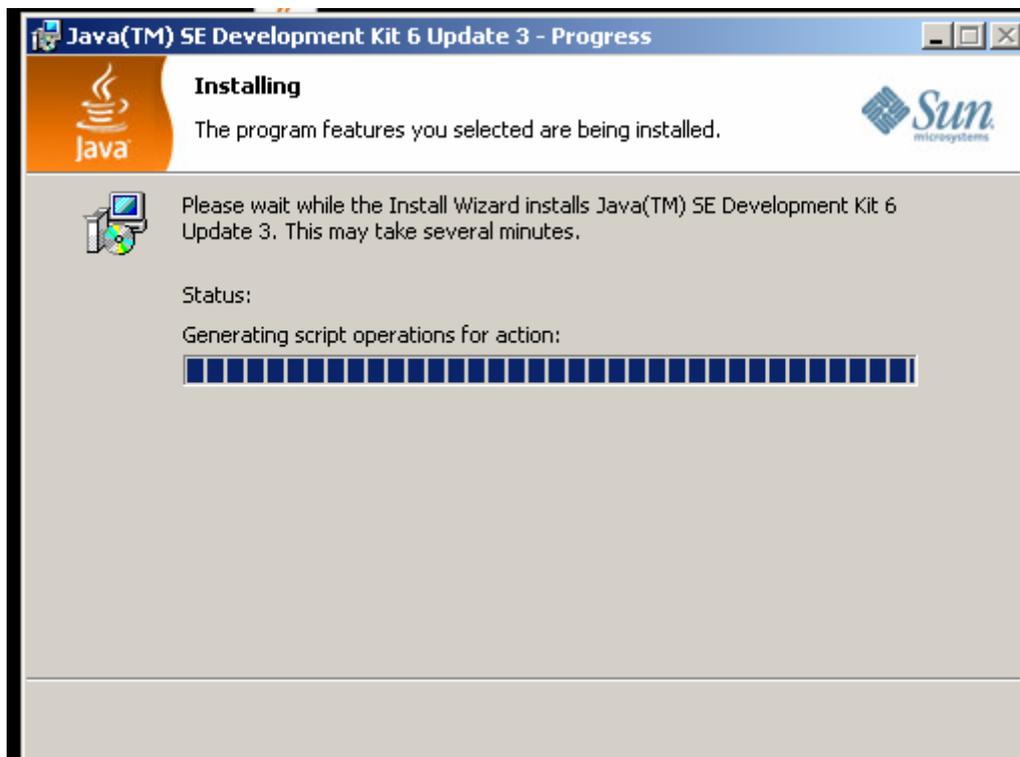
Step1: Accept the licence.



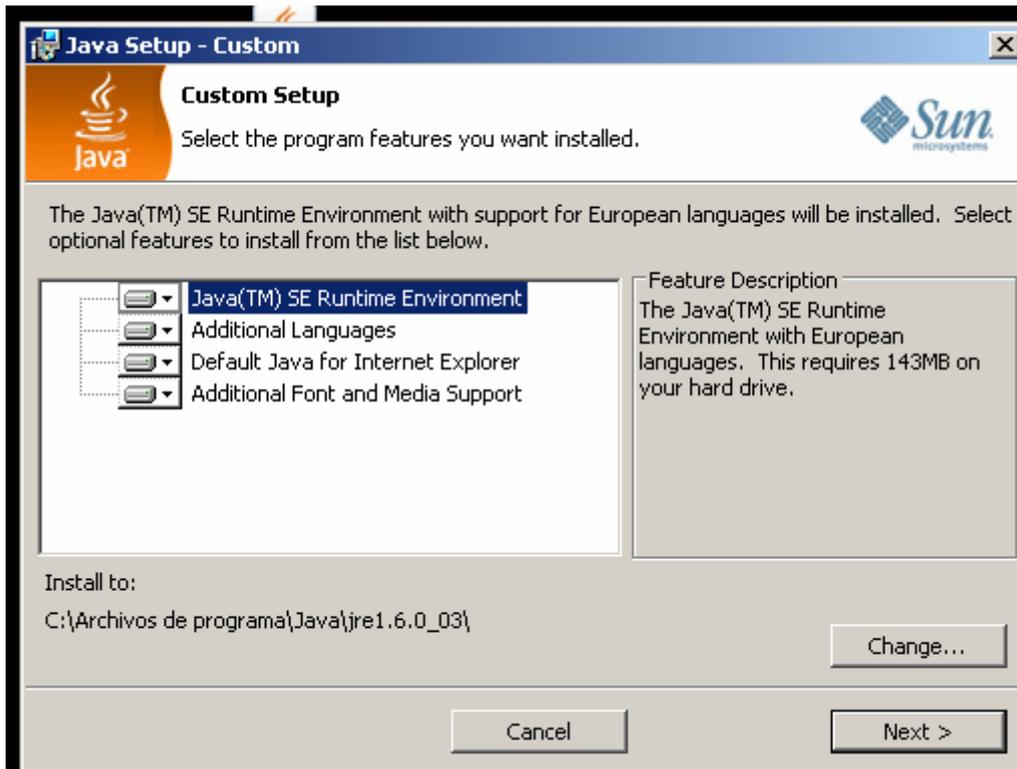
Step2: Select the components to install.



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

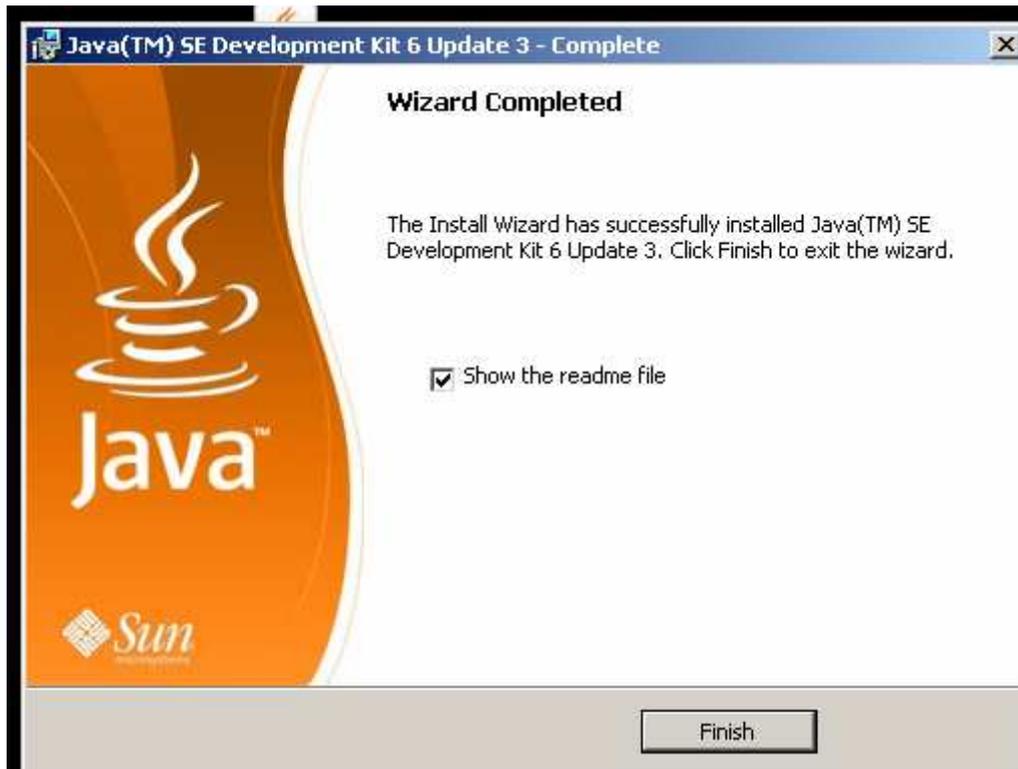


When the installer has finished with the component's installations, this one makes another question about optional features. Select all features.



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:

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\juanantonio.breña>java
Usage: java [-options] class [args...]
           (to execute a class)
 or java [-options] -jar jarfile [args...]
           (to execute a jar file)

where options include:
-client          to select the "client" VM
-server         to select the "server" VM
-hotspot        is a synonym for the "client" VM [deprecated]
                The default VM is client.

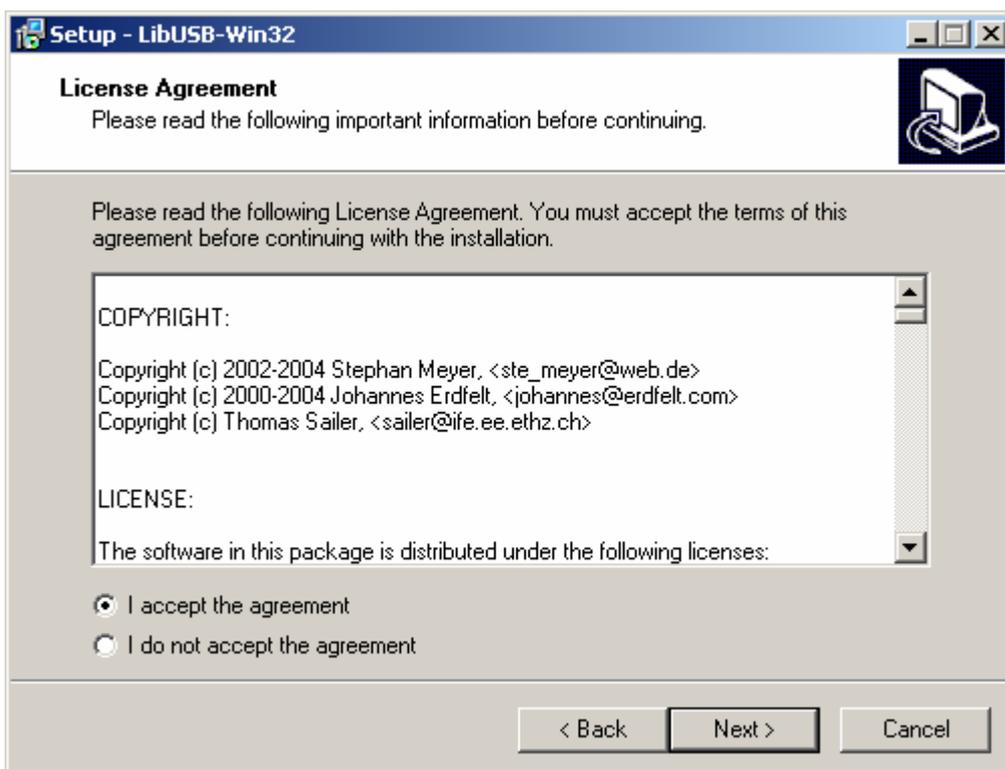
-cp <class search path of directories and zip/jar files>
-classpath <class search path of directories and zip/jar files>
            A ; separated list of directories, JAR archives,
            and ZIP archives to search for class files.
-D<name>=<value>
                set a system property
-verbose[:class!gc!jni]
                enable verbose output
-version        print product version and exit
-version:<value>

```

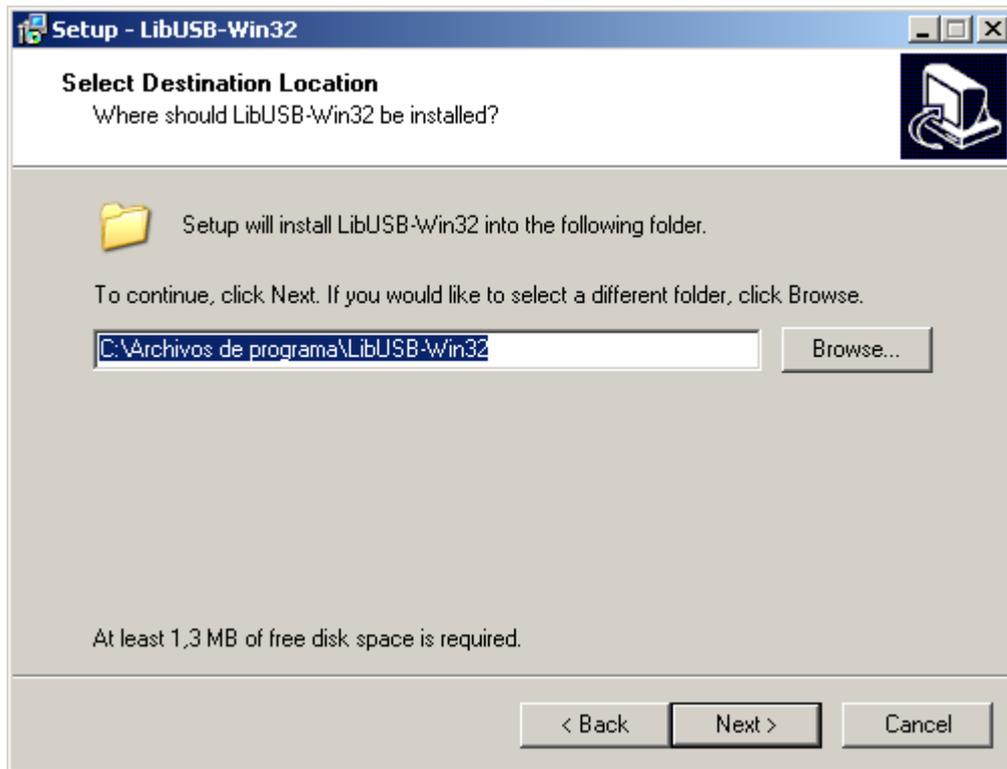
2.3.4.- LibUSB Filter Driver for Microsoft Windows

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.

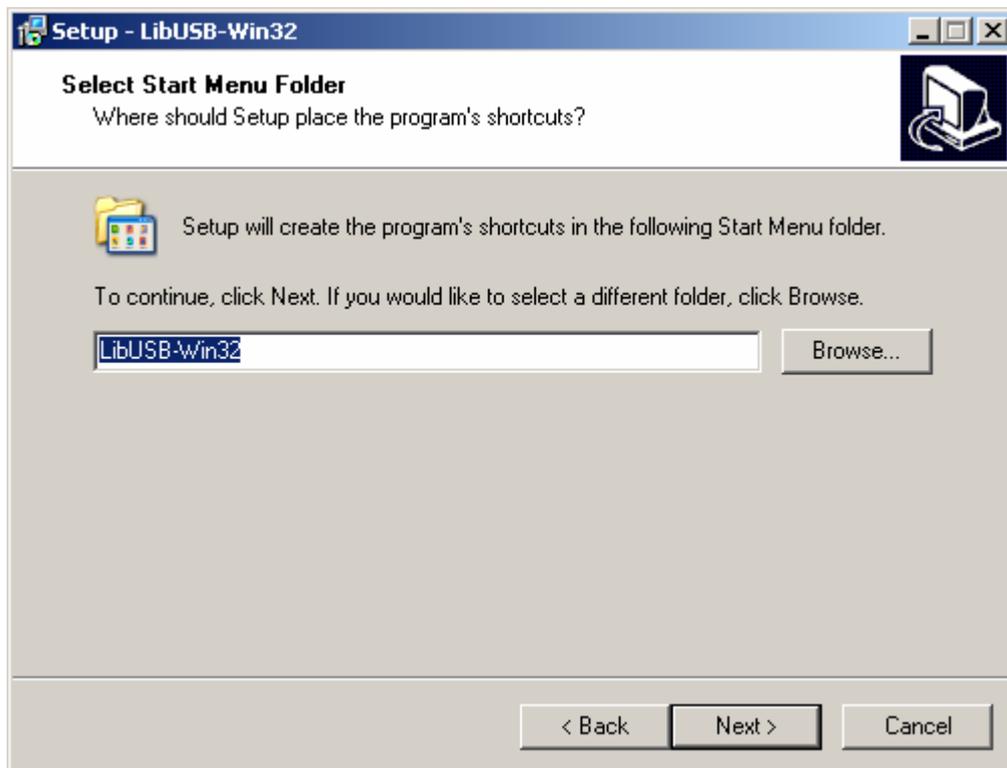
Step1: Accept the licence.



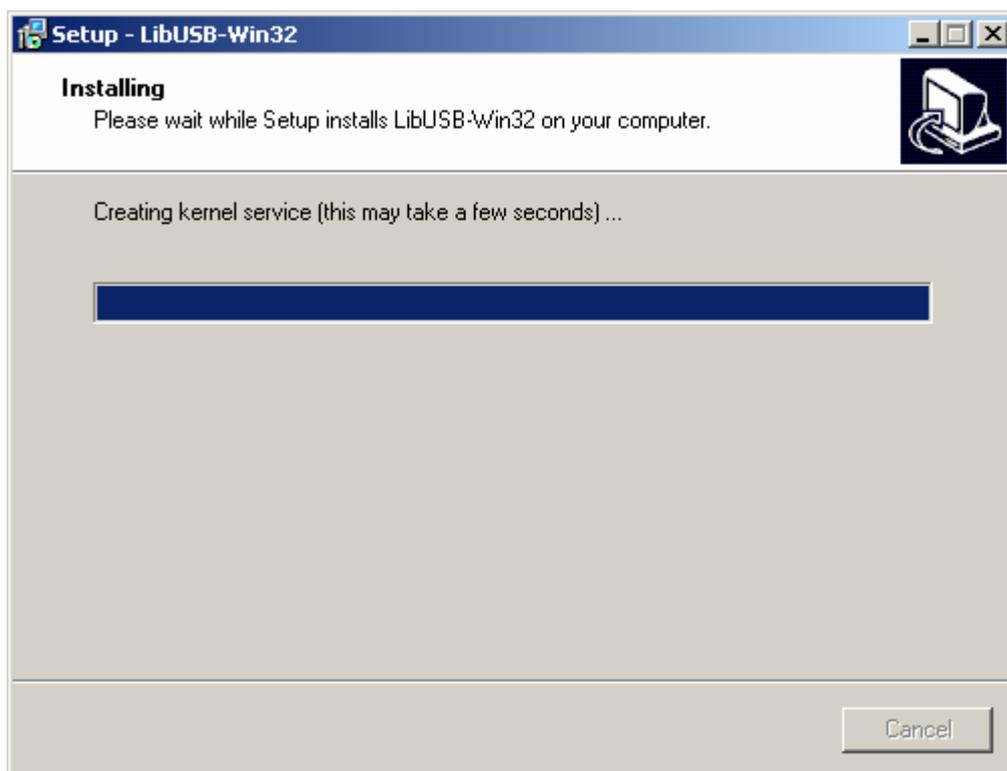
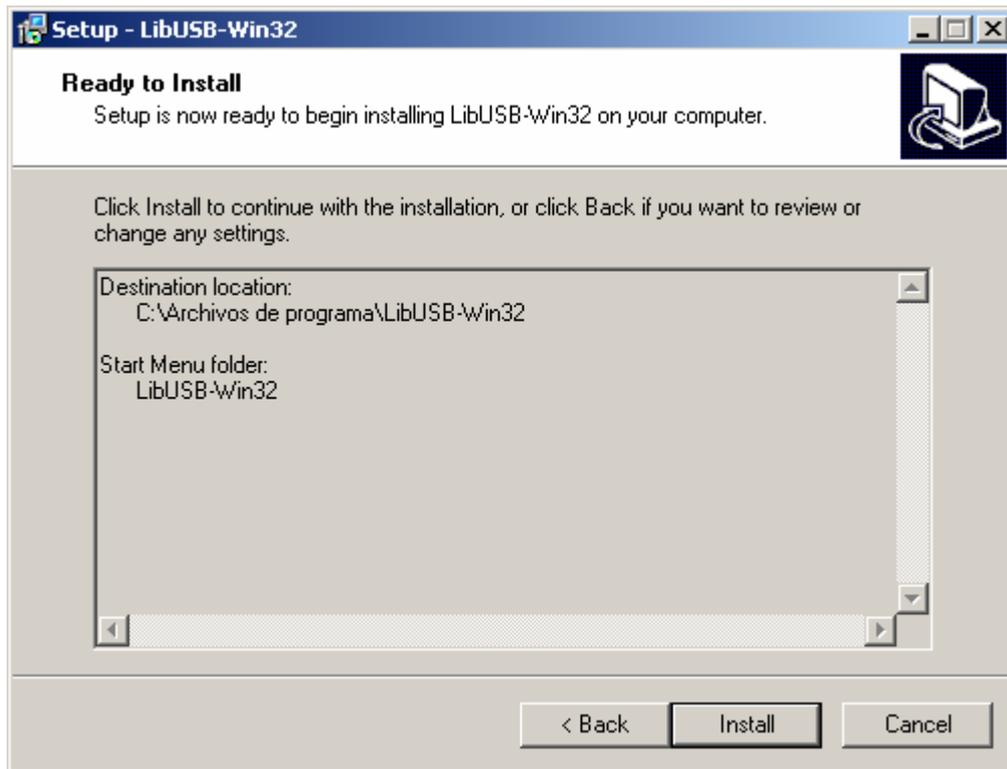
Step2: Select where you want to install the software



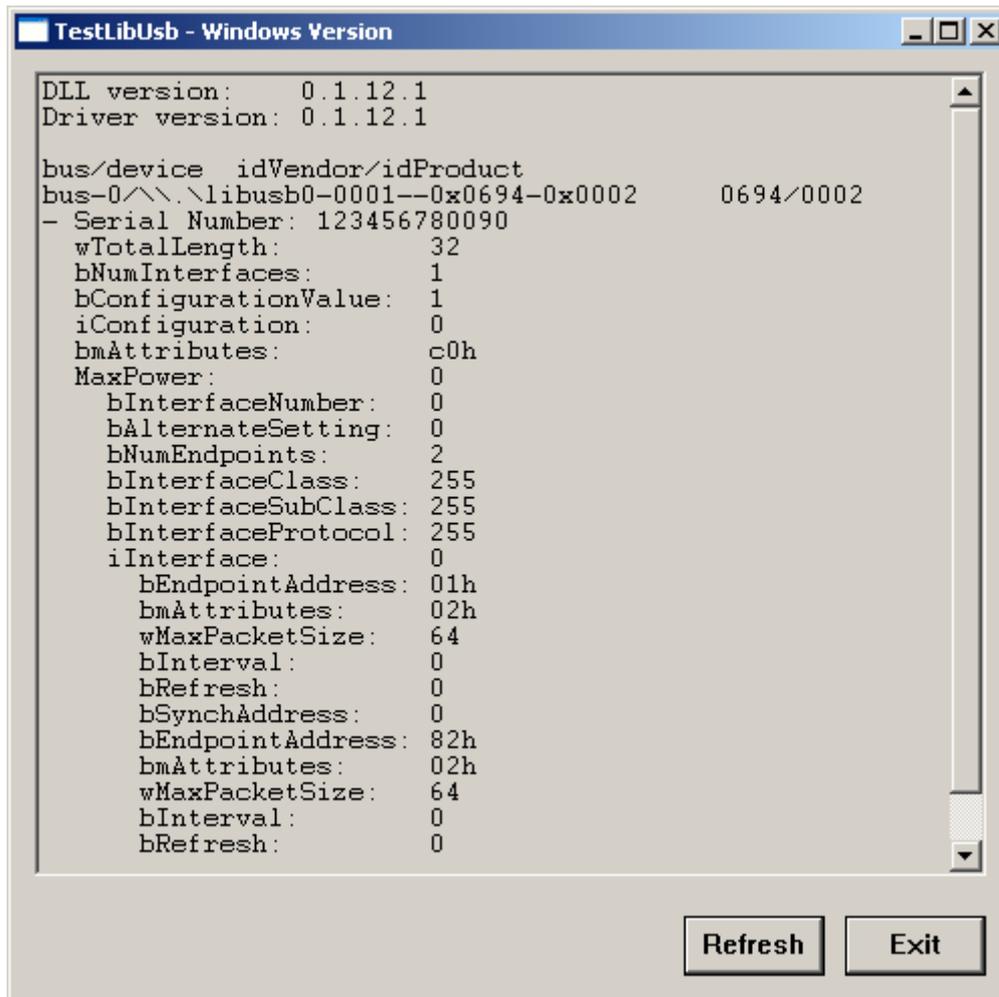
Step3: Choose the place when you will see in Windows Start menu



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:

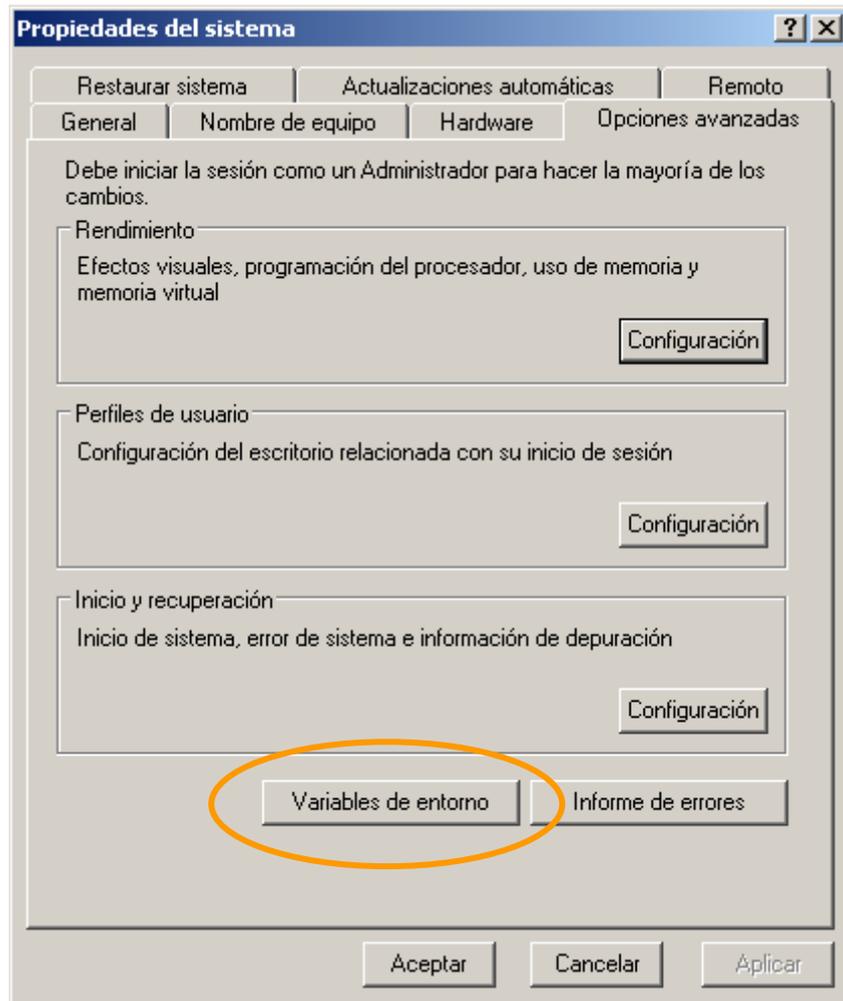


2.3.5.- LeJOS NXJ

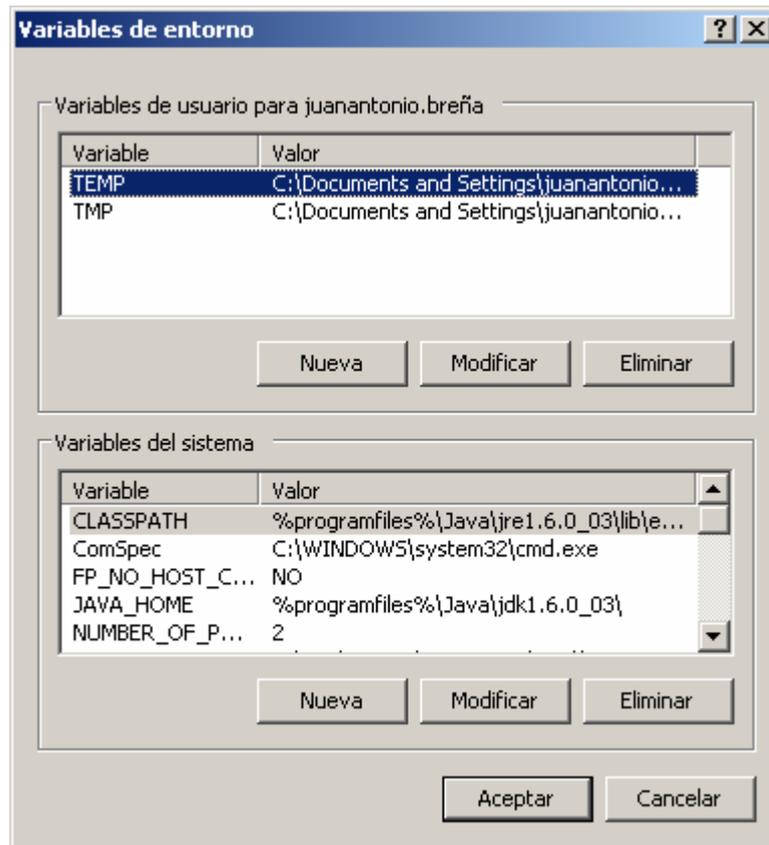
NXJ is a LeJOS project which is evolving along the time with new features. Currently LeJOS Research Team has launched the release 0.5. To download: http://sourceforge.net/project/showfiles.php?group_id=9339&package_id=217619

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

Once you have rebooted the system then your computer is ready to develop software for robots with leJOS.

2.4.- Install leJOS firmware into your NXT brick

2.4.1.- Introduction

When you purchase a Lego Mindstorms NXT kit, NXT brick has a standard firmware but this one it is not able to execute Java programs so it is necessary to replace the firmware by leJOS firmware.

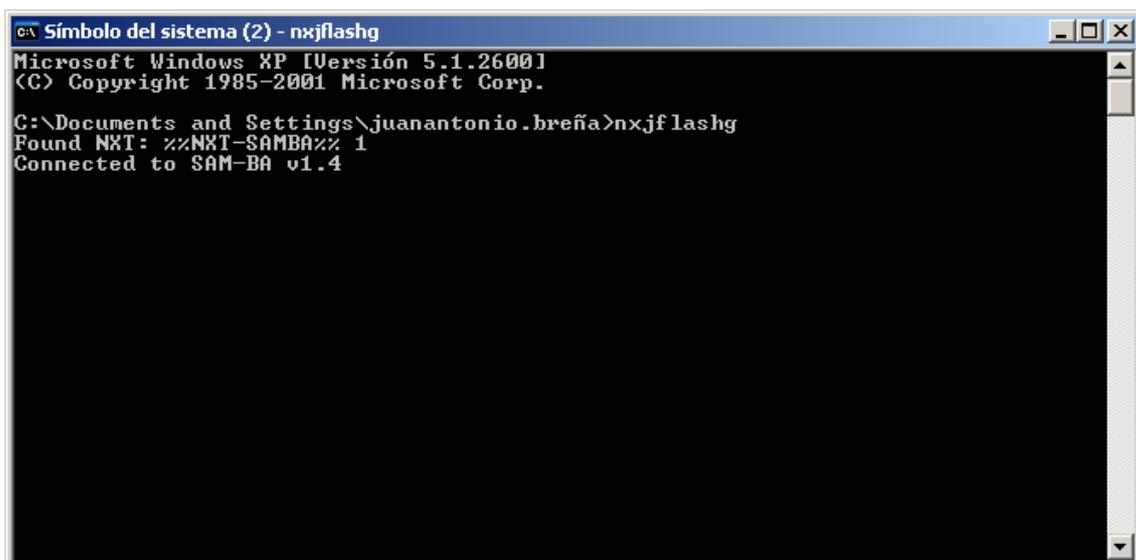
Currently, exists 3 ways to replace the current firmware:

1. Using a graphical utility, nxjflashg
2. Using a shell utility, nxjflash
3. Using a utility from leJOS plugin for eclipse IDE

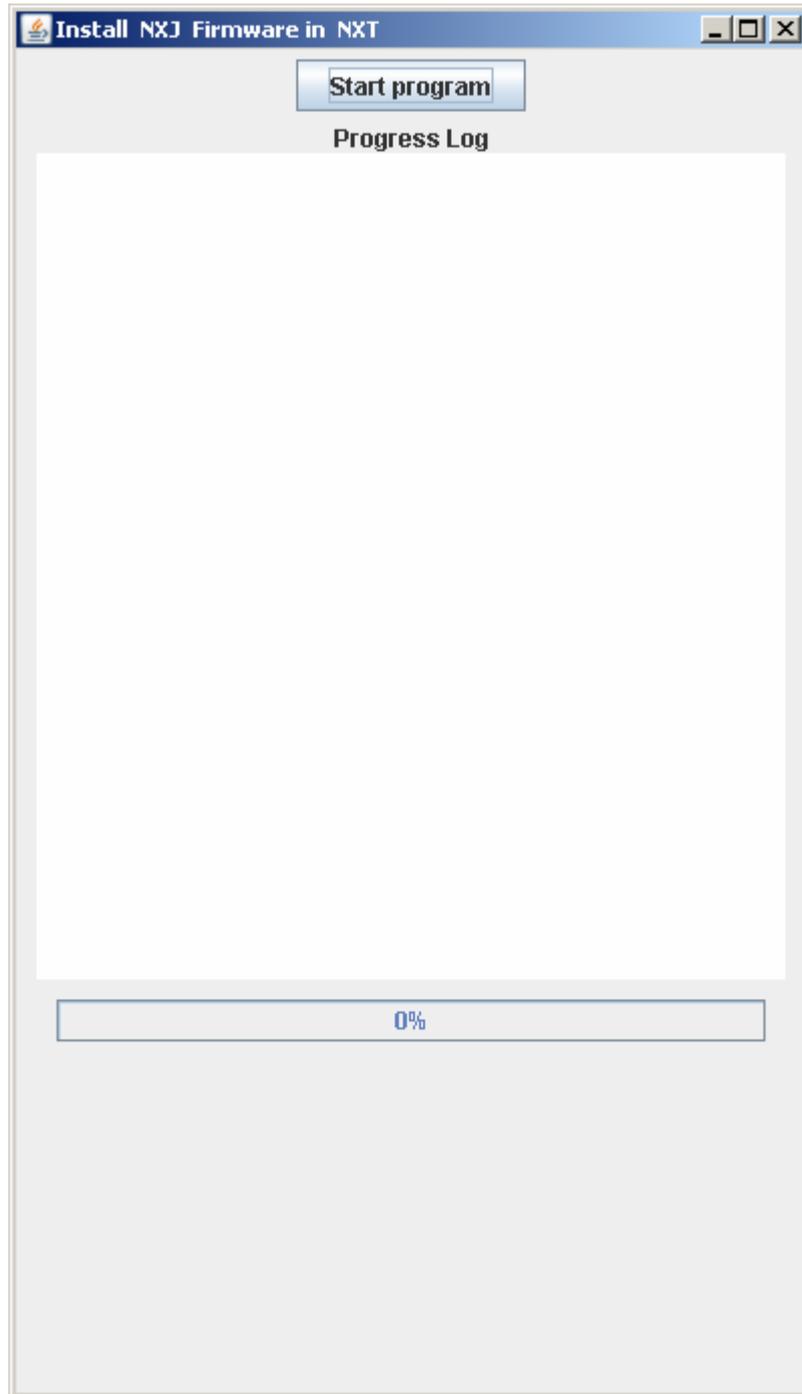
In latest release of the utility nxjflashg you don't need to click in reset button. Now, the way to install/reinstall leJOS firmware is really easy.

2.4.2.- Install leJOS firmware using a GUI

To install leJOS firmware in your NXT brick is really easy. Open a shell console and type the command nxjflashg



Then you will see a window which will help you in the process:



Click in the button start to begin the process. Click in the button "Yes" to erase old files from your NXT brick. Normally the programs installed with previous releases are not compatible with newer firmwares.

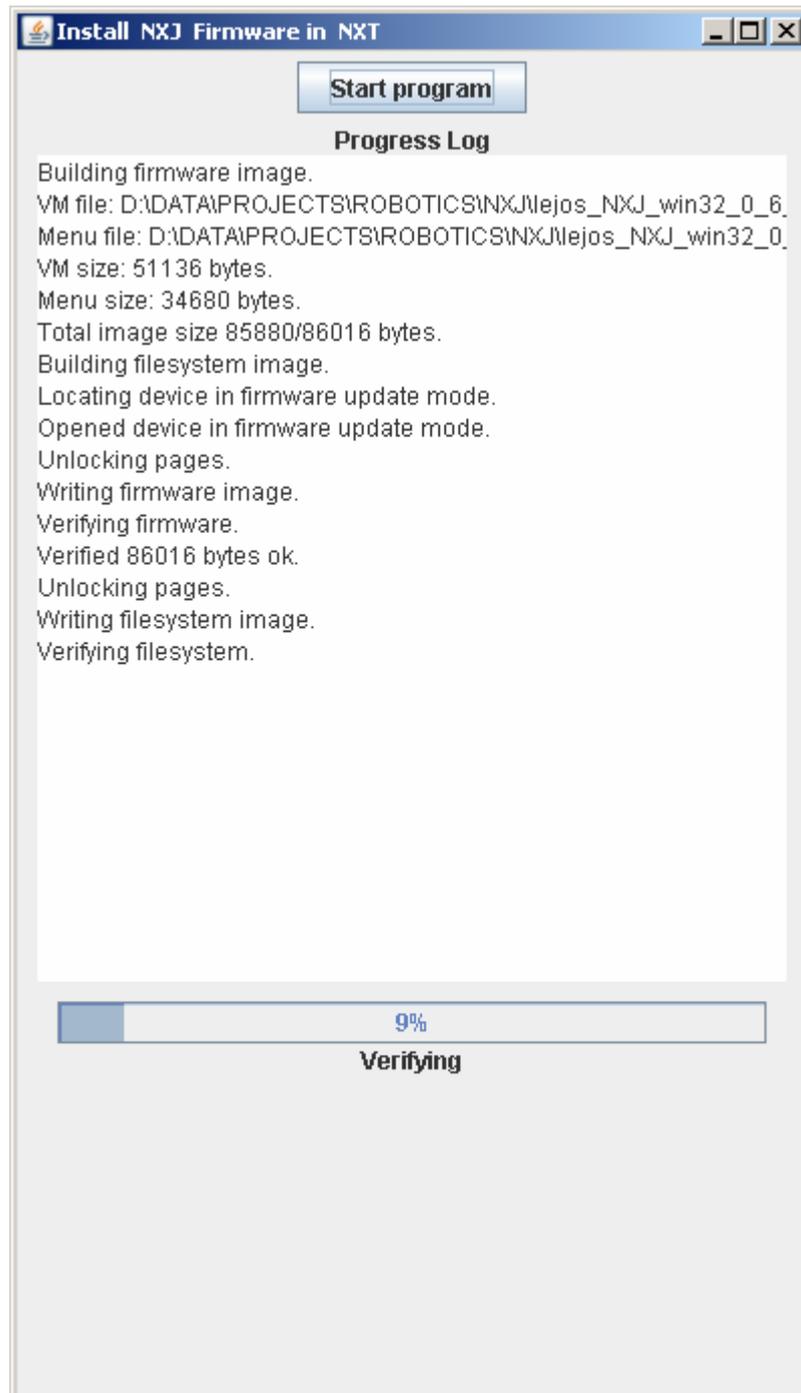


The process installs leJOS virtual machine and menu program. Click in the button "Yes" to verify the process.



In this moment, the utility will do the following tasks:

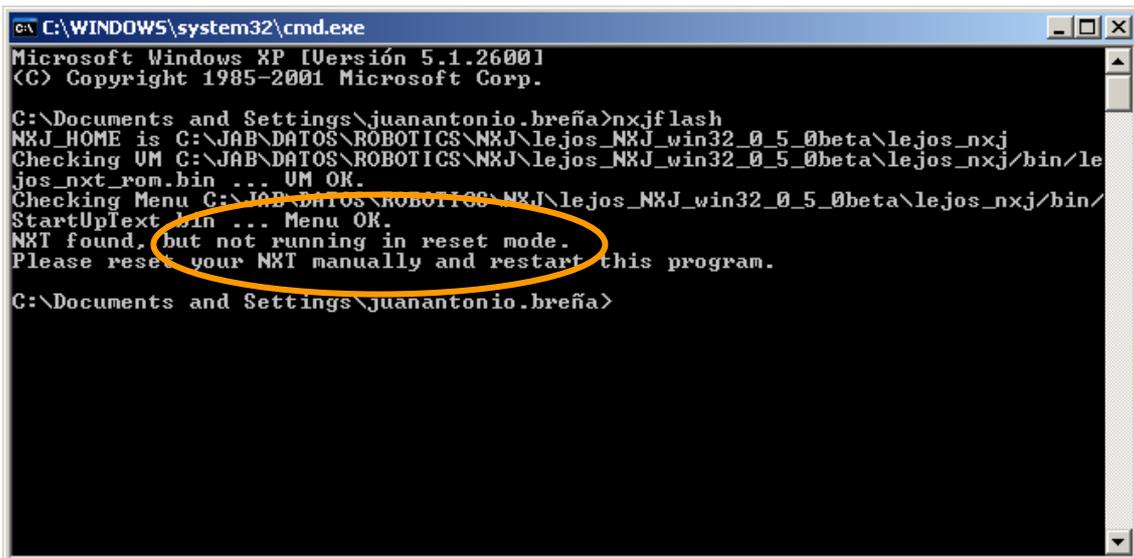
- Write a firmware image
- Write a filesystem image



Once the utility finish the tasks then your NXT brick will have installed leJOS firmware and your NXT brick is ready to execute leJOS programs.

2.4.3.- Install leJOS firmware using a shell console

leJOS firmware can be installed using a simple shell console. Open a shell console in your computer and type the command `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:



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

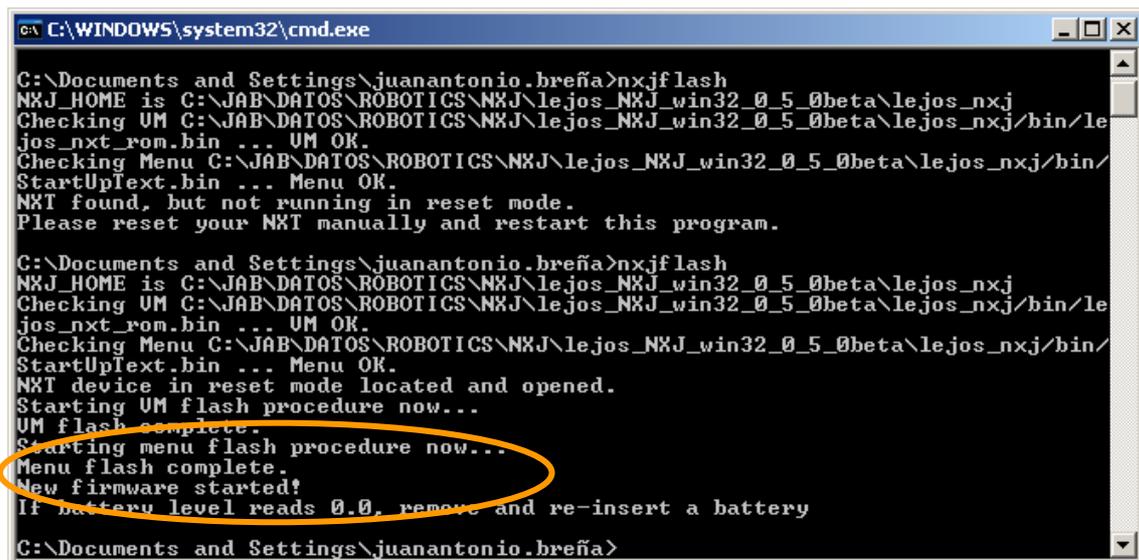
C:\Documents and Settings\juanantonio.breña>nxjflash
NXJ_HOME is C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj
Checking UM C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj/bin/le
jos_nxt_rom.bin ... UM OK.
Checking Menu C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj/bin/
StartUpText.bin ... Menu OK.
NXT found, but not running in reset mode.
Please reset your NXT manually and restart this program.

C:\Documents and Settings\juanantonio.breña>

```

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:



```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\juanantonio.breña>nxjflash
NXJ_HOME is C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj
Checking UM C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj/bin/le
jos_nxt_rom.bin ... UM OK.
Checking Menu C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj/bin/
StartUpText.bin ... Menu OK.
NXT found, but not running in reset mode.
Please reset your NXT manually and restart this program.

C:\Documents and Settings\juanantonio.breña>nxjflash
NXJ_HOME is C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj
Checking UM C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj/bin/le
jos_nxt_rom.bin ... UM OK.
Checking Menu C:\JAB\DATOS\ROBOTICS\NXJ\lejos_NXJ_win32_0_5_0beta\lejos_nxj/bin/
StartUpText.bin ... Menu OK.
NXT device in reset mode located and opened.
Starting UM flash procedure now...
UM flash complete.
Starting menu flash procedure now...
Menu flash complete.
New firmware started!
If battery level reads 0.0, remove and re-insert a battery

C:\Documents and Settings\juanantonio.breña>

```

If you reach this step then your brick has a leJOS firmware inside

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

2.5.- Eclipse IDE and Eclipse plugin for leJOS

2.5.1.- Introduction

Develop any Java program in general is easy, but it is more comfortable if you use a development tool. In the java market, many developers in the world use Eclipse IDE or Netbeans. In this section I will explain how to develop with Eclipse and the plugin for eclipse.

2.5.2.- Installing Eclipse

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/>

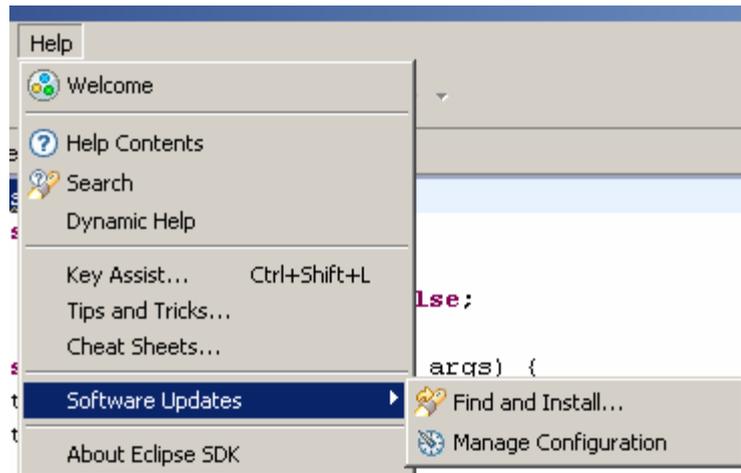
The screenshot shows the Eclipse website's 'Downloads' page. The main content area is titled 'Eclipse Downloads' and includes a warning: 'To download Eclipse, select a package below or choose one of the third party Eclipse distros. You will need a Java runtime environment (JRE) to use Eclipse (Java 5 JRE recommended). All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.' Below this, there is a section for 'Eclipse Europa Fall Maintenance Packages - Windows' with a link to 'compare packages'. A table lists several download packages:

Package Name	Size	Operating Systems
Eclipse IDE for Java Developers	78 MB	Windows, Linux, Mac OS X
Eclipse IDE for Java EE Developers	126 MB	Windows, Linux, Mac OS X
Eclipse IDE for C/C++ Developers	63 MB	Windows, Linux, Mac OS X
Eclipse for RCP/Plug-in Developers	153 MB	Windows, Linux, Mac OS X
Eclipse Classic 3.3.1.1	140 MB	Windows, Linux, Mac OS X

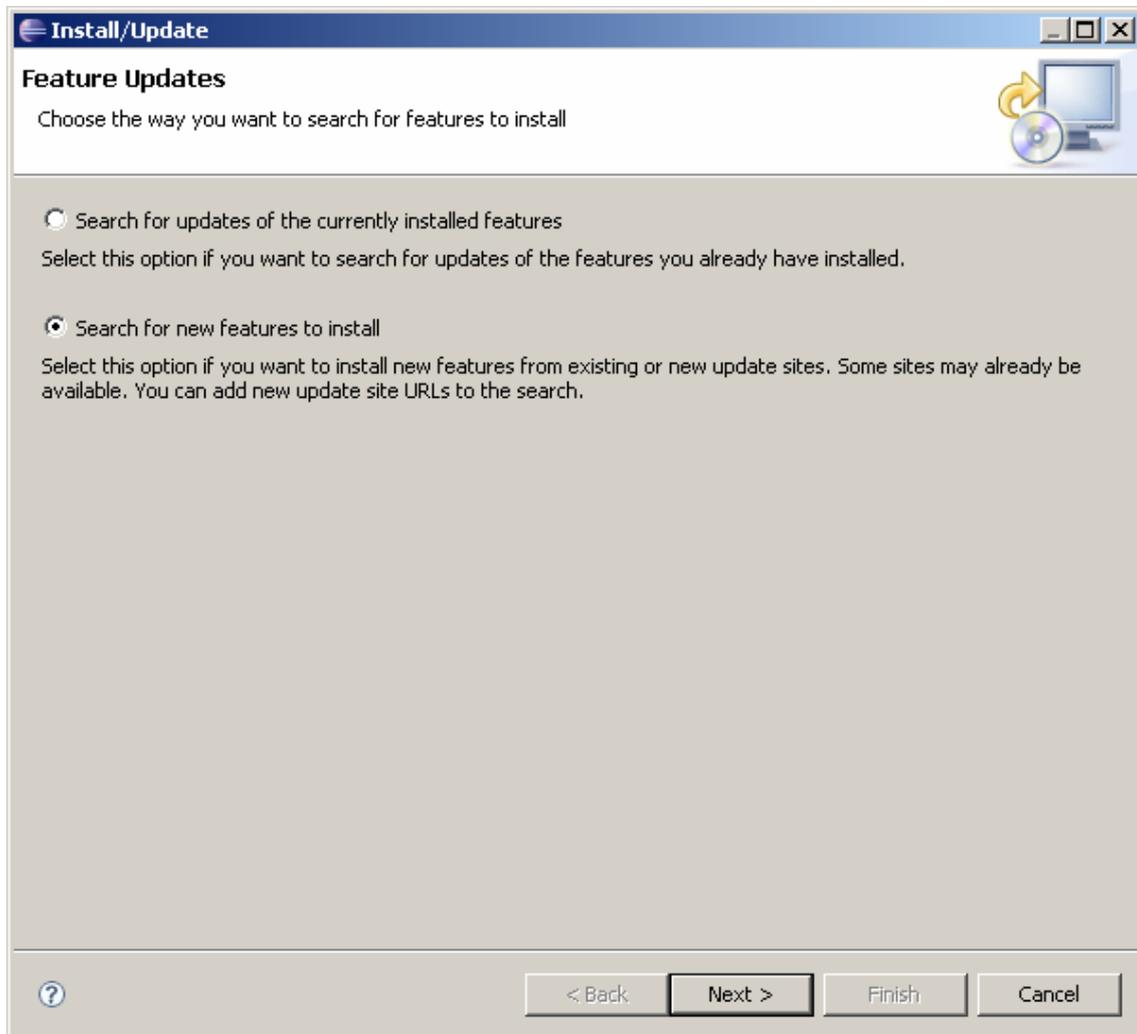
The right sidebar contains a 'Browse downloads' section with links for Bit Torrents, By Project, By Topic, Source code, and Ganymede M4 Packages. It also features a 'Popular projects' list including Web Tools, PHP Development (PDT), Modeling Tools (MDT), C/C++ Development (CDT), Visual Editor (VE), Business Intelligence and Reporting (BIRT), Modeling Framework (EMF), Mylyn, Test & Performance (TPTP), and AspectJ.

2.5.3.- Installing Eclipse plugin for leJOS

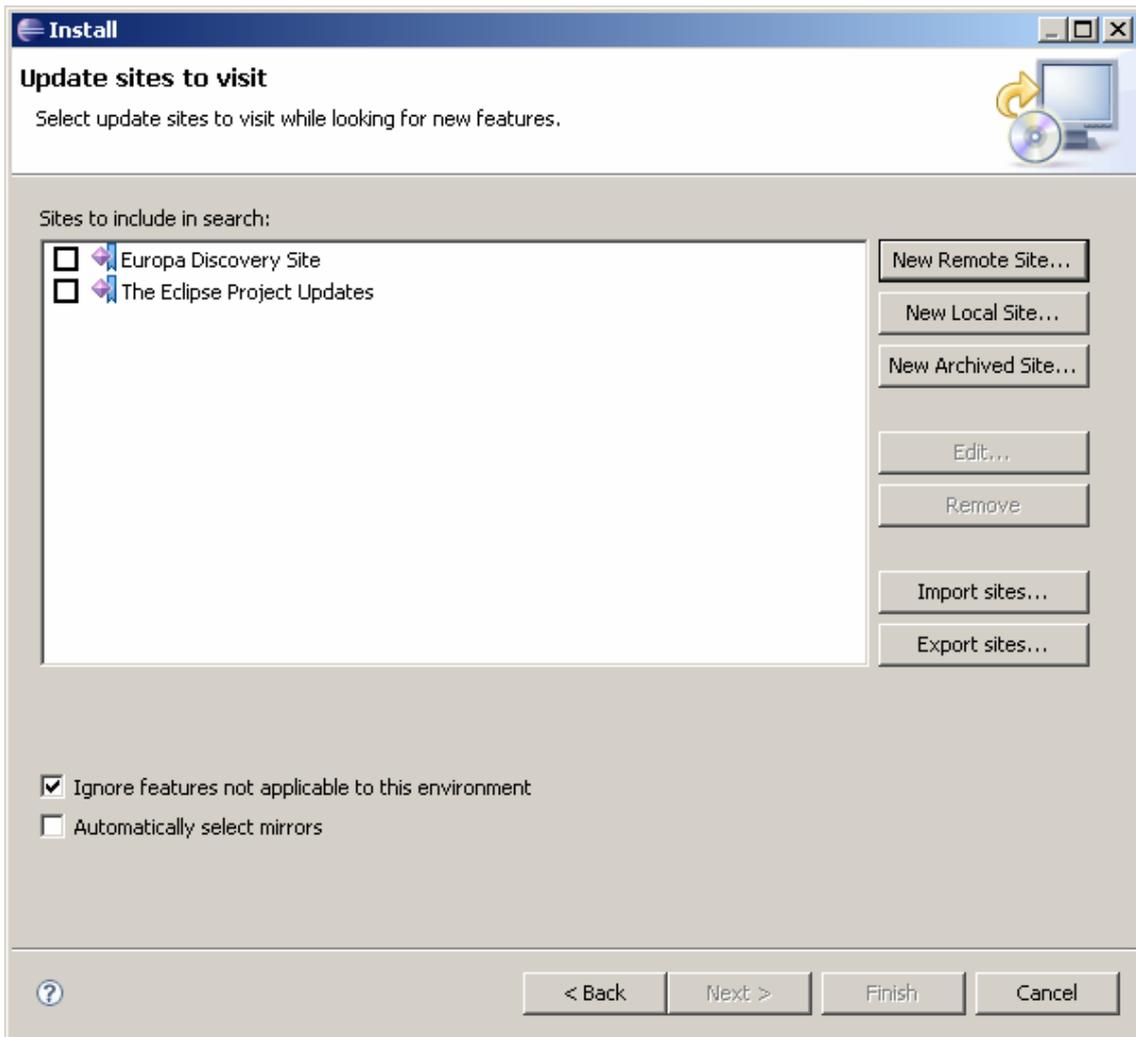
When you execute Eclipse and you want to install any Eclipse plug-in, in this case NXJ Plug-in, you have to go to *help > software updates > find and install*:



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



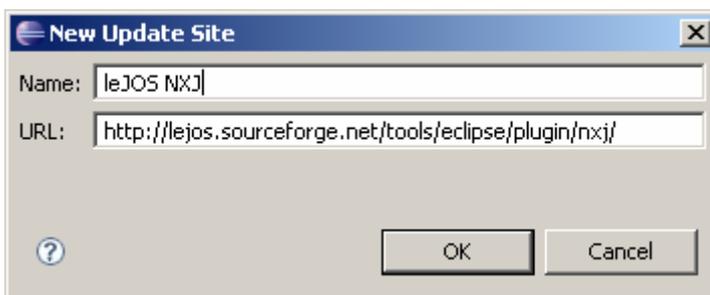
Click in the button next to indicate where is NXT Plug-in. Click in New remote Site:



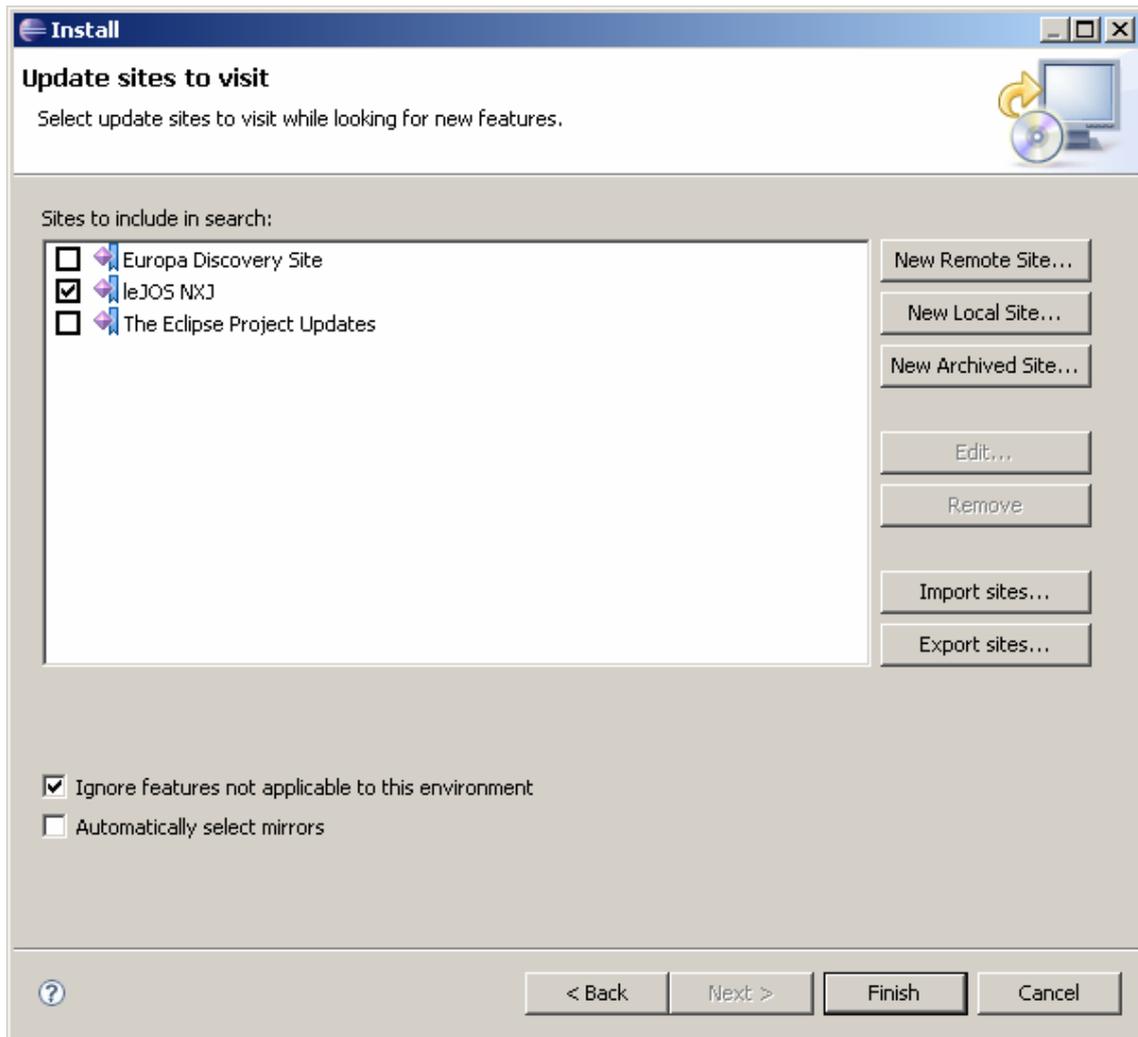
The parameters to write in the next window are:

Name: leJOS NXJ

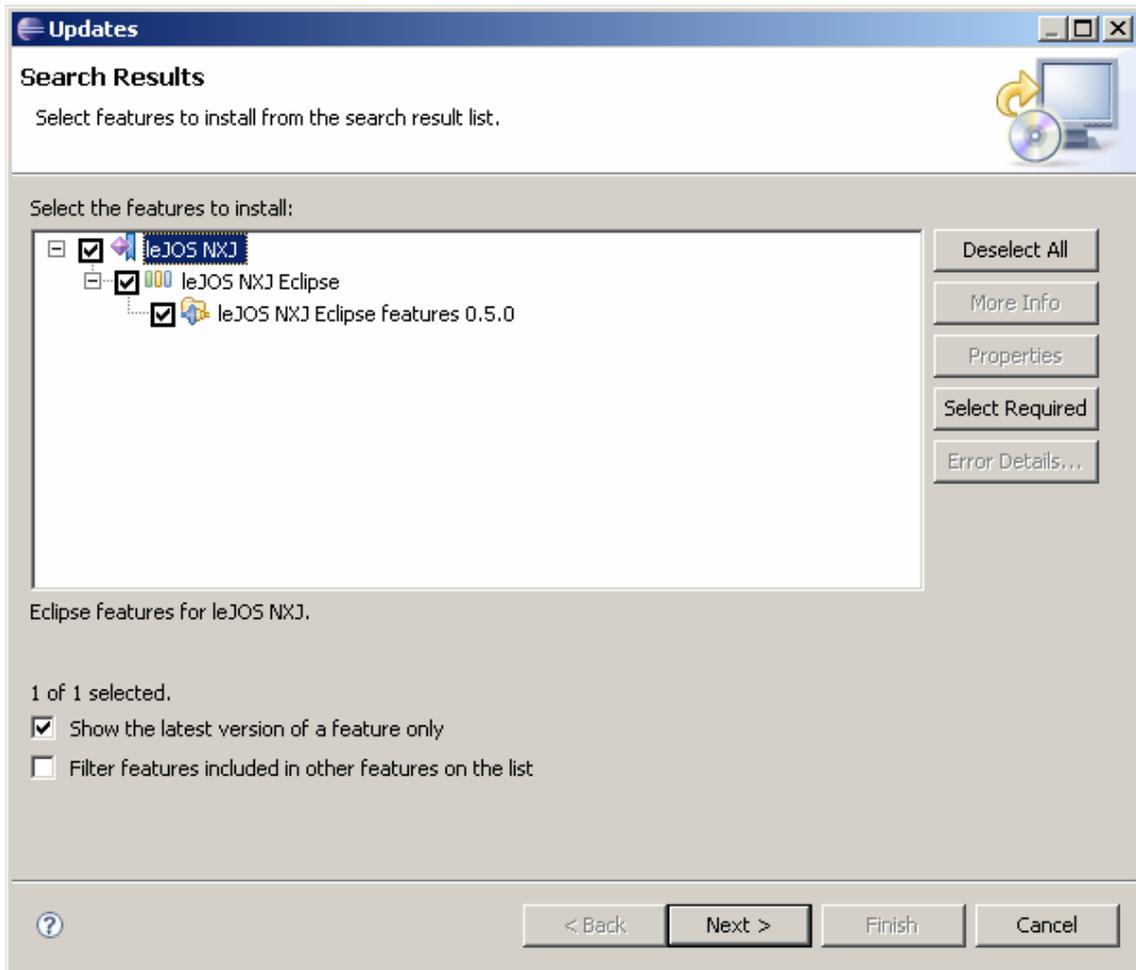
URL: <http://lejos.sourceforge.net/tools/eclipse/plugin/nxj/>

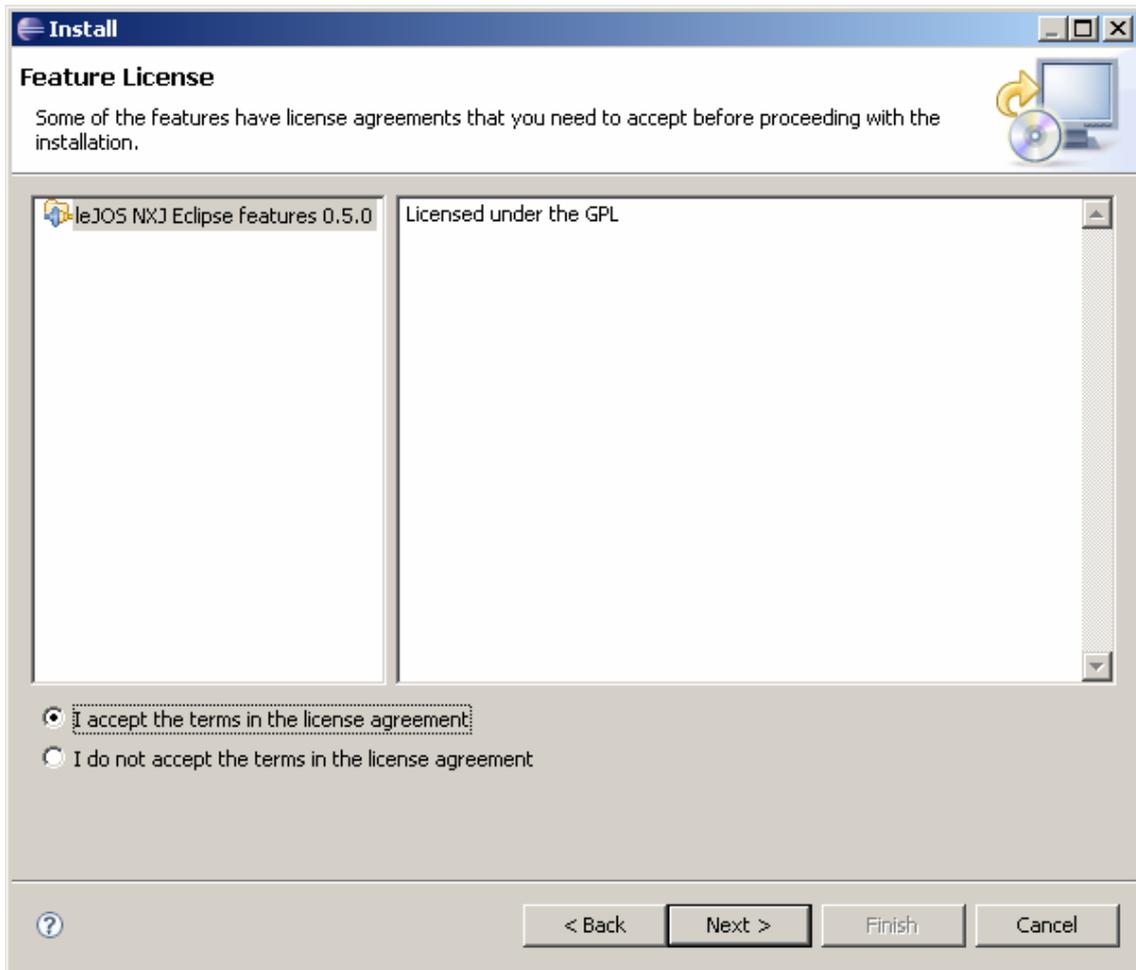


Then, select the site leJOS NXJ:

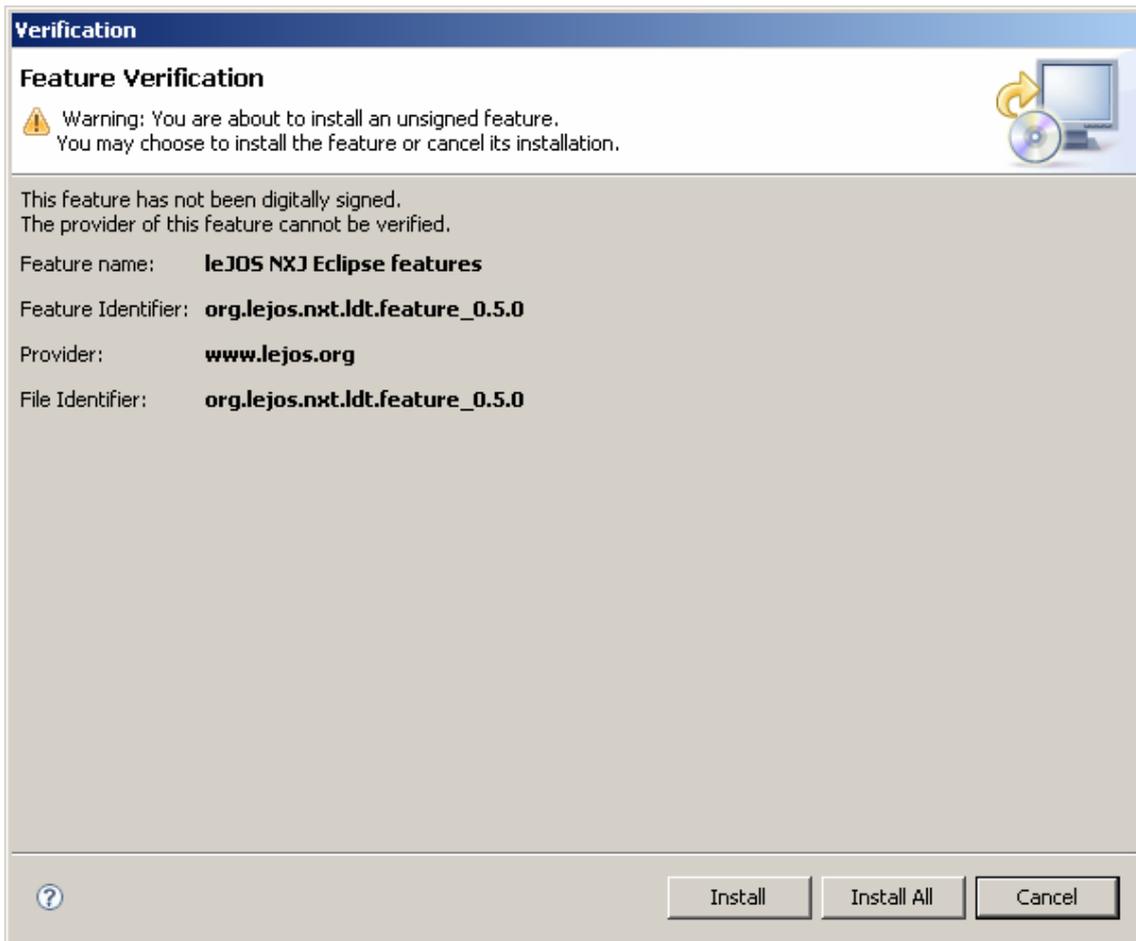


And click in Finish button. In next window, Check all options and click in next button.





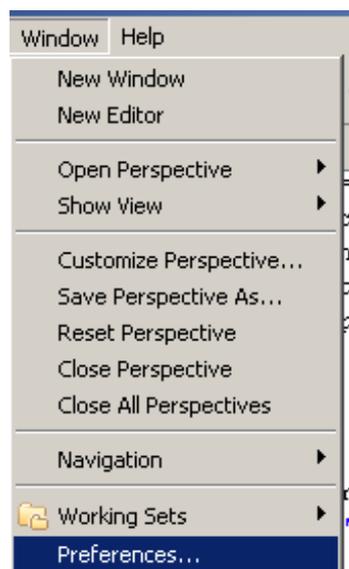
Then you have to accept the terms in the licence agreement and click in next button to install the NXJ plug-in.



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

2.5.4.- Configuring Eclipse plugin for leJOS

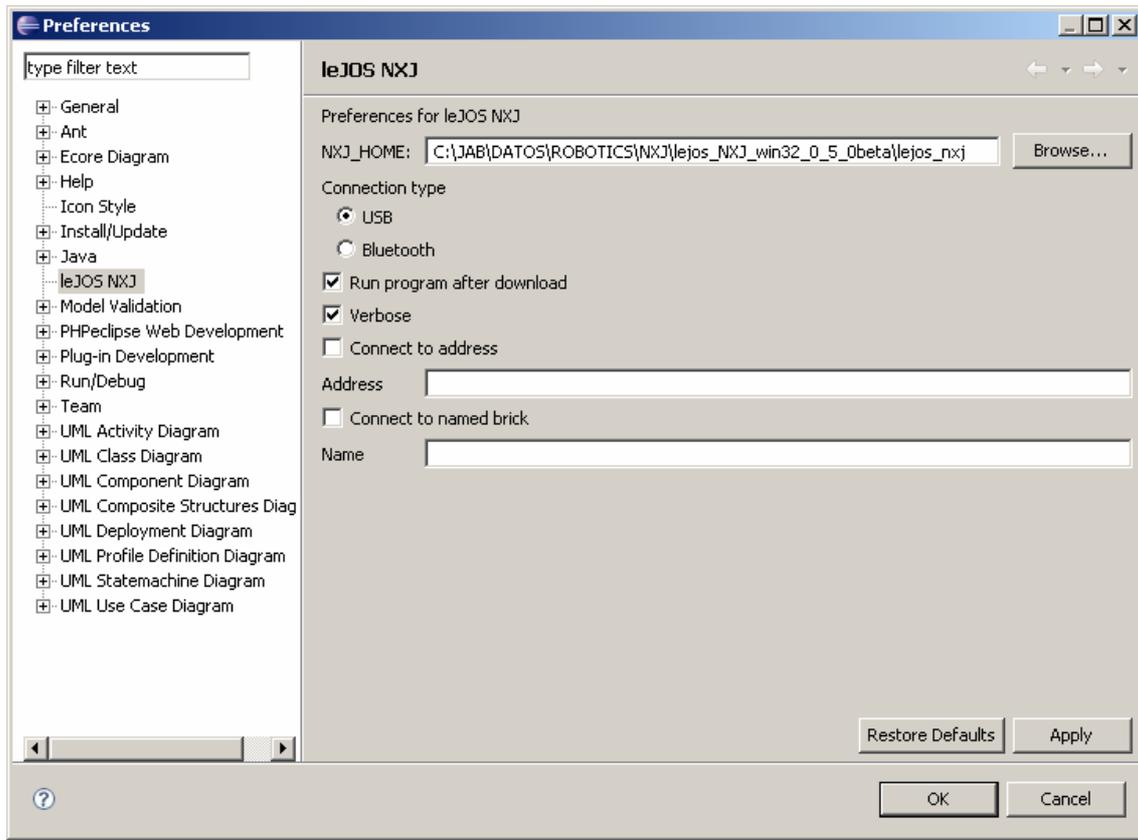
Once you have installed NXJ Plug-in, you have to set NXJ_HOME variable in preference area in Eclipse.



NXJ_HOME

=

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

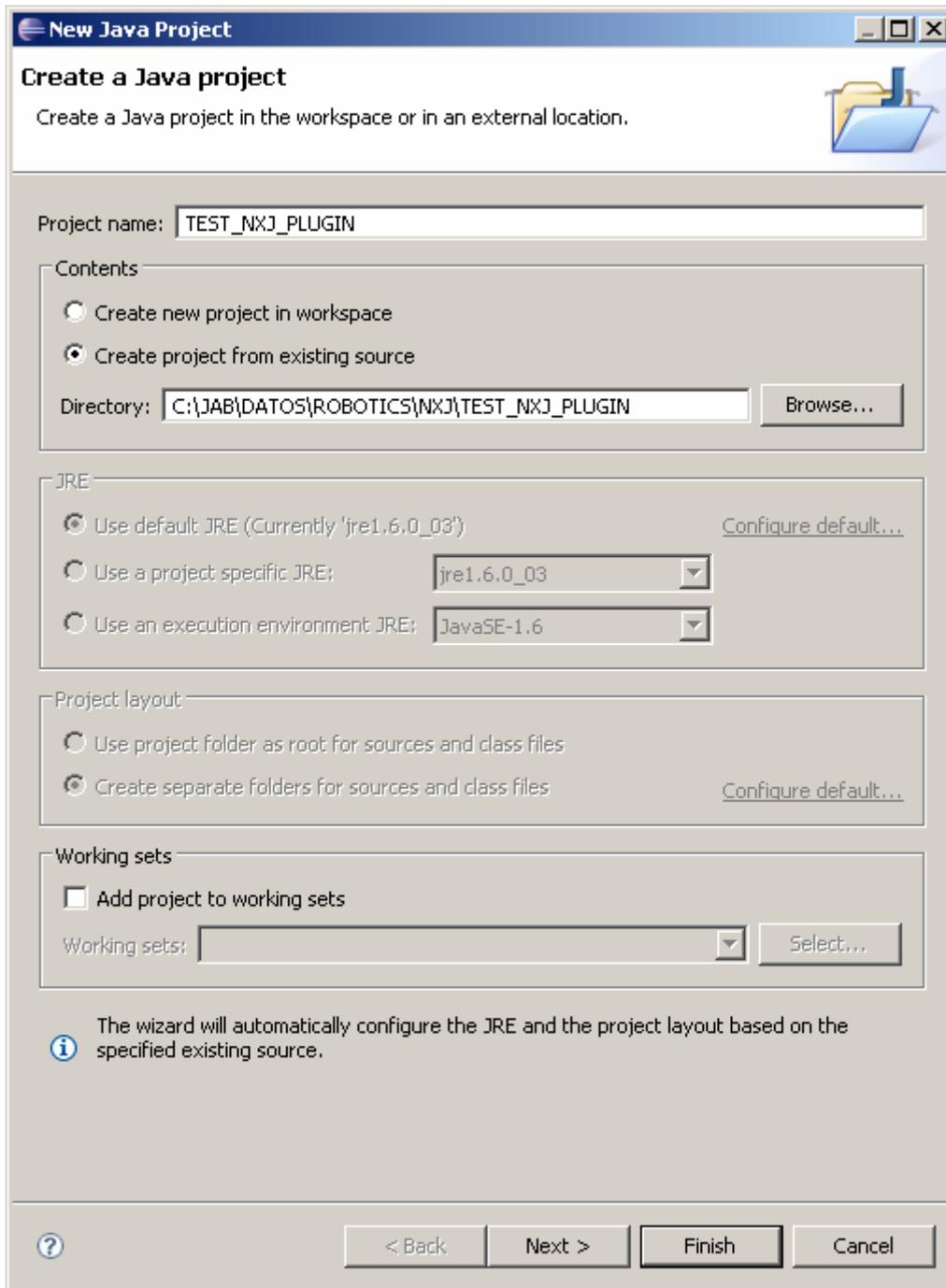


2.5.5.- Creating a new project in Eclipse IDE

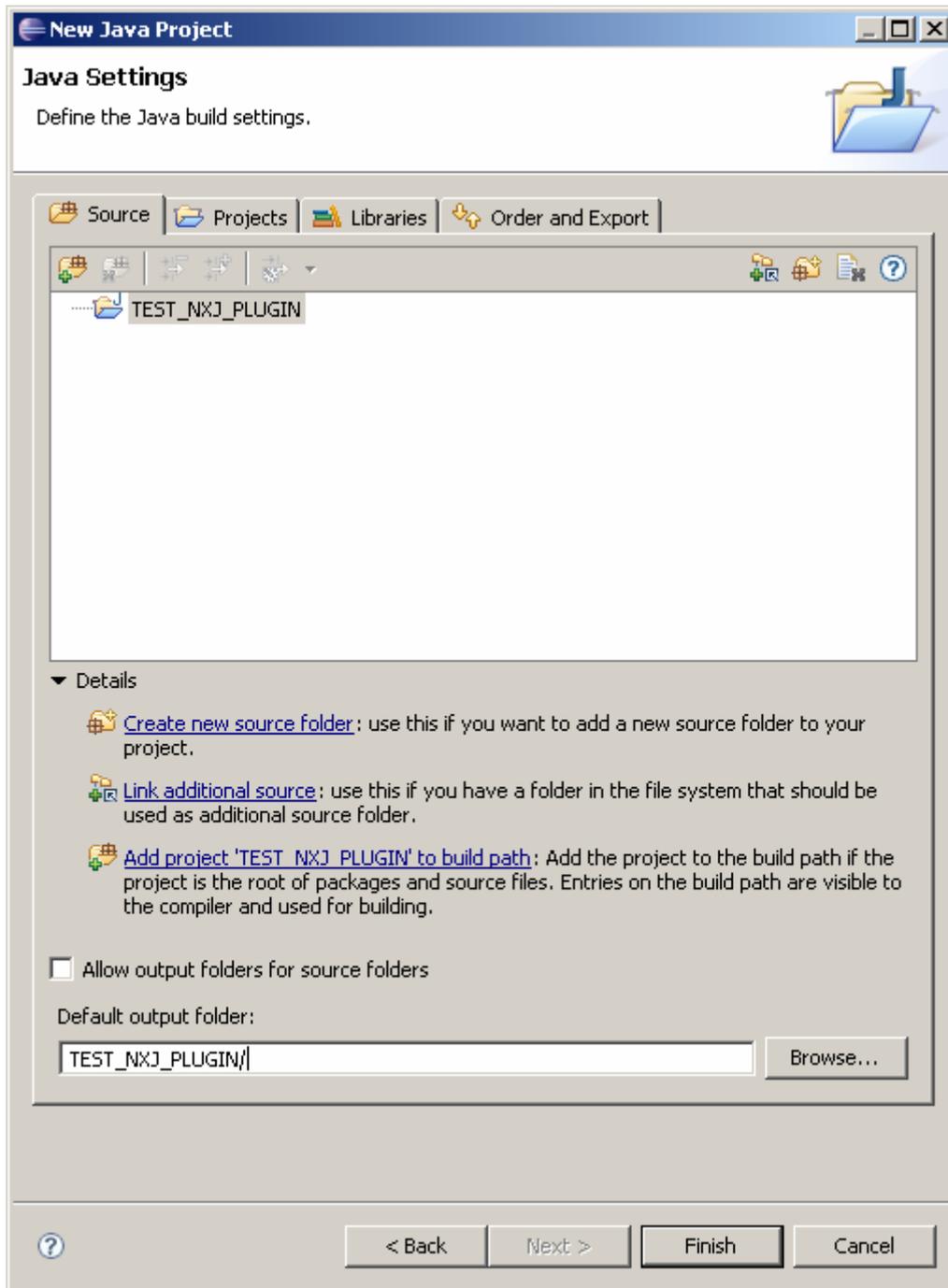
In Eclipse, you can create a Java Project. To create a Java project, go to *File > New > Java Project*:



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"



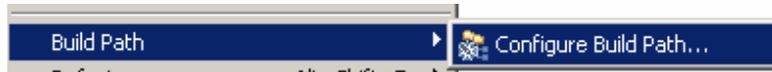
Then your project will be associated with lib path in your NXJ release.



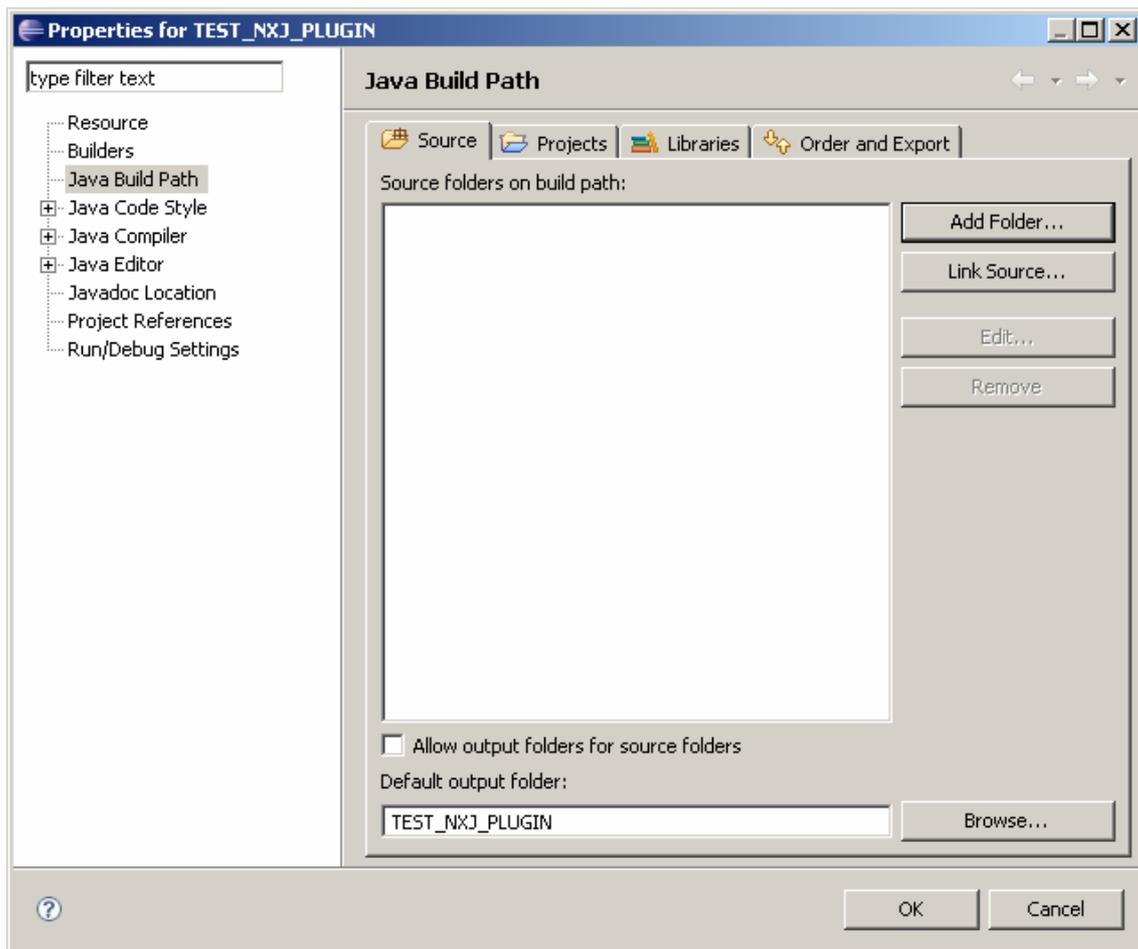
Besides, in Eclipse console window, you will see the following message:

Project TESTING_NXJ_PLUGIN now is a leJOS NXJ project

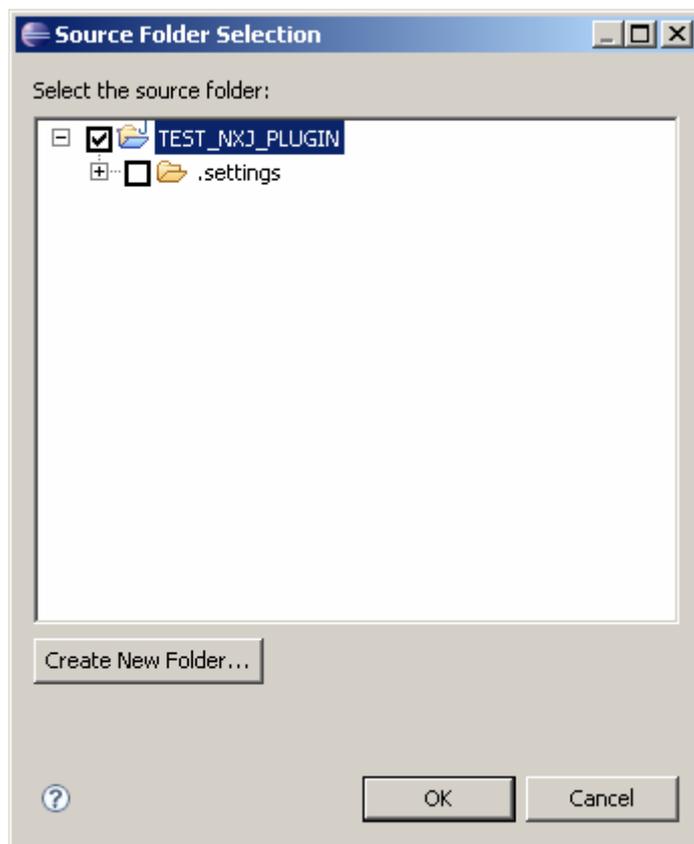
Then, it is necessary to define Build path. Select your NXJ Project and click in Configure Build Path in context menu:



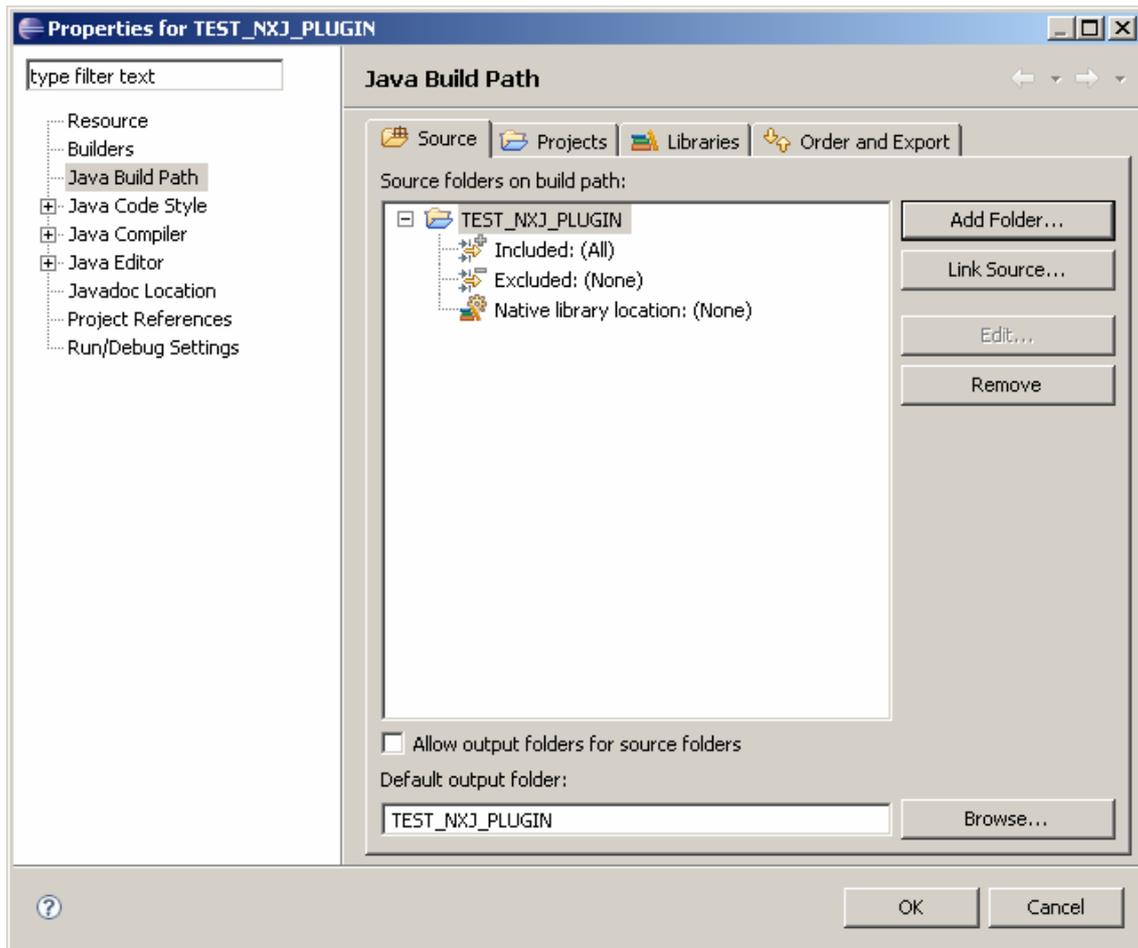
Add new folder:



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.



2.5.5.1.- NXJ Plug-in Features

2.5.5.1.1.- Introduction

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

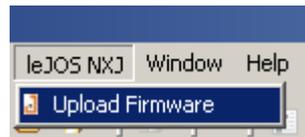
1. Upload latest firmware into NXT Brick
2. Upload NXJ programs into your NXT Brick
3. Convert any Java project into NXJ project

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

Besides, NXJ Plugin includes a excellent documentation.

2.5.5.1.2.- 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*:



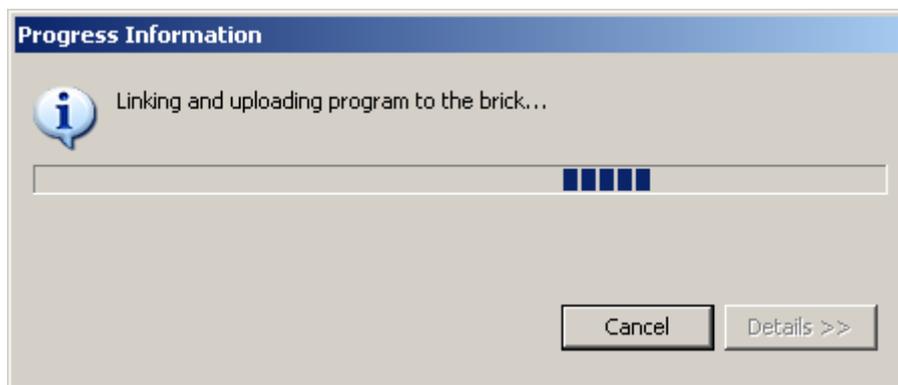
2.5.5.1.3.- Upload Program to the NXT brick

Normally you send programs to NXT brick using DOS console or using Eclipse manually or another 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*:



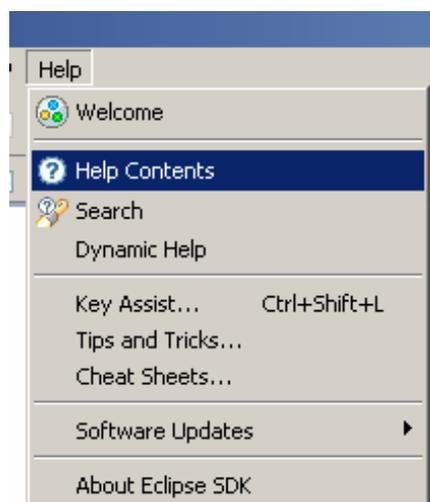
Then if your class doesn't have any syntactic error, connect your NXT brick and click in the option, then you will see the following window:



Then your program will be running in your brick!

2.5.5.1.4.- NXJ Plug-in documentation

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

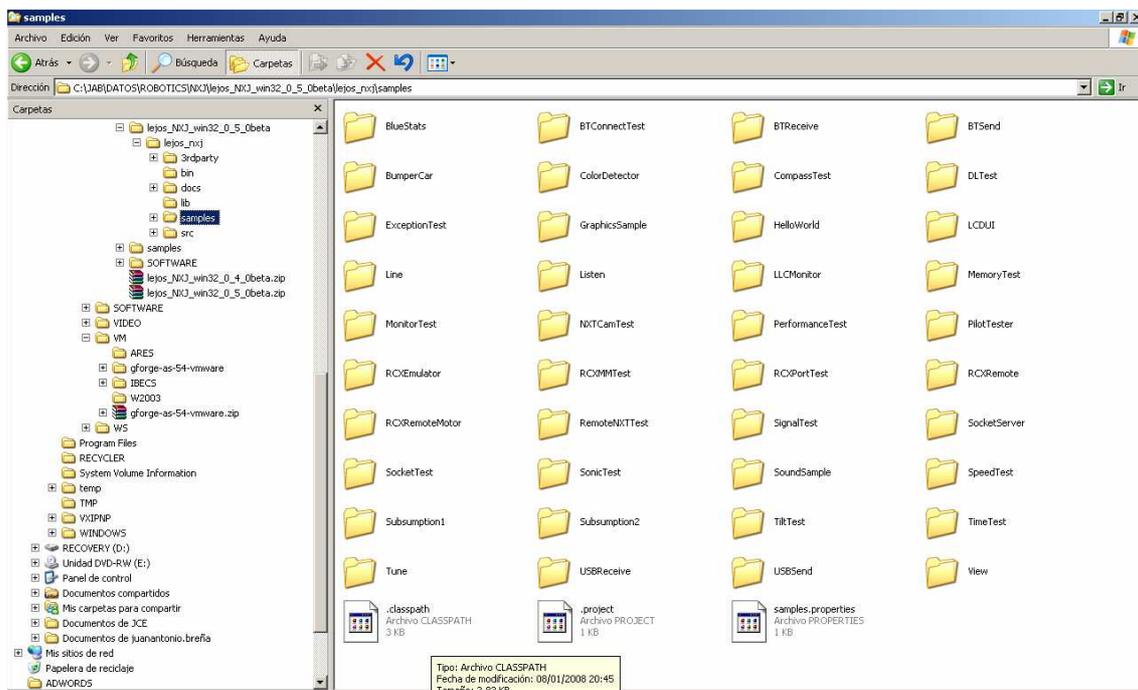


Then you will see a new section named leJOS NXJ.



2.6.- 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

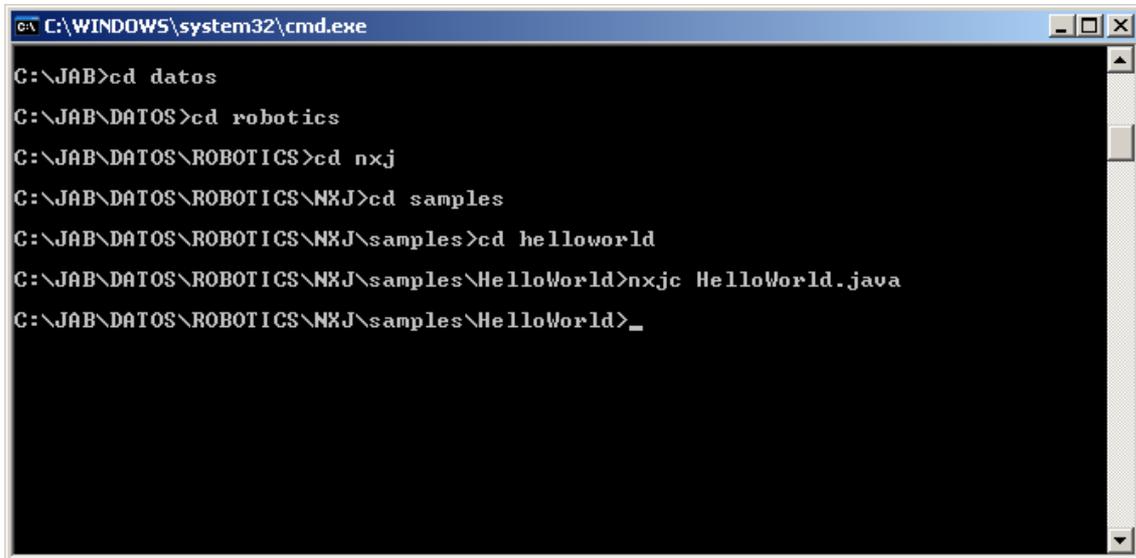
```
import lejos.nxt.*;

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`

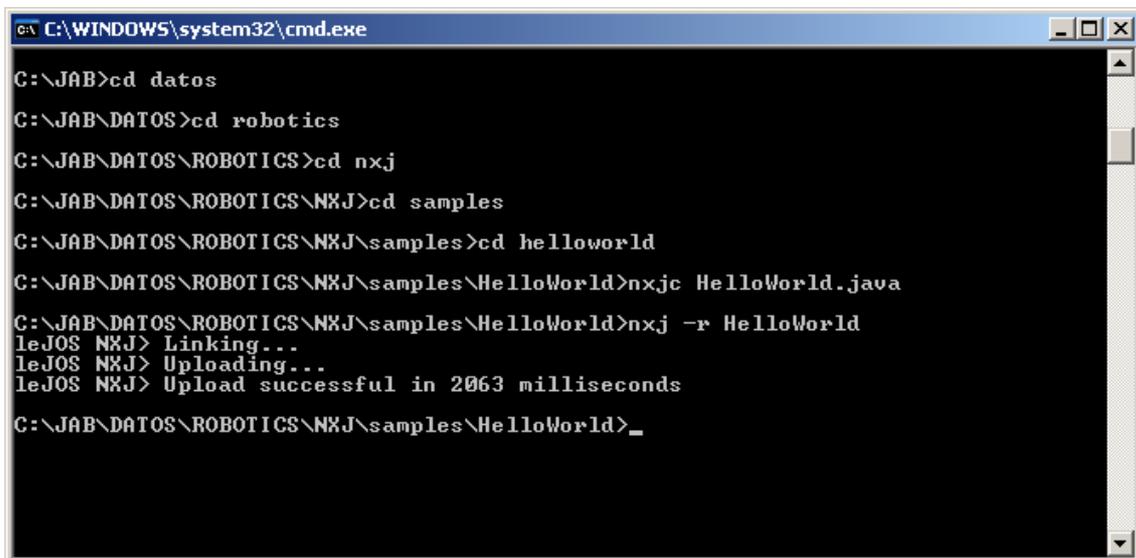


```

C:\WINDOWS\system32\cmd.exe
C:\JAB>cd datos
C:\JAB\DATOS>cd robotics
C:\JAB\DATOS\ROBOTICS>cd nxj
C:\JAB\DATOS\ROBOTICS\NXJ>cd samples
C:\JAB\DATOS\ROBOTICS\NXJ\samples>cd helloworld
C:\JAB\DATOS\ROBOTICS\NXJ\samples\HelloWorld>nxjc HelloWorld.java
C:\JAB\DATOS\ROBOTICS\NXJ\samples\HelloWorld>_
  
```

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>`



```

C:\WINDOWS\system32\cmd.exe
C:\JAB>cd datos
C:\JAB\DATOS>cd robotics
C:\JAB\DATOS\ROBOTICS>cd nxj
C:\JAB\DATOS\ROBOTICS\NXJ>cd samples
C:\JAB\DATOS\ROBOTICS\NXJ\samples>cd helloworld
C:\JAB\DATOS\ROBOTICS\NXJ\samples\HelloWorld>nxjc HelloWorld.java
C:\JAB\DATOS\ROBOTICS\NXJ\samples\HelloWorld>nxj -r HelloWorld
leJOS NXJ> Linking...
leJOS NXJ> Uploading...
leJOS NXJ> Upload successful in 2063 milliseconds
C:\JAB\DATOS\ROBOTICS\NXJ\samples\HelloWorld>_
  
```

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

2.7.- Summary

In this chapter you should learn:

1. How to install the prerequisites to install leJOS software
2. How to install leJOS software

3. How to install leJOS firmware into your NXT brick
4. How to install Eclipse IDE
5. How to install Eclipse plugin for leJOS
6. How to compile and run into the NXT brick your first program:
HelloWorld.java

3.- Basic concepts about Java

3.1.- Introduction

LeJOS project uses the Java as the language to develop software for robots. Java is an Object Oriented programming language. This ebook is not a ebook to learn Java, but I think that it is important to know some features about Java.

3.2.- Learning the example HelloWorld.java

Any Java program has a main program which manages all operations. In Java all files are classes, but only one Java class has the method main. If you notice the first program, HelloWorld.java, then you will see the public method main

```
import lejos.nxt.*;

public class HelloWorld{
    private static final String welcomeMessage = "Hello World";
    public static void main (String[] args){
        //System.out.println("Hello World");
        LCD.drawString(welcomeMessage,0,0);
        LCD.refresh();
    }
}
```

If the project is complex you will have multiple classes and one class which manipulate the rest of classes. In this case HelloWorld.java is a simple project with a unique class.

3.3.- Discovering the sections in any Java class

Any Java class has the following areas in the code:

1. The Import Area
2. Class encapsulation
3. The main method

3.3.1.- The Import Area:

The import area is the part of any Java Class used to declare what Java package you are going to use in this class. In this example we use NXT LCD features, then it is necessary to import the package `lejos.nxt.*`; if you want to import the unique class which you use in your program, use this way: `import lejos.nxt.LCD`;

3.3.2.- Class encapsulation:

Any Java Class has the following areas: Properties, Methods and Events. In this easy example the Class HelloWorld:

1. Properties:
 1. wellcomeMessage
2. Methods:
 1. main

3.3.3.- The main method:

The main class needs a main method to execute it. In this example the class HelloWorld has a main class used to execute some instructions:

```
public static void main (String[] args){  
    //System.out.println("Hello World");  
    LCD.drawString(welcomeMessage,0,0);  
    LCD.refresh();  
}
```

3.4.- Summary

In this chapter you learnt some concepts about Java technology

4.- Sensors

4.1.- Introduction

LeJOS is able to manage several sensors. The unique limitations is the number of input ports which is reduced to 4 input ports.

In this section we explain how to use the following sensors:

- Ultrasonic sensor
- Compass sensor
- GPS
- NXTCam
- NXTLine
- Touch sensor
- Sound sensor



4.2.- Ultrasonic Sensor

The Ultrasonic Sensor helps your NXT robot measure distances and "see" where objects are.

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 with curved shapes 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 an instance of UltrasonicSensor:

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

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

If your program needs 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/nxt/nxj/api/lejos/nxt/UltrasonicSensor.html>

4.3.- Compass sensor

The NXT Compass Sensor contains a digital magnetic compass that measures the earth's magnetic field and calculates a heading angle. The Compass Sensor

connects to an NXT sensor port using a standard NXT wire and uses the digital I2C communications protocol. The current heading is calculated to the nearest 1° and refreshed 100 times per second.

NXT has the possibility to use a digital compass sensor to specifically to aid navigation

A compass can be a valuable sensor that your robot can use to navigate the world. The Dinsmore 1490 compass is a cheap and durable unit that is relatively easy to interface to. It provides 8 headings (N, NE, E, SE, S, SW, W, and NW). 4 signals, N,S,E,W are read by the Basic Stamp II controller to determine the robot's heading.



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

/**
 * Simple test of compass sensors.
 *
 * Works with Mindsensors and HiTechnic compass sensors.
 *
 * @author Lawrie Griffiths
 */
public class CompassTest {

    public static void main(String[] args) throws Exception {
        CompassSensor compass = new CompassSensor(SensorPort.S1);
        while(!Button.ESCAPE.isPressed()) {
            LCD.clear();
            LCD.drawInt((int) compass.getDegrees(), 0, 0);
            LCD.refresh();
            Thread.sleep(500);
        }
    }
}
```

Compass sensor is sensor which is not included into Lego Mindstorms NXT so you have to import the class from the package `lejos.nxt.addon.*`.

The way to get the degrees from the sensor is `compass.getDegrees()`

This kind of servo is necessary to many projects which use navigation. I recommend to purchase a unit from Mindsensors or Hitechnic.

Further information:

<http://lejos.sourceforge.net/nxt/nxj/api/lejos/nxt/addon/CompassSensor.html>

4.4.- GPS

With leJOS is very easy to manage information from a GPS device. In LeJOS project exists 2 ways to manage GPS data:

1. Using the package `javax.microedition.location`
2. Using the package `lejos.gps`

4.4.1.- Using the package `javax.microedition.location`

4.4.2.- Using the package `lejos.gps`

The package `lejos.gps` allows the developer to manage gps information from the following NMEA Sentences:

- GGA - essential fix data which provide 3D location and accuracy data.
- GSA - GPS DOP and active satellites
- GPV - GPS Satellites in view
- RMC - NMEA has its own version of essential gps pvt (position, velocity, time) data. It is called RMC
- VTG - Track Made Good and Ground Speed.

With data from this NMEA sentences it is possible to return Date objects to manage date information or Coordinates objects to calculate distances.

A GPS is perfect if you have a robot which operates in a outdoor area. GPS devices are sensitives to the place where it operates. You can measure the quality of the data from any GPS device if you see the number of satellites and error data which you can get from the following methods:

- `getModeValue()`
- `getPDOP()`
- `getHDOP()`
- `getVDOP()`

In the following example you will show how to use a GPS with leJOS:

```
import java.io.*;
import java.util.*

import javax.bluetooth.RemoteDevice;
import javax.microedition.location.Coordinates;

import lejos.gps.GPS;
import lejos.nxt.comm.*;
import lejos.nxt.*;
import lejos.util.Stopwatch;

/**
 *
 * @author Juan Antonio Brenha Moral
 *
 */
public class GPSThread extends Thread{
    //GPS Receiver
    private String GPSPattern = "";
    static private byte[] pin = {};

    //Inquire code
```

```
private static byte[] cod = {0,0,0,0}; // 0,0,0,0 picks up every
Bluetooth device regardless of Class of Device (cod).
```

```
//Bluetooth Connection messages
private String MESSAGE_SEARCHING = "Searching";
private String MESSAGE_CONNECTING = "Connecting";
private String MESSAGE_CONNECTED = "Connected";
private String MESSAGE_RECONNECTING = "Reconnecting";
private String MESSAGE_CONNECTION_FAILED = "Connection failed";
private String MESSAGE_NO_GPS = "No detected GPS";
```

```
private String MESSAGE_CLOSING = "Closing...";
```

```
//Bluetooth
static private RemoteDevice GPSDevice = null;
static private BTConnection btGPS = null;
static private InputStream in = null;
static protected GPS gps = null;
```

```
//Detect GPS Device
private boolean GPSDetected = false;
//Connect with GPS Device
private int connectionStatus = 0;
```

```
//GPS Data
private Date init;
private Date now;
protected Coordinates origin;
protected Coordinates current;
```

```
//Command to process
private int CMD = 0;
```

```
//Connection counter
private int connectionCounter = 0;
```

```
//Exchange Data Object
private LRDataBridge db;
```

```
//LCD
//private int LCDRow = 6;
```

```
/* CONSTRUCTOR */
```

```
public GPSThread(String gpcsp, byte[] p){
    GPSPattern = gpcsp;
    pin = p;
}
```

```
public GPSThread(String gpcsp, byte[] p, LRDataBridge dObj){
    GPSPattern = gpcsp;
    pin = p;

    db = dObj;
}
```

```
public GPSThread(LRDataBridge dObj){
    db = dObj;
}
```

```
/* Setters & Getters */
```

```

public void setGPSDevice(String gps){
    this.GPSPattern = gps;
}

public String getGPSDevice(){
    return GPSPattern;
}

public void setGPSPin(byte[] p){
    pin = p;
}

public byte[] getGPSPin(){
    return pin;
}

public void setCommand(int c){
    CMD = c;
}

public void run(){
    while(true){
        if(CMD == 1){
            fetchData();
        }else if(CMD == 2){
            readGPS();
        }
    }
}

/**
 * Methods used to discover a predefined GPS receiver and you
need
 * to connect with it directly.
 *
 * @param BTPatternName
 * @return
 */
private boolean discoverBTDevice(String BTPatternName){
    boolean GPSDetected = false;
    RemoteDevice btrd = null;
    String BTDeviceName;

    //LCD.drawString(MESSAGE_SEARCHING, 0, LCDRow);
    //LCD.refresh();
    db.setBTGPSMSG(MESSAGE_SEARCHING);
    Vector devList = Bluetooth.getKnownDevicesList();

    if(devList.size() > 0){
        for (int i = 0; i < devList.size(); i++) {
            btrd = ((RemoteDevice) devList.elementAt(i));

            BTDeviceName = btrd.getFriendlyName(true);
            if(BTDeviceName.indexOf(BTPatternName) != -1){
                GPSDevice = btrd;
                GPSDetected = true;
                break;
            }
        }
    }
}

```

```

    }
}

return GPSTDetected;
}

static boolean discoverBTDevices(){
    boolean GPSTDetected = false;

    LCD.clear();
    LCD.drawString("Searching...", 0, 0);
    LCD.refresh();
    //Make an BT inquire to get all Devices with BT Services
enable
    Vector devList = Bluetooth.inquire(5, 10,cod);

    //If exist GPS Devices near
    if (devList.size() > 0){
        String[] names = new String[devList.size()];
        for (int i = 0; i < devList.size(); i++) {
            RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
            names[i] = btrd.getFriendlyName(true);
        }

        TextMenu searchMenu = new TextMenu(names,1);
        String[] subItems = {"Connect"};
        TextMenu subMenu = new TextMenu(subItems,4);

        int selected;
        do {
            LCD.clear();
            LCD.drawString("Found",6,0);
            LCD.refresh();
            //Menu 1: Show all BT Devices
            selected = searchMenu.select();
            if (selected >=0){
                RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(selected));
                LCD.clear();
                LCD.drawString("Found",6,0);
                LCD.drawString(names[selected],0,1);

                LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
                //Menu 2: Show GPS Device
                int subSelection = subMenu.select();
                if (subSelection == 0){
                    GPSTDetected = true;
                    GPSDevice = btrd;
                    break;
                }
            }
        } while (selected >= 0);
    }else{
        GPSTDetected = false;
    }

    return GPSTDetected;
}

/**

```

```

    * This method connect with a RemoteDevice.
    * If the connection has success then the method create an
instance of
    * the class GPS which manages an InputStream
    *
    * @return
    */
static int connectGPS(){
    int result;
    //Bluetooth.addDevice(GPSDevice);

    //BTConnection btGPS = null;
    btGPS = Bluetooth.connect(GPSDevice.getDeviceAddr(),
NXTConnection.RAW,pin);

    if(btGPS == null){
        result = -1;//No connection
    }else{
        result = 1;//Connection Sucessful
    }

    try{
        in = btGPS.openInputStream();
        gps = new GPS(in);
        gps.updateValues(true);//Update values always

        result = 2;//
    }catch(Exception e){
        result = -2;
    }

    return result;
}

private void readGPS(){
    //Store the initial coordinate and the moment
    boolean firstMomentFlag = false;

    while(true){
        //1. Detect
        GPSPattern = db.getGPSDevice();
        if(GPSPattern != ""){
            GPSDetected = discoverBTDevice(GPSPattern);
        }else{
            GPSDetected = discoverBTDevices();
        }

        if(GPSDetected){
            //2. Connecting
            //LCD.drawString(MESSAGE_CONNECTING, 0,
LCDRow);

            //LCD.refresh();
            db.setBTGPSMSG(MESSAGE_CONNECTING);
            connectionStatus = connectGPS();

            if(connectionStatus == 2){
                db.setGPSEnabled(true);
                //LCD.drawString(MESSAGE_CONNECTED, 0,
LCDRow);

                //LCD.refresh();
                db.setBTGPSMSG(MESSAGE_CONNECTED);

```

```

        Sound.beep();

        connectionCounter++;

        //While exist a BT Connection (GPS Object
process InputStream)
        while(!gps.existInternalError()){

            //This block store origin GPS Data
            and Time
            if(!firstMomentFlag){
                //A way to isolate date
                Date tempDate =
                int hours =
                int minutes =
                int seconds =
                int year =
                int month =

                int day = tempDate.getDay();
                init = new Date();
                init.setHours(hours);
                init.setMinutes(minutes);
                init.setSeconds(seconds);
                init.setYear(year - 2000);
                init.setMonth(month);
                init.setDay(day);

                origin = new
Coordinates(gps.getLatitude(),gps.getLongitude(),gps.getAltitude());

                //Repeat the operation until
you have valid data:
                if(
                    (init.getSeconds() !=
                    (init.getYear() !=
                    (gps.getLatitude() !=
                    (gps.getGPSStatus())
                    ){
                        db.setOrigin(origin);
                        db.setInit(init);

                        firstMomentFlag = true;

                db.setInitDateFixed(true);
            }
        }
        current = new
Coordinates(gps.getLatitude(),gps.getLongitude(),gps.getAltitude());
        db.setCurrent(current);
        db.setNow(gps.getDate());

```

```

        db.setSatellitesTracked(gps.getSatellitesTracked());
        db.setGPSSpeed(gps.getSpeed());
    }
    //LCD.drawString(MESSAGE_RECONNECTING, 0,
LCDRow);

    //LCD.refresh();
    db.setBTGPSMSG(MESSAGE_RECONNECTING);
    db.setGPSEnabled(false);
}
}

//LCD.drawString(MESSAGE_CONNECTION_FAILED, 0, LCDRow);
//LCD.refresh();

db.setBTGPSMSG(MESSAGE_CONNECTION_FAILED);
db.setGPSEnabled(false);
}
}
}
}

//CMD1: FetchData: Connect, GetGPSData, Disconnect
private void fetchData(){
    //1. Detect
    GPSPattern = db.getGPSDevice();
    if(GPSPattern != ""){
        GPSDetected = discoverBTDevice(GPSPattern);
    }else{
        GPSDetected = discoverBTDevices();
    }

    if(GPSDetected){
        //2. Connecting
        //LCD.drawString(MESSAGE_CONNECTING, 0, LCDRow);
        //LCD.refresh();
        db.setBTGPSMSG(MESSAGE_CONNECTING);
        connectionStatus = connectGPS();

        if(connectionStatus == 2){
            db.setGPSEnabled(true);
            //LCD.drawString(MESSAGE_CONNECTED, 0, LCDRow);
            //LCD.refresh();
            db.setBTGPSMSG(MESSAGE_CONNECTED);
            Sound.beep();

            boolean flag = true;
            Stopwatch sw = new Stopwatch();

            //While exist a BT Connection (GPS Object
process InputStream)
            while(!(gps.existInternalError()) && (flag)){

                if(sw.elapsed() >= 10000){
                    sw.reset();
                    //Sound.buff();

```



```

        for (int i = 0; i < devList.size(); i++) {
            btrd = ((RemoteDevice) devList.elementAt(i));

            BTDeviceName = btrd.getFriendlyName(true);
            if(BTDeviceName.indexOf(BTPatternName) != -1){
                GPSDevice = btrd;
                GPSDetected = true;
                break;
            }
        }
    }
    return GPSDetected;
}

```

This method can be used to know if you GPS device is turn on. If the method returns true, then you can connect with it.

The another way to connect with a GPS is using a generic routine as the following:

```

static boolean discoverBTDevices(){
    boolean GPSDetected = false;

    LCD.clear();
    LCD.drawString("Searching...", 0, 0);
    LCD.refresh();
    //Make an BT inquire to get all Devices with BT Services
enable
    Vector devList = Bluetooth.inquire(5, 10,cod);

    //If exist GPS Devices near
    if (devList.size() > 0){
        String[] names = new String[devList.size()];
        for (int i = 0; i < devList.size(); i++) {
            RemoteDevice    btrd    =    ((RemoteDevice)
devList.elementAt(i));
            names[i] = btrd.getFriendlyName(true);
        }

        TextMenu searchMenu = new TextMenu(names,1);
        String[] subItems = {"Connect"};
        TextMenu subMenu = new TextMenu(subItems,4);

        int selected;
        do {
            LCD.clear();
            LCD.drawString("Found",6,0);
            LCD.refresh();
            //Menu 1: Show all BT Devices
            selected = searchMenu.select();
            if (selected >=0){
                RemoteDevice    btrd    =    ((RemoteDevice)
devList.elementAt(selected));
                LCD.clear();
                LCD.drawString("Found",6,0);
                LCD.drawString(names[selected],0,1);

                LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
                //Menu 2: Show GPS Device
                int subSelection = subMenu.select();
            }
        } while (selected >= 0);
    }
}

```

```

        if (subSelection == 0){
            GPSDetected = true;
            GPSDevice = btrd;
            break;
        }
    } while (selected >= 0);
} else {
    GPSDetected = false;
}
return GPSDetected;
}

```

With this method you will discover all Bluetooth devices then you will choose your GPS device to connect.

Once you know what GPS device you are going to connect, you can use the following method to connect:

```

static int connectGPS(){
    int result;
    //Bluetooth.addDevice(GPSDevice);

    //BTConnection btGPS = null;
    btGPS = Bluetooth.connect(GPSDevice.getDeviceAddr(),
    NXTConnection.RAW,pin);

    if(btGPS == null){
        result = -1;//No connection
    } else {
        result = 1;//Connection Sucessful
    }

    try{
        in = btGPS.openInputStream();
        gps = new GPS(in);
        gps.updateValues(true);//Update values always

        result = 2;//
    } catch (Exception e){
        result = -2;
    }

    return result;
}

```

If you study with detail the code then you will see that the method stablish a Bluetooth connection and return an InputStream object which is used by a GPS Object to process NMEA sentences.

4.5.- NXCcam

4.5.1.- Introduction

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

The NXCcam provides following capabilities:

- Track up to 8 different colorful objects at 30 frames/second
- Configure the NXCcam 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 NXCcam 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.



Mindsensors has created a sourceforge project to de create a tool that you need to install and use to train your NXCcam. <http://nxcview.sourceforge.net/>

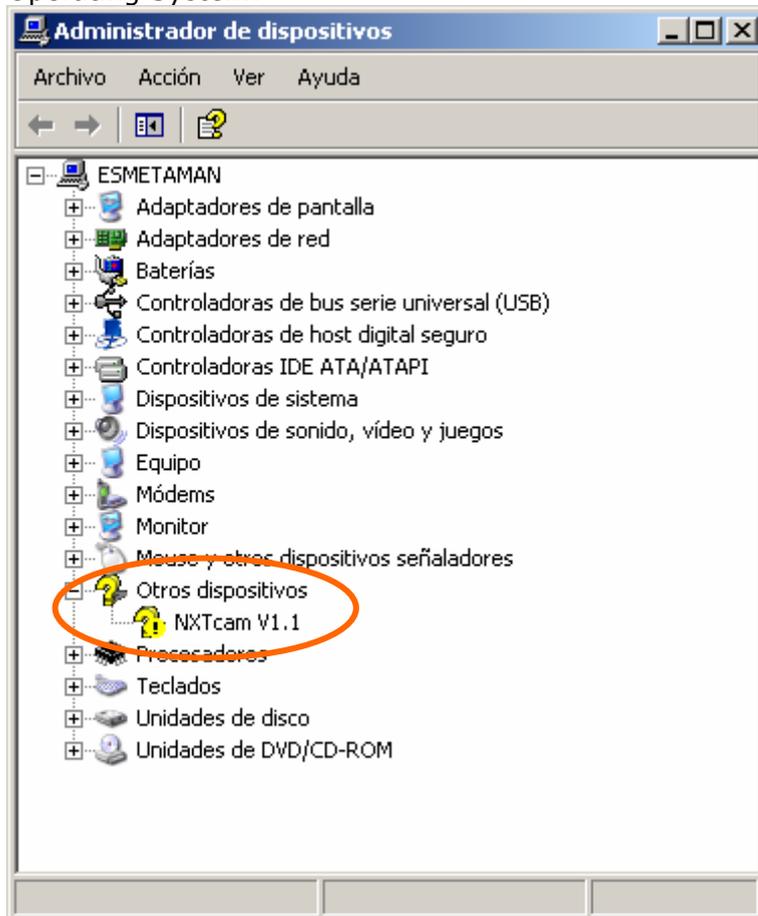
NXCcamView

In this chapter we are going to explain:

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

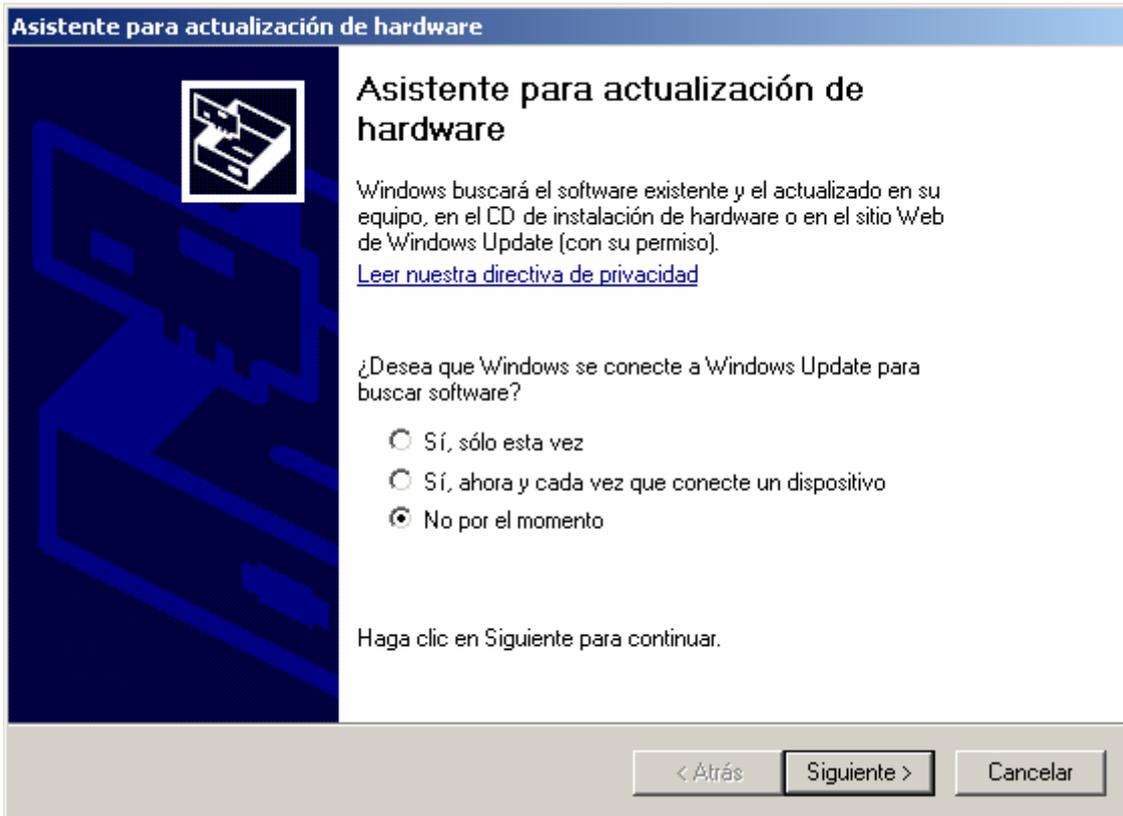
4.5.2.- Install NXTCam driver

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:

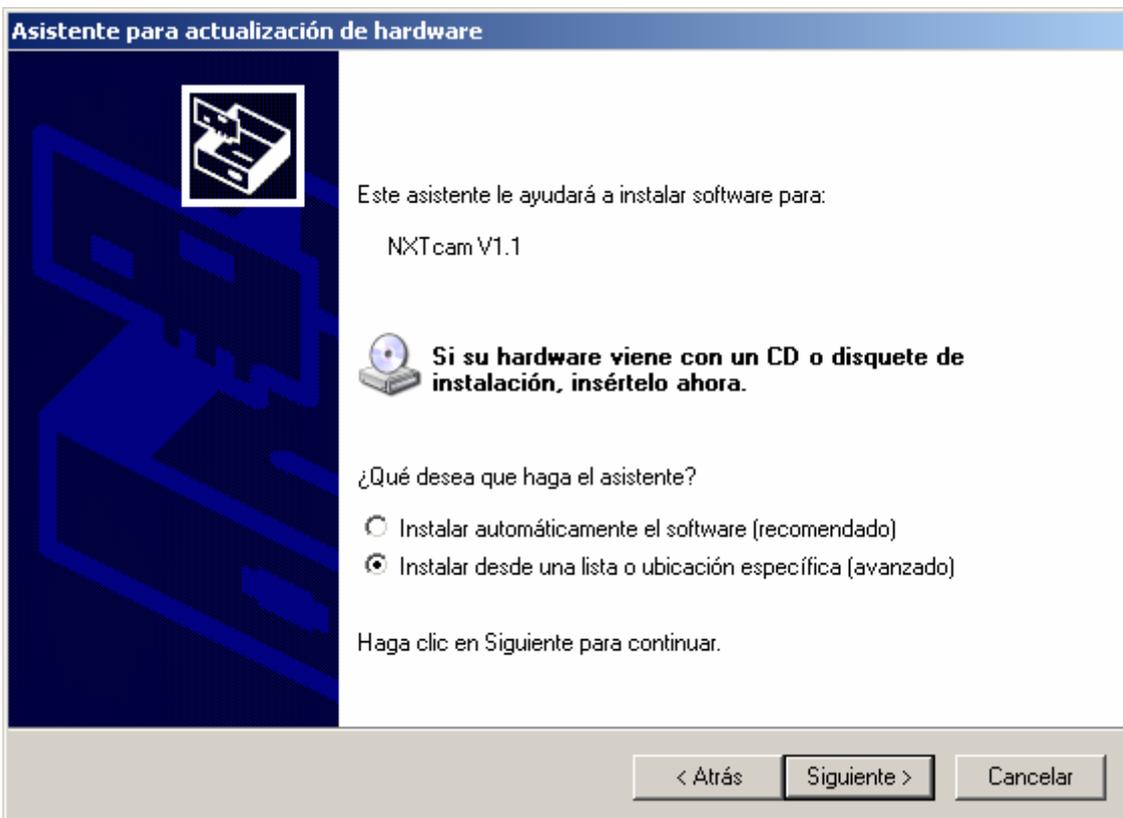


To associate this device with the correct driver, it is necessary to indicate Windows the location of NXTCam Drivers using the assistant.

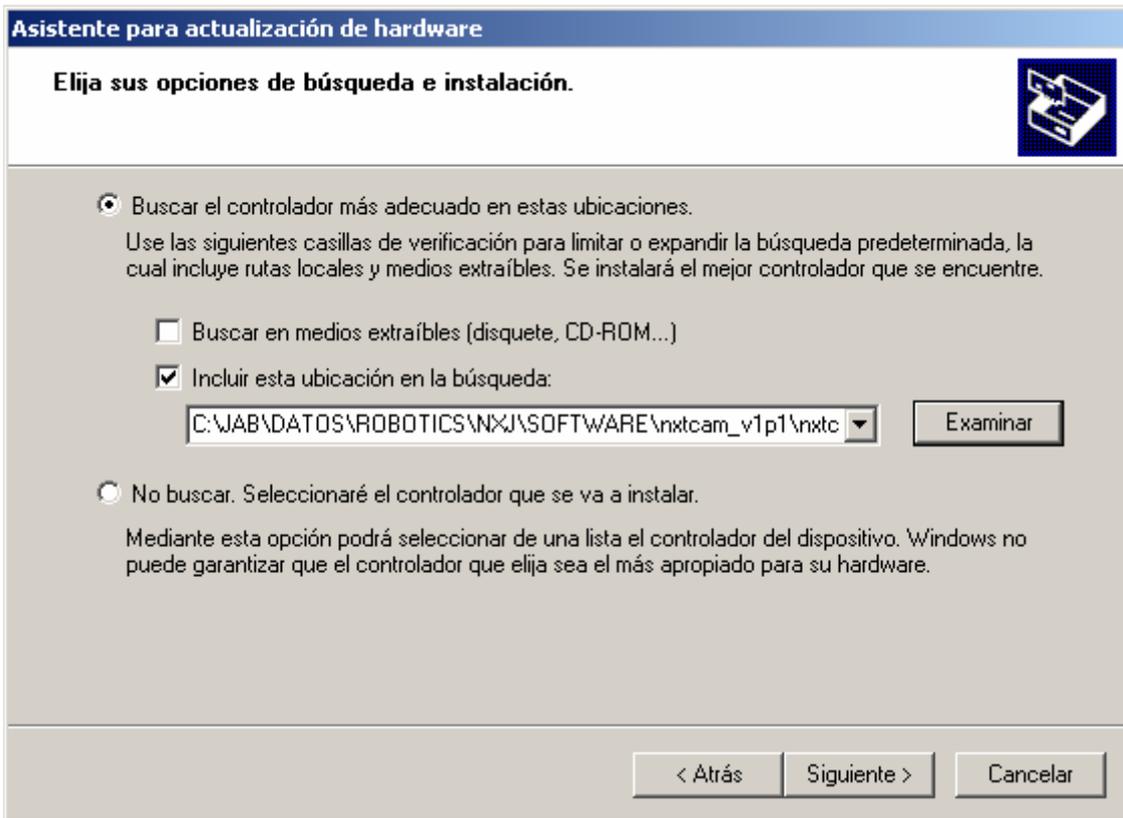
Step1: Choose the option to not use Windows Update to discover NXTCam driver. You can download the driver from Mindsensors website. Use the following URL:



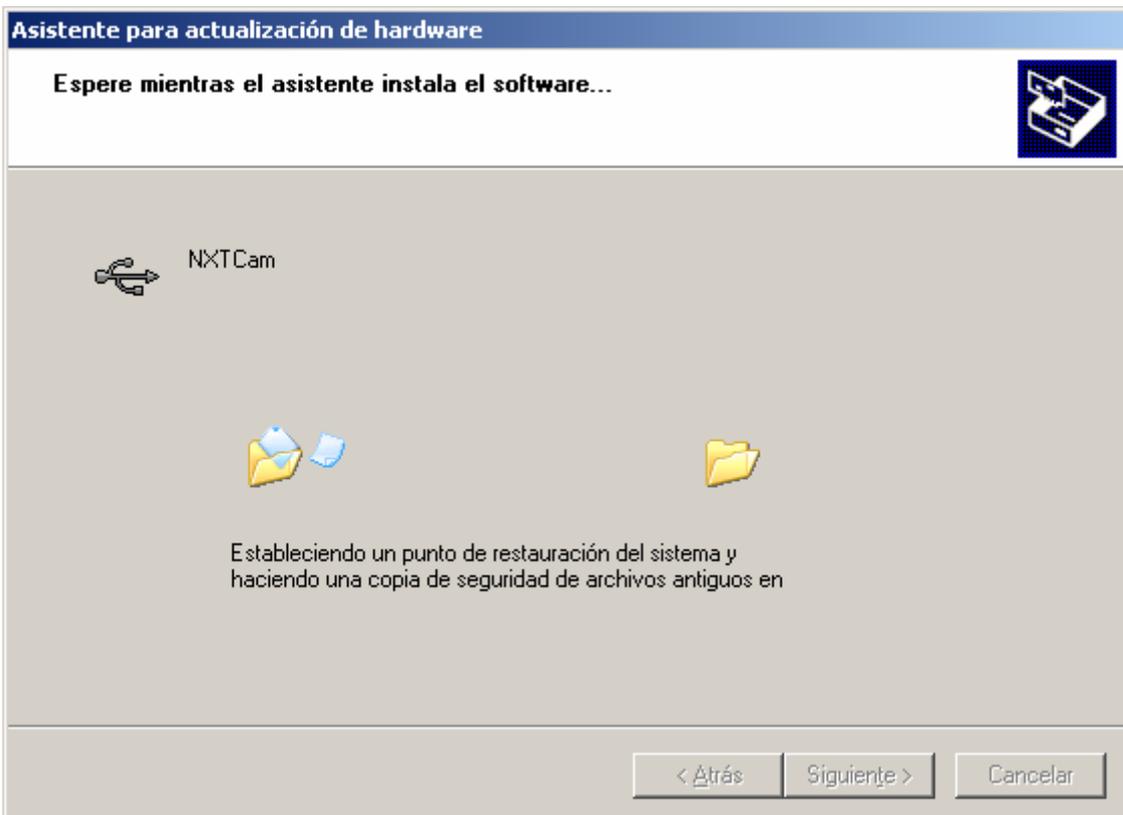
Step2: Edit the path where you have stored NXTCam driver.
 Indicate that you know the location where you have stored the driver.



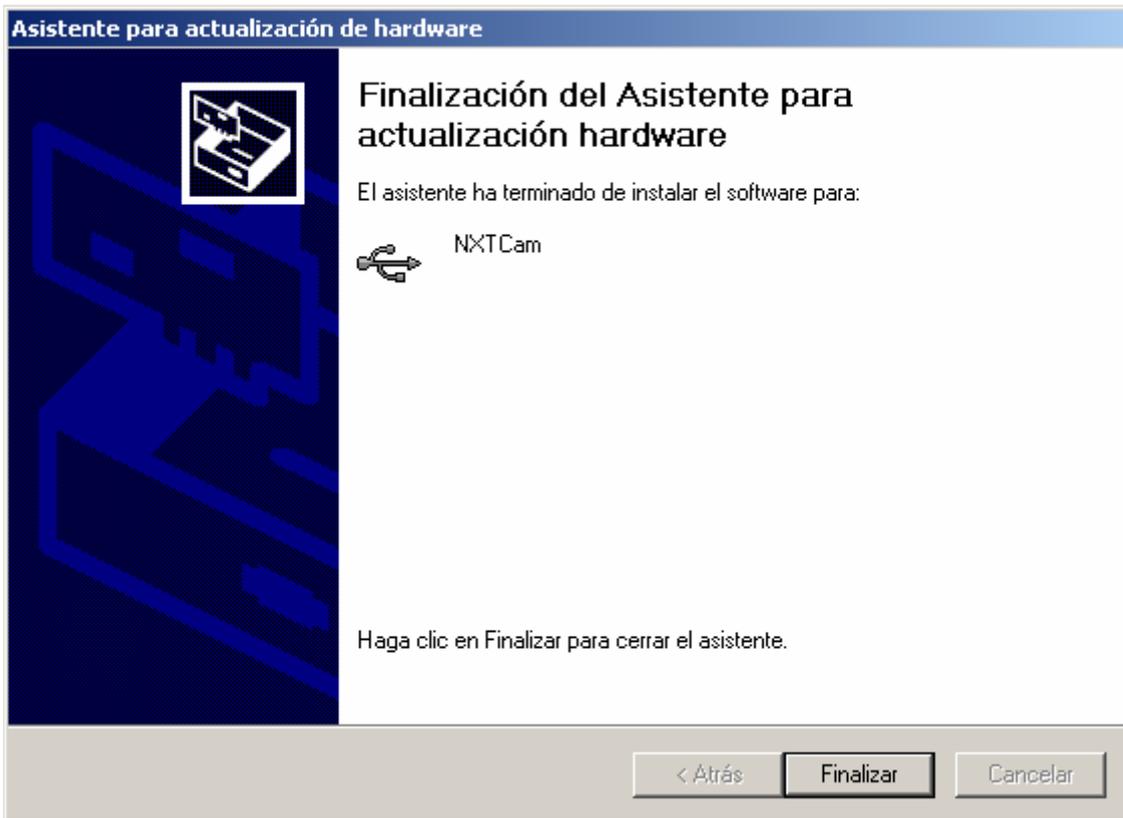
Type the path:



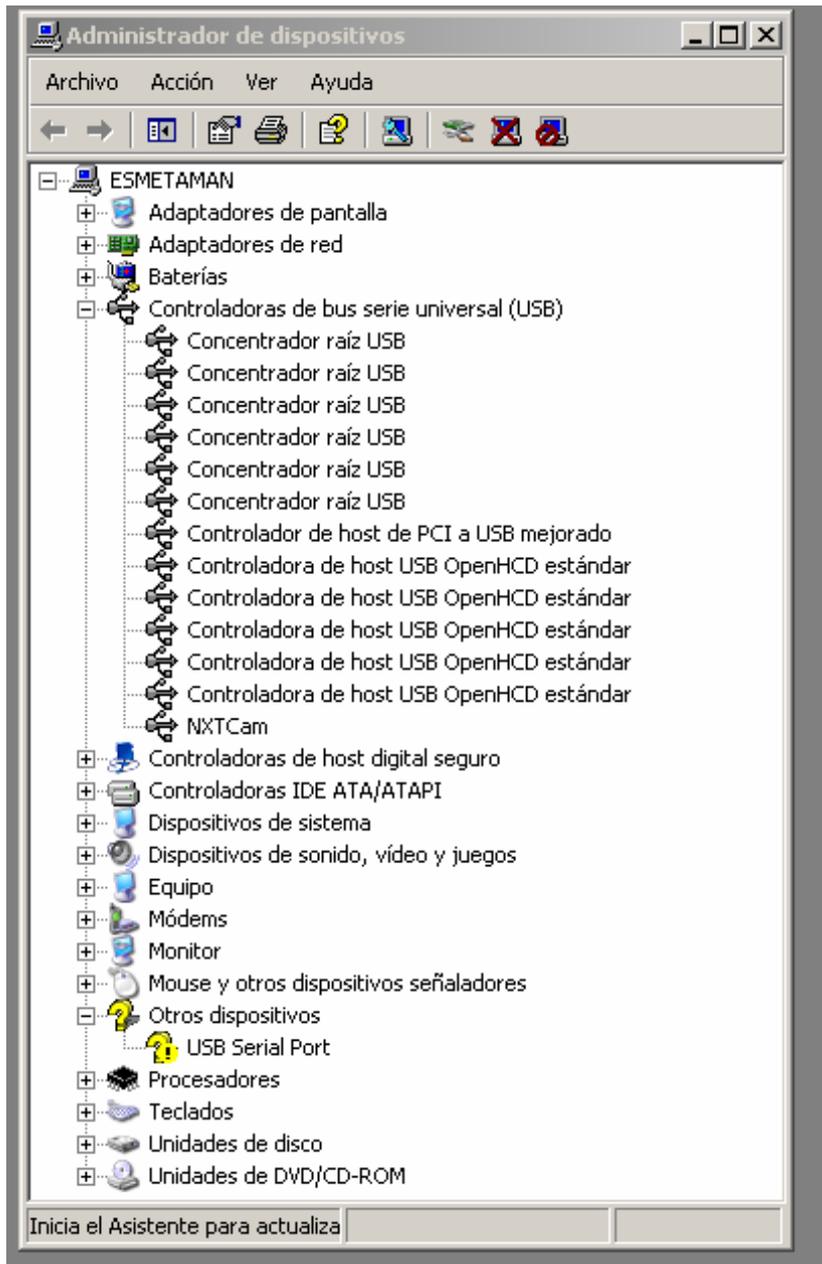
When you click in Next button, Windows will associate NXTCam with the driver.



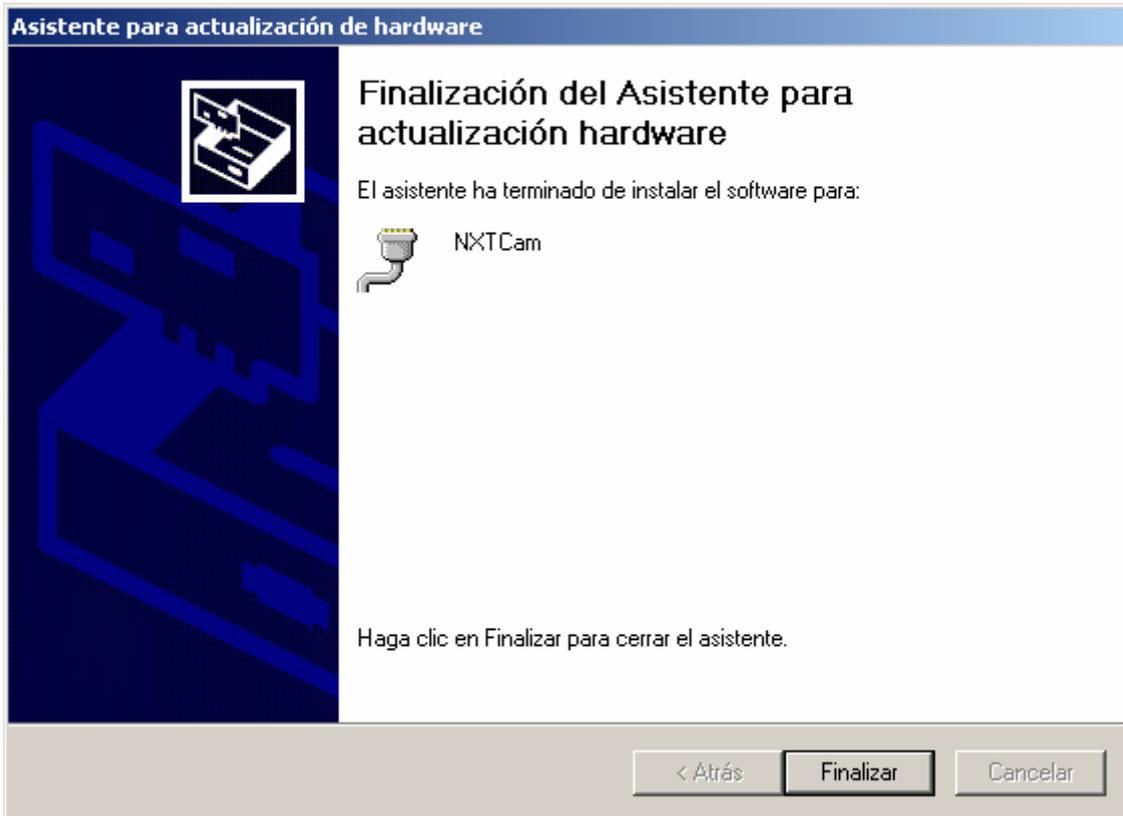
In this moment, your device has been recognized.



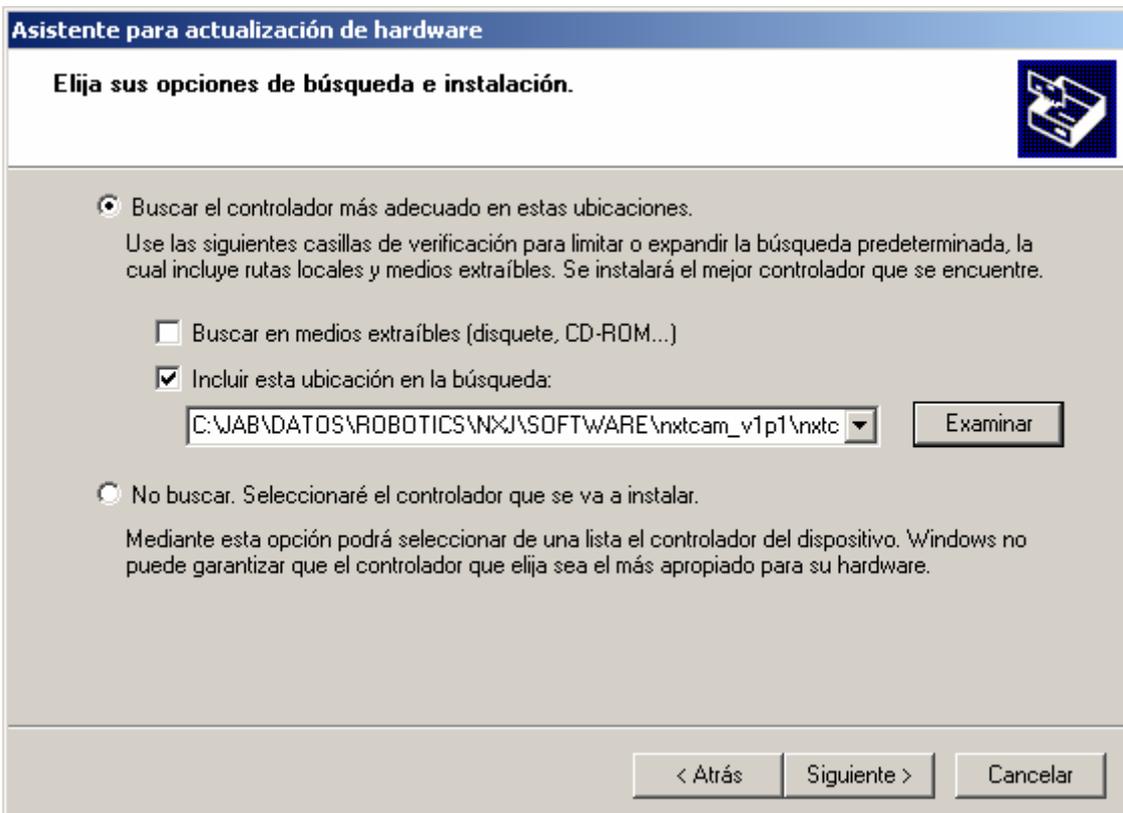
If we try to see Device Manager Window again you will see a new problem, it is necessary to install a driver for USB Serial Port, then we have to repeat the previous task twice to finish.



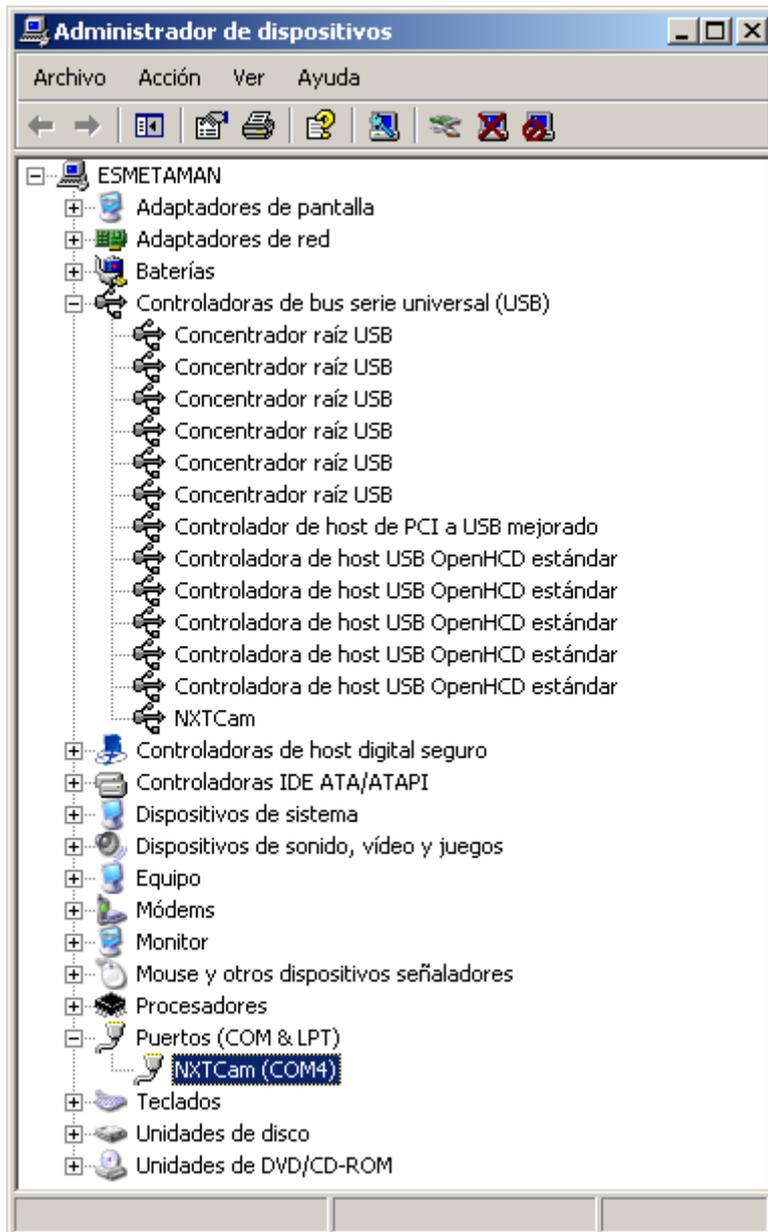
Select the node USB Serial Port in the tree menu and right-click to update the driver then follow the indications showed by the assistant:



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



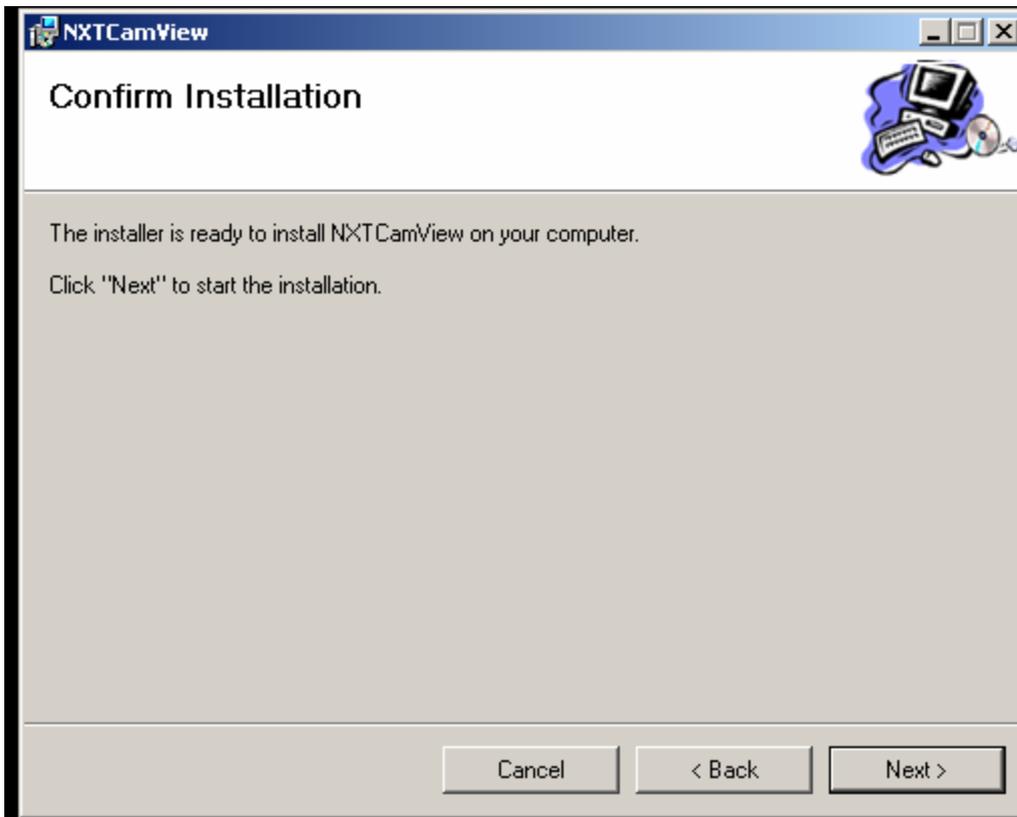
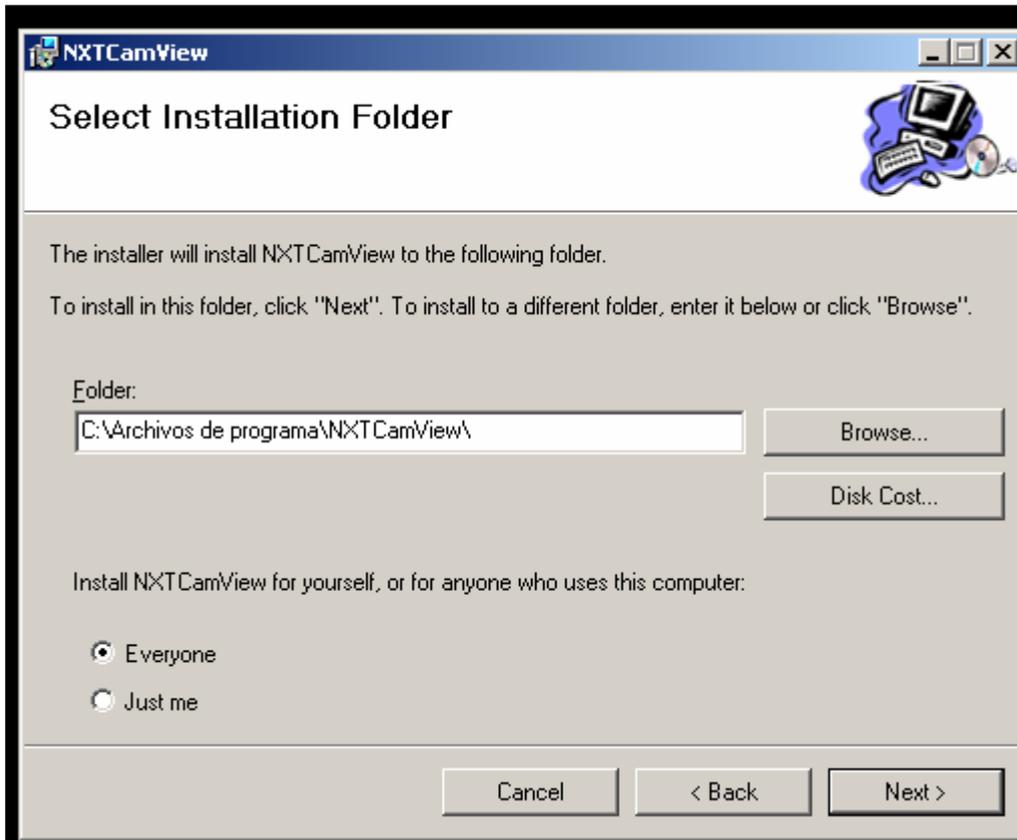
4.5.3.- Install NXTCamView

NXTCamView is a .NET application developed by Mindsensors 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



4.5.4.- 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:

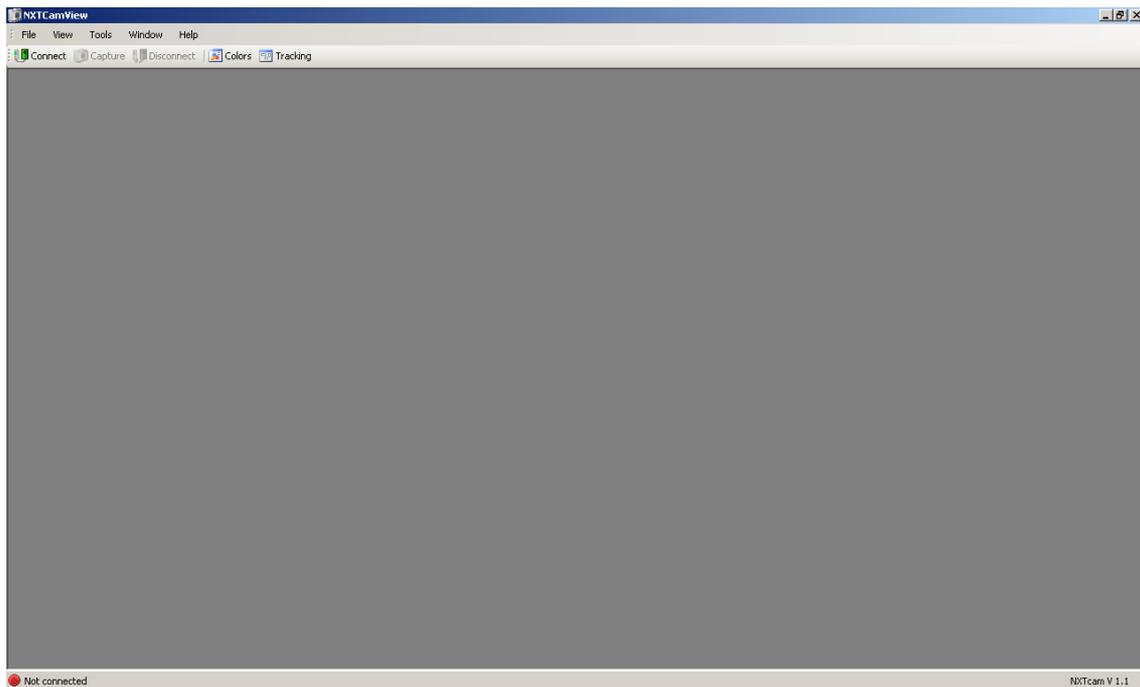
1. Lego Air Tank
2. Fluorescent Text liner

4.5.4.1.- Lego Air Tank

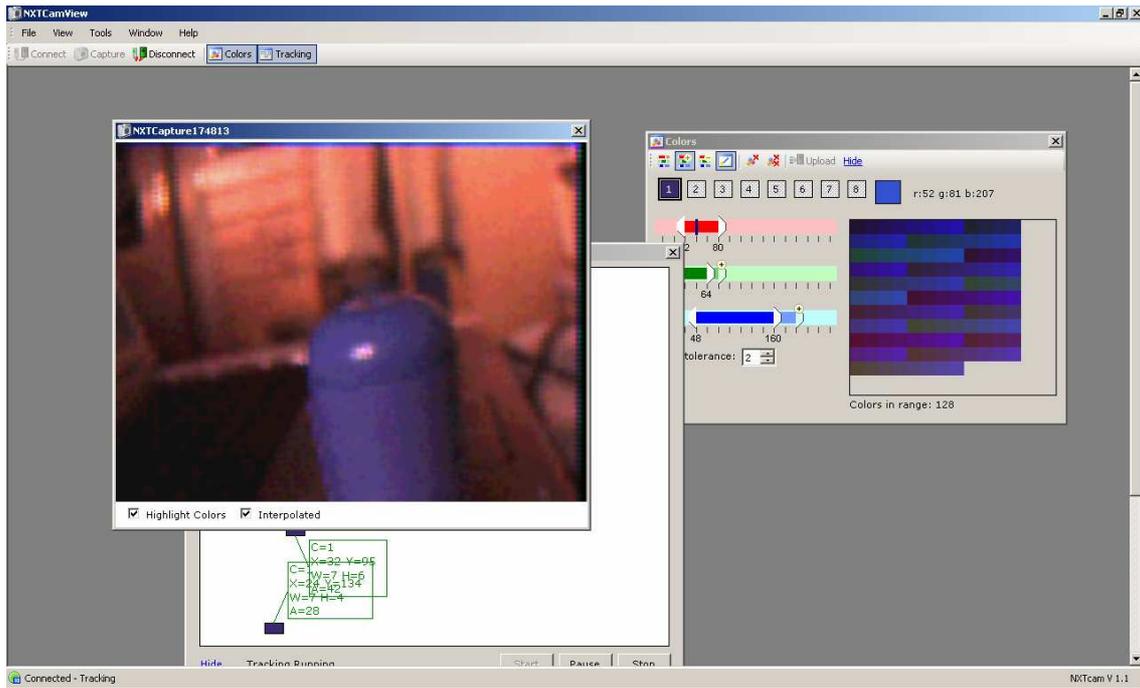
When you use Pneumatic Pieces, one typical piece is a Lego Air Tank:



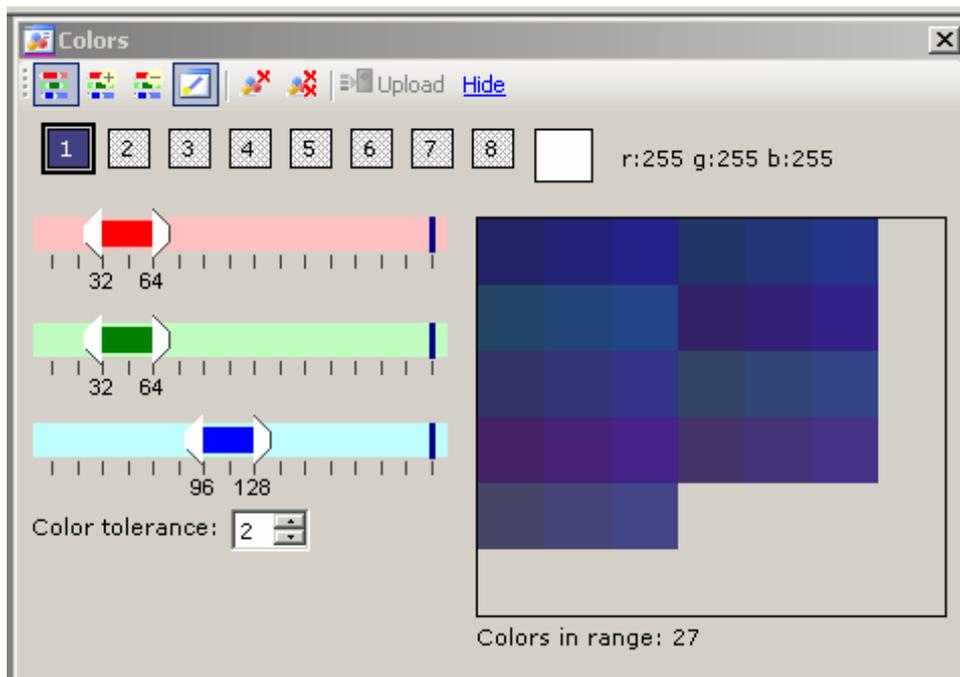
If we want to add a color map to detect Lego Air Tank Pieces then execute NXTCamView and connect NXTCam in your computer.



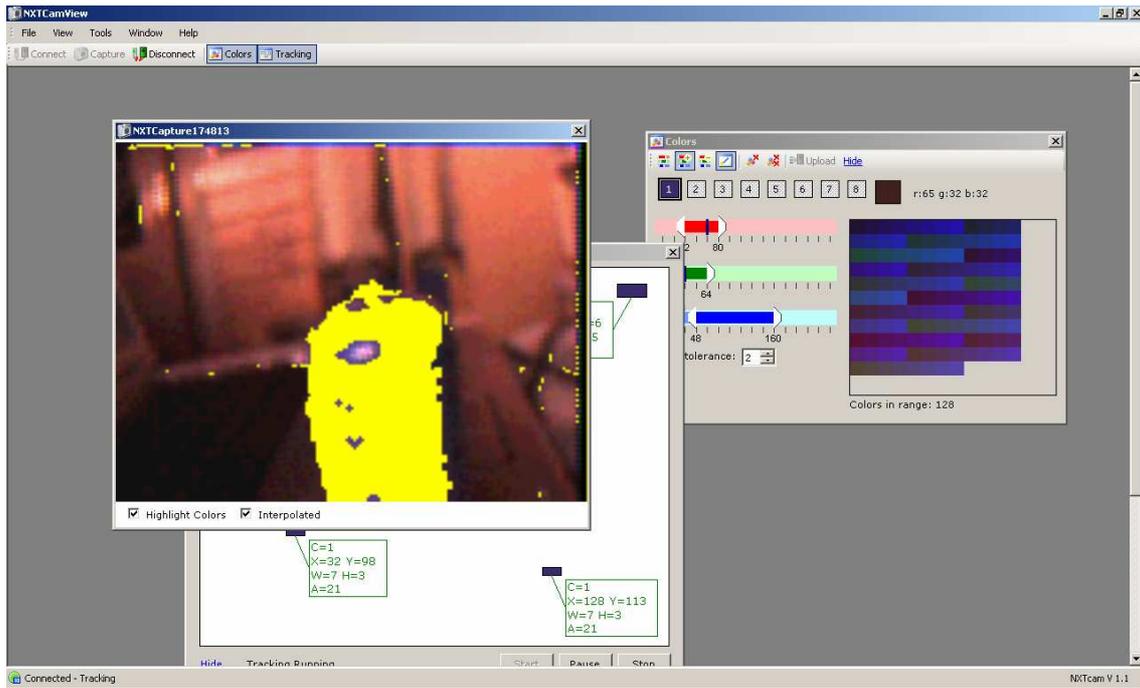
Once you have connected NXTCam, make an image capture:



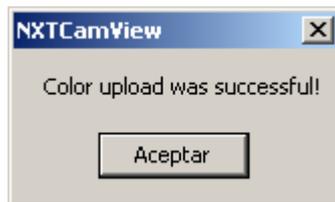
When you have an image that you have taken with NXTCam, create a color pattern using this tool:



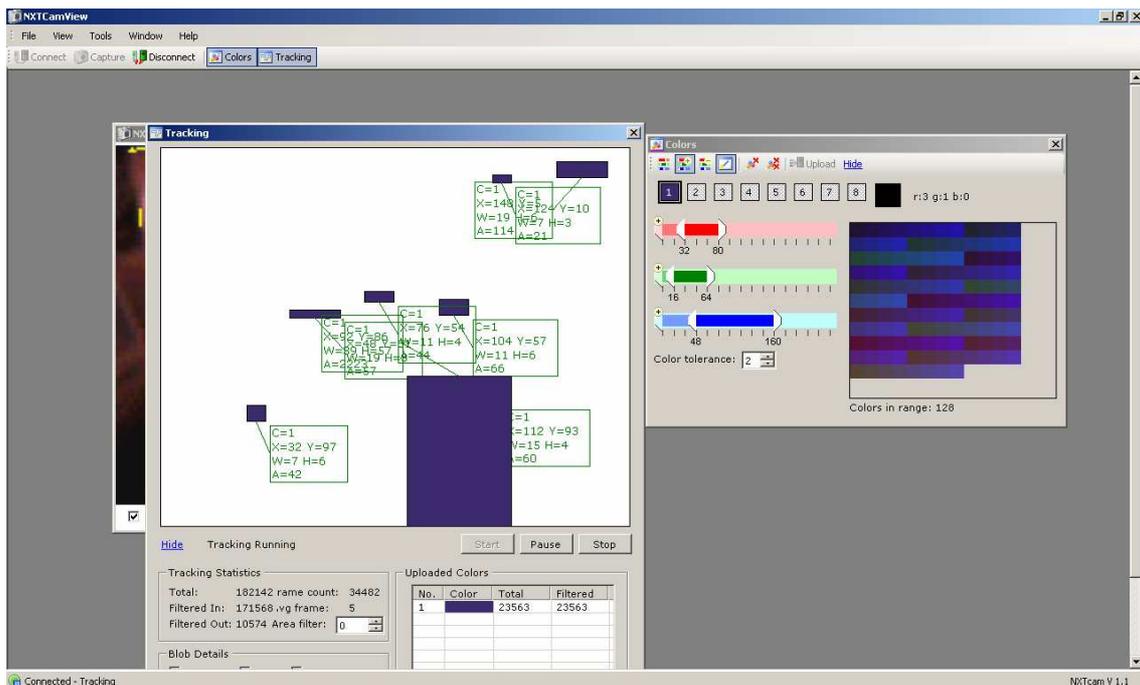
At the end of the process, you have a color pattern as the following:



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:

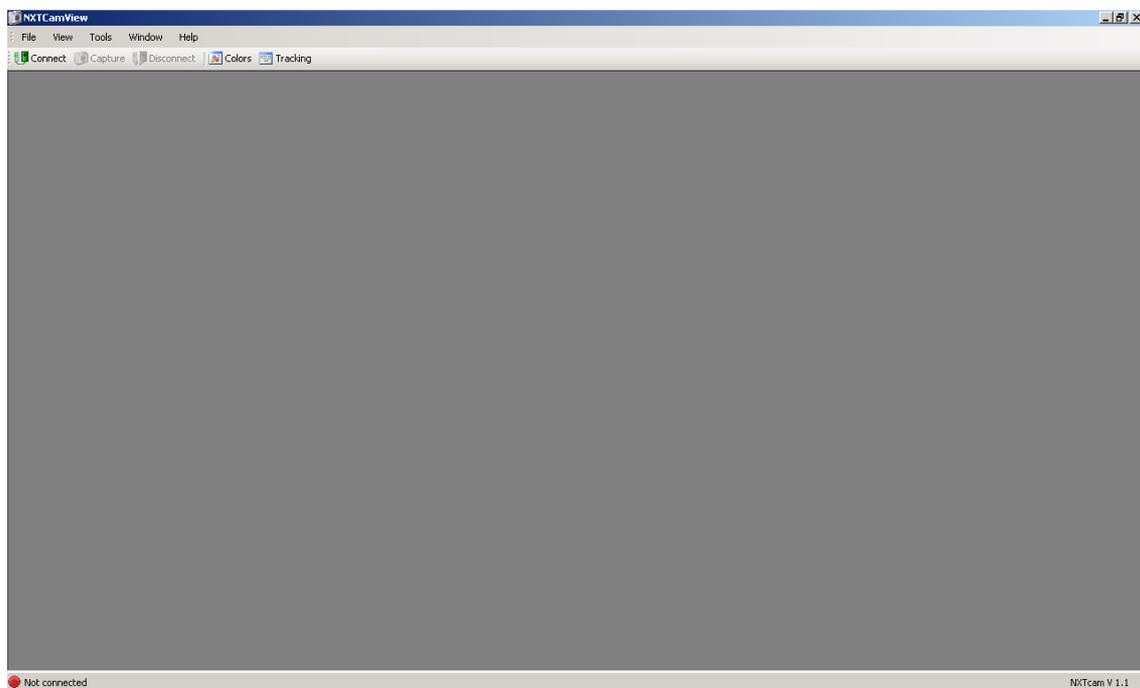


4.5.4.2.- Fluorescent Text liner

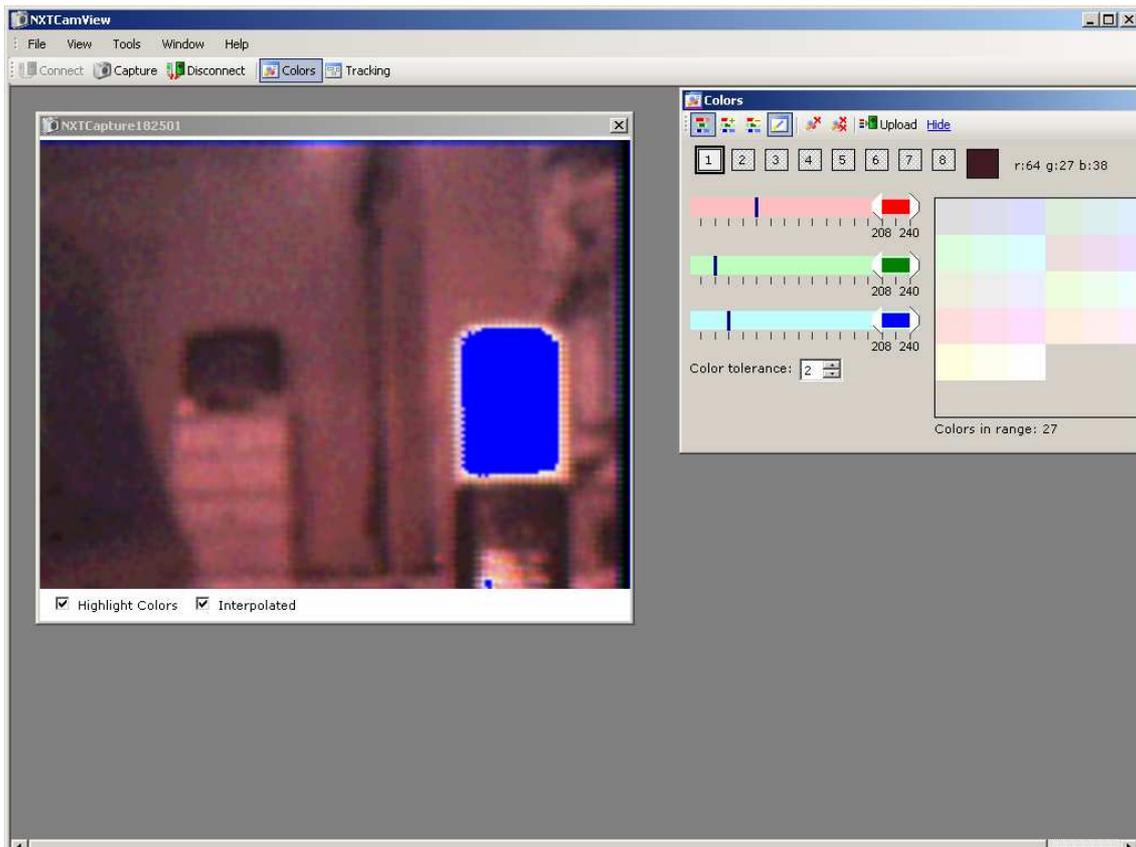
It is very common to have a Fluorescent Text liner at home or in your office, but our training it is a good item because the color is not usual in the environment then we create a pattern to detect this object.



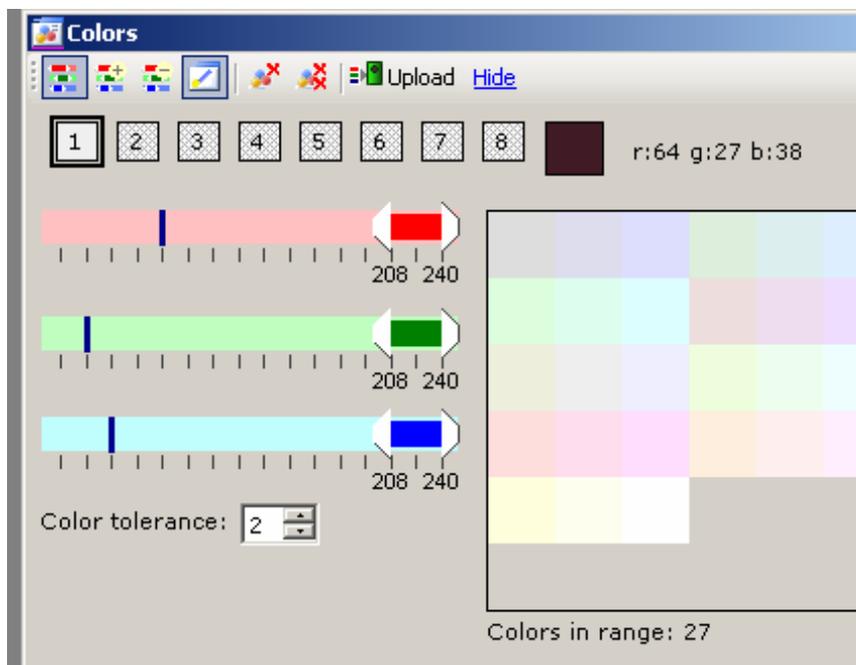
If we want to add a color map to detect Fluorescent Text liner then execute NXTCamView and connect NXTCam in your computer.



Once you have connected NXTCam, make an image capture:

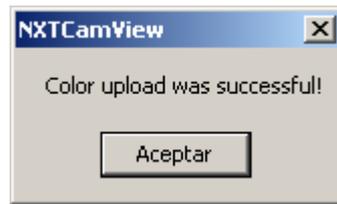


When you have an image that you have taken with NXTCam, create a color pattern using this tool:

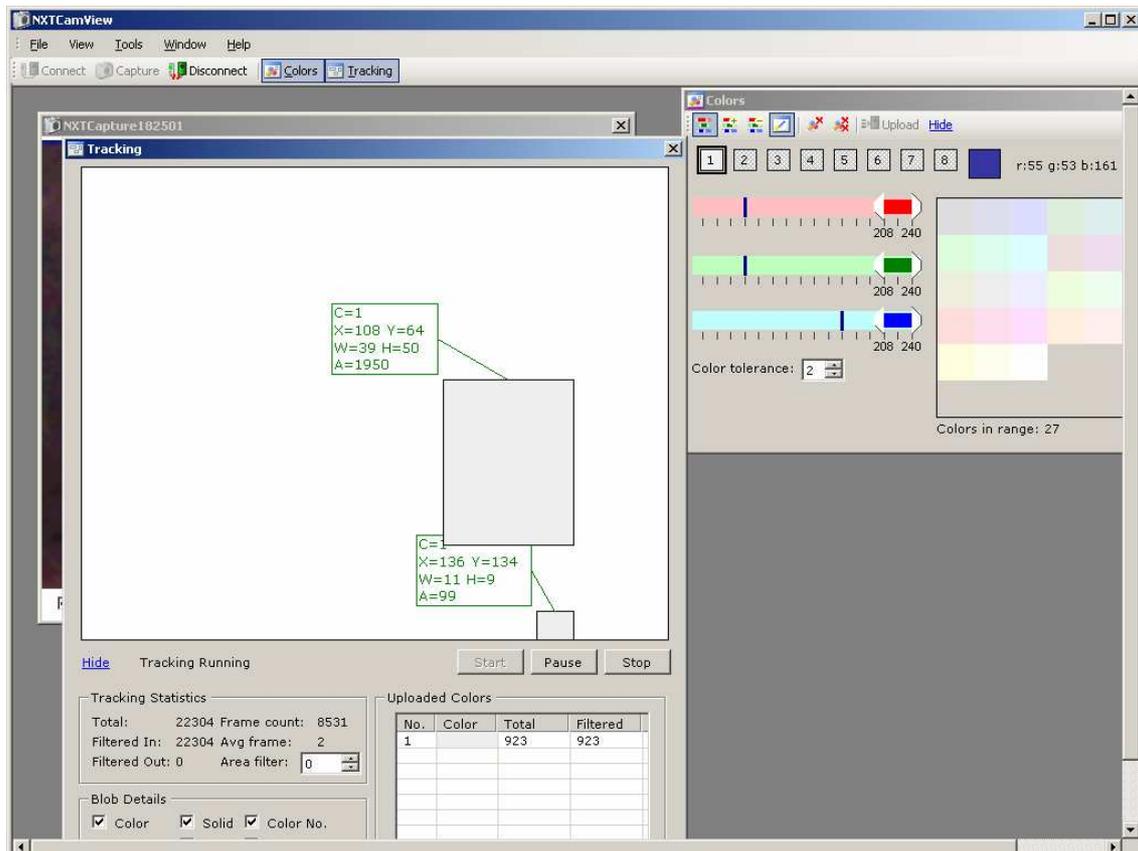


At the end of the process, you have a color.

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)

4.5.5.- 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.

4.5.5.1.- 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 <i>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.</i>
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 - mndsnrs	-	
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	1 st object color	-	This is the first object color as per the sorting method selected.
0x44 ¹	1 st object - X upper left		Upper left X coordinate of first object
0x45	1 st object - Y upper left		Upper left Y coordinate of first object
0x46	1 st object - X lower right		Lower right X coordinate of first object
0x47 ²	1 st object - Y lower right		Lower right Y coordinate of first object
0x48	2 nd object color		
0x49-0x4C	2 nd object co-ordinates		
0x4D	3 rd object color		
0x4E-0x51	3 rd object co-ordinates		
0x52	4 th object color		
0x53-0x56	4 th object co-ordinates		
0x57	5 th object color		
0x58-0x5B	5 th object co-ordinates		

Register	Read	Write	Comments
0x5C	6 th object color		
0x5D-0x60	6 th object co-ordinates		
0x61	7 th object color		
0x62-0x65	7 th object co-ordinates		
0x66	8 th object color		
0x67-0x6A	8 th 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	1 st Camera register Address	1 st Camera register Address	
0x6D ³	1 st Camera register Data	1 st Camera register Data	1 st register Data read from image sensor or written to image sensor
.....	
0x7A	8 th Camera register Address	8 th Camera register Address	
0x7B	8 th Camera register Data	8 th 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	

4.6.- NXTLine

NXTLine sensor is a new sensor from Mindsensors which has been designed to read values from a grid of 8 light sensors. This sensor is perfect to solve the classic challenge follow the line.



4.7.- Touch sensor

The Touch Sensor gives your robot a sense of touch. The Touch Sensor detects when it is being pressed by something and when it is released again



An example using this sensor is the following:

```
import lejos.navigation.TachoNavigator;
import lejos.nxt.*;

public class FindCatch{

    public static void main(String [] args)throws Exception {
        TachoNavigator tn=new
        TachoNavigator(5.5f,11.2f,Motor.B,Motor.C,false);
        UltrasonicSensor uss = new UltrasonicSensor(SensorPort.S3);
        TouchSensor ts=new TouchSensor(SensorPort.S1);
        LightSensor ls=new LightSensor(SensorPort.S4);

        int distance =30;
        for (int i=0;i<6;i++) {
            Sound.beep();
            try { Thread.sleep(500); } catch (Exception e) {}
        }

        Motor.C.setSpeed(400);
        Motor.B.setSpeed(400);
        Motor.C.backward();
        Motor.B.forward();
        while (Button.readButtons()==0) {
            float dist = uss.getDistance(); //distance
            calculate
            if (dist <= distance) {
                tn.stop();

                Motor.A.setSpeed(200);
                Motor.A.forward();
                try {
                    Thread.sleep(2000);
                } catch (Exception e) {
                }
                tn.setSpeed(400);
                tn.forward();
            }else if(ts.isPressed()){ //if touch is pressed
                Sound.beep();

                tn.stop();
                LCD.drawString("Object Found", 3, 4);
                try {
                    Thread.sleep(3000);
                } catch (Exception e) {
                }
            }
            Motor.A.setSpeed(200);
            Motor.A.backward();
            int colorval=ls.readValue();//learn color value
```




An example using sound sensors is the following:

```
public class Listen implements SensorPortListener
{
    String changed = "State changed";
    String val = "Value:";
    String oldVal = "old Value:";
    String free = "Free Mem:";
    SoundSensor sound = new SoundSensor(SensorPort.S1);

    public static void main (String[] aArg)
    throws Exception
    {
        Listen listen = new Listen();
        listen.run();
        Button.ESCAPE.waitForPressAndRelease();
        LCD.clear();
        LCD.drawString("Finished", 3, 4);
        LCD.refresh();
        Thread.sleep(2000);
    }

    public void stateChanged(SensorPort port, int value, int
oldValue)
    {
        if (port == SensorPort.S1 && sound.readValue() > 50)
        {
            LCD.clear();
            LCD.drawString(changed,0,0);
            LCD.drawString(val, 0, 1);
            LCD.drawInt(value,7,1);
            LCD.drawInt(sound.readValue(), 12, 1);
            LCD.drawString(oldVal, 0, 2);
            LCD.drawInt(oldValue, 11, 2);
            LCD.drawString(free, 0, 4);

            LCD.drawInt((int)(Runtime.getRuntime().freeMemory()),10,4);
            LCD.refresh();
        }
    }

    private void run()
    throws InterruptedException
    {
        SensorPort.S1.addSensorPortListener(this);
    }
}
```

4.9.- Summary

In this chapter you learn how to use many sensors which you have with your Lego Mindstorms NXT Kit or adquired with your favorite provider.

5.- Actuators

5.1.- Introduction

LeJOS is able to manage several kind of actuators.

In this section we explain how to use the following actuators:

- NXT Motors
- PF Motors
- RC Servos and DC Motors
- RC Servo systems

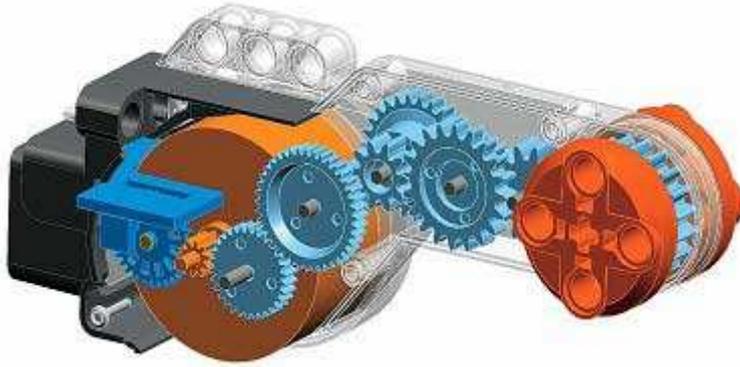


5.2.- NXT Motors

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();
```

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.

Further information about Motor class here:

http://lejos.sourceforge.net/p_technologies/nxt/nxj/api/lejos/nxt/Motor.html

5.3.- PF Motors

In 2008, lego launched a new kind of motors which have the feature that they use their own battery to get the energy. Lego Mindstorms NXT can be manage this kind of motors using the sensor PFMate from Mindsensors. It is true that it is possible to manage PF motors with others sensors as IRLink but PFMate has been designed to manage PF IR Controller easily.



In the following example, you will learn how to manage PF Motors with the sensor PFMate easily.



```
import lejos.nxt.*;
import lejos.nxt.addon.*;
import lejos.util.*;

/**
 * @author Juan Antonio Brenha Moral
 *
 */
```

```

public class PFMateTest{

    private static PFMate pfObj;
    private static DebugMessages dm;
    private static final int PFDelayCMD = 50; //ms.

    public static void main(String[] args){
        pfObj = new PFMate(SensorPort.S1,1);

        dm = new DebugMessages();
        dm.setLCDLines(6);
        dm.setDelayEnabled(true);

        int i = 7;
        pfObj.A.setSpeed(i);
        try {Thread.sleep(PFDelayCMD);} catch (Exception e) {}
        pfObj.B.setSpeed(i);

        while(!Button.ESCAPE.isPressed()){
            dm.echo(i);

            while(Button.ENTER.isPressed()){
                pfObj.A.forward();
                try {Thread.sleep(PFDelayCMD);} catch
(Exception e) {}

                pfObj.B.forward();

                Sound.beep();
            }

            while(Button.LEFT.isPressed()){
                if(i >1){
                    i--;
                }else{
                    i = 1;
                }

                pfObj.A.setSpeed(i);
                try {Thread.sleep(PFDelayCMD);} catch
(Exception e) {}

                pfObj.B.setSpeed(i);

                try {Thread.sleep(500);} catch (Exception e) {}
            }

            while(Button.RIGHT.isPressed()){
                if(i <7){
                    i++;
                }else{
                    i = 7;
                }

                pfObj.A.setSpeed(i);
                try {Thread.sleep(PFDelayCMD);} catch
(Exception e) {}

                pfObj.B.setSpeed(i);

                try {Thread.sleep(500);} catch (Exception e) {}
            }
        }
    }
}

```

```

    pfObj.A.stop();
    try {Thread.sleep(PFDelayCMD);} catch (Exception e) {}
    pfObj.B.stop();
}
}

```

5.4.- RC Servos and DC Motors with NXTe/LSC

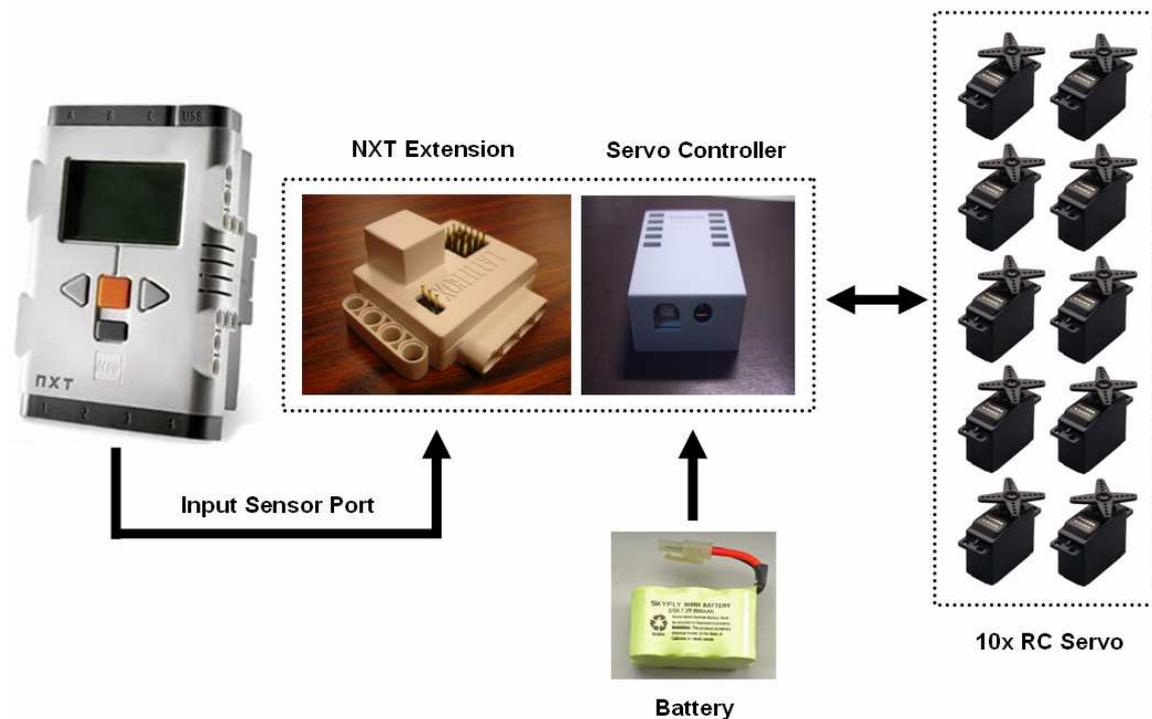
5.4.1.- Introduction

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.

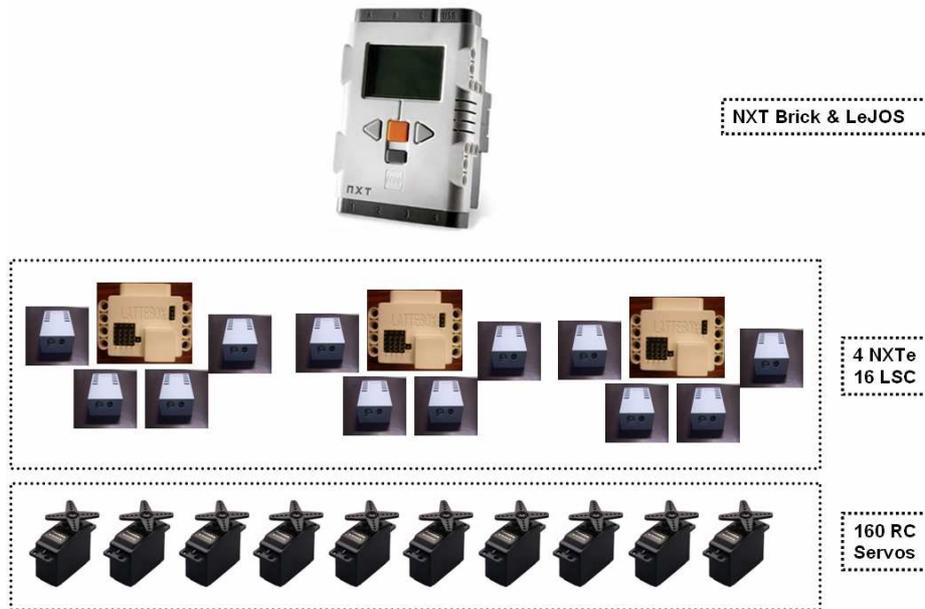
5.4.2.- NXTe architecture

The NXTe architecture is the following:



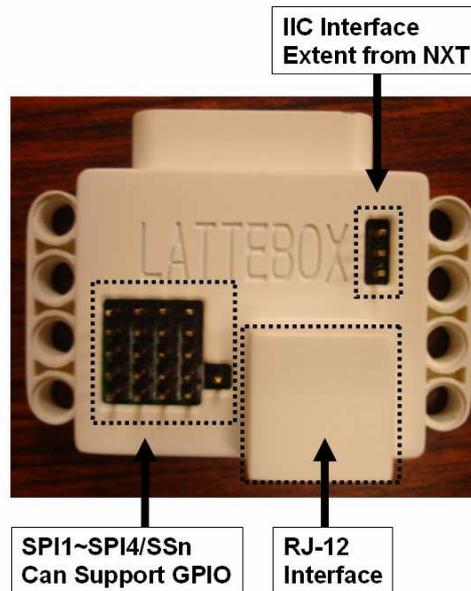
If you notice, with NXTe technology you can control:

$$4x \text{ NXTe} * 4x \text{ LSC} * 10x \text{ RC Servos} = 160 \text{ Servos}$$



5.4.2.1.- NXT Extension, NXTe

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



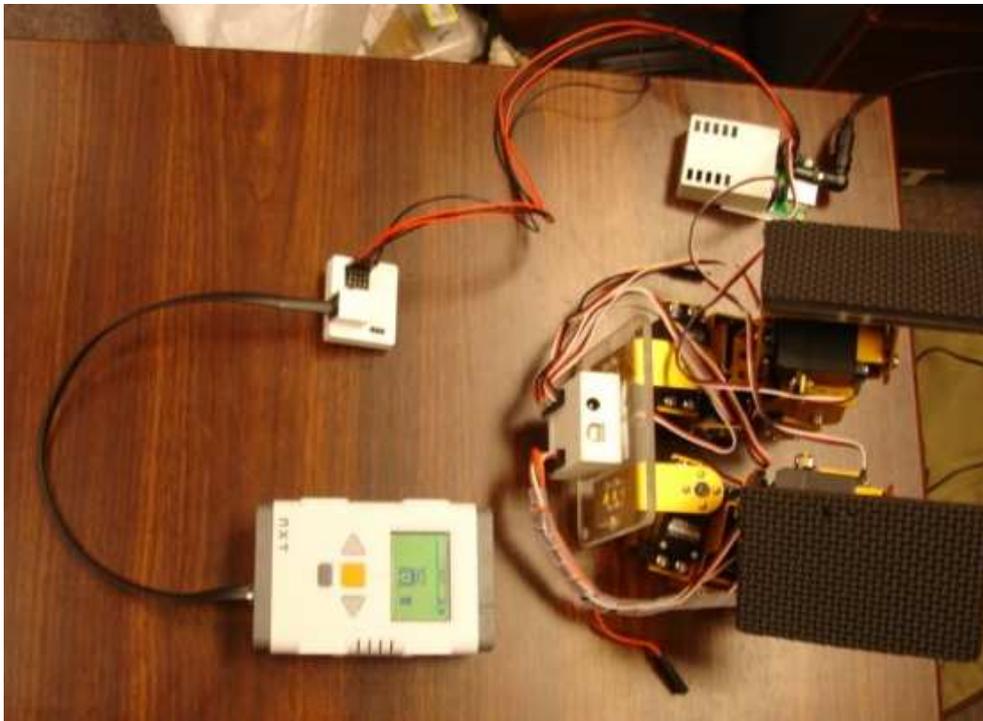
5.4.2.2.- Servo Controller

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



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

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



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

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

5.4.3.- 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.



5.4.3.1.- 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.

5.4.4.- LeJOS and NXTe/LSC

5.4.4.1.- NXTe/LSC Support in LeJOS

Currently leJOS support NXTe but it is necessary 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

5.4.4.2.- Example using NXTe/LSC 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
            plugged 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()){
        NXTObj.LSC(0).Servo(0).goToMaxAngle();
        NXTObj.LSC(0).Servo(1).goToMaxAngle();
        while(NXTObj.LSC(0).Servo(0).isMoving()
== true){}
        angle =
NXTObj.LSC(0).Servo(0).getAngle();
        angle =
NXTObj.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");
}
}

```

5.5.- RC Servos with NXTServo

NXTServo is a new sensor designed by Mindsensors to manage RC Servos. NXTServo has the feature to measure the battery.

```

import lejos.nxt.addon.*;
import lejos.nxt.*;

/**
 * Example designed to test Mindsensors NXT Servo
 *
 * @author Juan Antonio Brenha Moral
 *
 */
public class NXTServoTest{
    private static String appName = "NXTServo Test";
    private static String appVersion = "v0.3";

    private static MSC msc;

    public static void main(String[] args){
        LCD.drawString(appName, 0, 0);
        LCD.drawString("#####", 0, 2);
        LCD.drawString("#####", 0, 6);

        msc = new MSC(SensorPort.S1);
        //Set to initial angle
        msc.servo1.setAngle(90);
    }
}

```

```

int angle = 0;
int pulse = 0;
int NXTServoBattery = 0;

while(!Button.ESCAPE.isPressed()){
    NXTServoBattery = msc.getBattery();

    if (Button.LEFT.isPressed()){
        angle = 0;
        msc.servo1.setAngle(angle);
    }

    if (Button.ENTER.isPressed()){
        angle = 90;
        msc.servo1.setAngle(angle);
    }

    if (Button.RIGHT.isPressed()){
        angle = 180;
        msc.servo1.setAngle(angle);
    }

    clearRows();
    LCD.drawString("Battery: " + NXTServoBattery, 0, 3);
    LCD.drawString("Pulse:      " + msc.servo1.getPulse(),
0, 4);
    LCD.drawString("Angle:      " + msc.servo1.getAngle(),
0, 5);
    LCD.refresh();
}

//Set to initial angle
msc.servo1.setAngle(90);

LCD.drawString("Test finished",0,7);
LCD.refresh();
try {Thread.sleep(1000);} catch (Exception e) {}
credits(3);
System.exit(0);
}

/**
 * Internal method used to clear some rows in User Interface
 */
private static void clearRows(){
    LCD.drawString("          ", 0, 3);
    LCD.drawString("          ", 0, 4);
    LCD.drawString("          ", 0, 5);
}

/**
 * Final Message
 *
 * @param seconds
 */
private static void credits(int seconds){
    LCD.clear();
    LCD.drawString("LEGO Mindstorms",0,1);
    LCD.drawString("NXT Robots  ",0,2);
    LCD.drawString("run better with",0,3);
    LCD.drawString("Java leJOS",0,4);
}

```

```
LCD.drawString("www.lejos.org",0,6);  
LCD.refresh();  
try {Thread.sleep(seconds*1000);} catch (Exception e) {}  
}  
}
```

5.6.- RC Servos systems with MRS H01

MRS H01 is a hexapod robot designed by Micromagic which has 2 ways to communicate with it:

- Bluetooth
- I2C

MRS H01 is built with 18 RC Servos. NXT brick is able to manage them using BrainEngine a Micromagic chip which manage Inverted Kinematics equations.



5.6.1.- Using MRS H01 with Bluetooth

Not Available

5.6.2.- Using MRS H01 with I2C

HexEngine can be managed by a NXT brick with I2C protocol. Connect your NXT brick with HexEngine with the following way:



5.7.- Summary

In this chapter you learnt how to use actuators with NXT. Currently only NXT motors allow calculating odometry. Exist a way to calculate odometry with PF Motors if you use the old Rotation sensor.

RC Servos and DC Motors offers new features for your NXT robots.

6.- Graphical user interfaces with leJOS

6.1.- Introduction

With leJOS project, you can develop GUI for your project. It is really easy manage the display with the class LCD or design complex GUI with the package javax.microedition.lcdui

6.2.- LCD

NXT brick has a display with 7 rows and 22 columns.

LCD is really useful class to show values from your robot status. In the following example, you will see how to use the class LCD.

```
import lejos.nxt.*;

public class HelloWorld{
    private static final String welcomeMessage = "Hello World";
    public static void main (String[] args){
        //System.out.println("Hello World");
        LCD.drawString(welcomeMessage,0,0);
        LCD.refresh();
    }
}
```

6.3.- System.out

LeJOS offers the way to use the display as a shell console.

```
import lejos.nxt.*;

public class HelloWorld2{
    private static final String welcomeMessage = "Hello World";
    public static void main (String[] args){
        int i= 0;
        while(!Button.ESCAPE.isPressed()){
            i++;
            System.out.println("Hello World " + i);
            //LCD.drawString(welcomeMessage,0,0);
            //LCD.refresh();
        }
    }
}
```

6.4.- RConsole

LeJOS offers the way to use the display as a shell console.

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

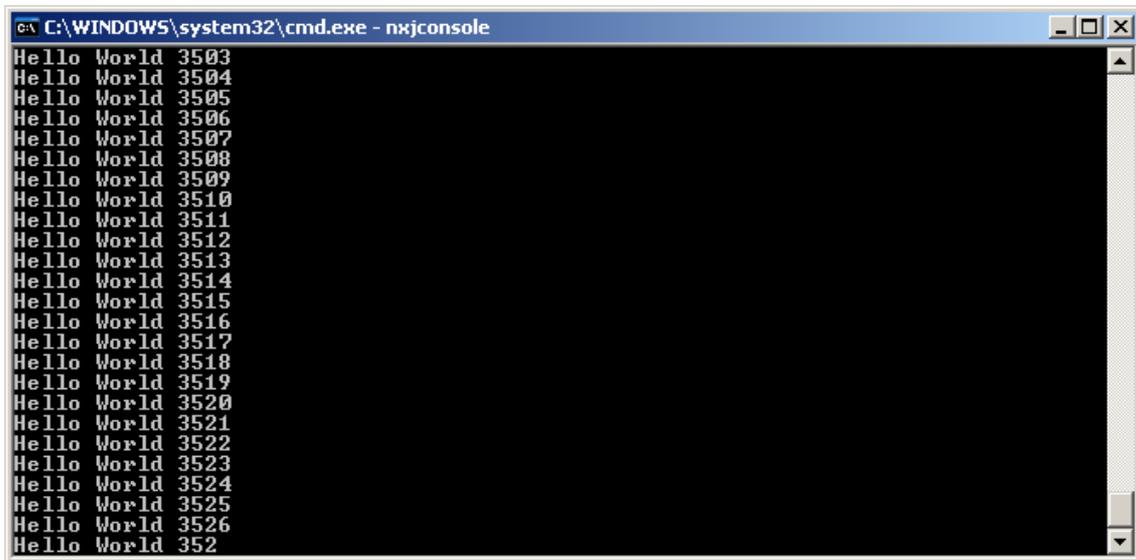
```

import lejos.nxt.comm.*;

public class RConsoleTest {
    public static void main(String[] args){
        int i= 0;
        RConsole.open();
        while(!Button.ESCAPE.isPressed()){
            i++;
            RConsole.println("Hello World " + i);
        }
        RConsole.close();
    }
}

```

To see the results, open a shell console and type nxjconsole



```

C:\WINDOWS\system32\cmd.exe - nxjconsole
Hello World 3503
Hello World 3504
Hello World 3505
Hello World 3506
Hello World 3507
Hello World 3508
Hello World 3509
Hello World 3510
Hello World 3511
Hello World 3512
Hello World 3513
Hello World 3514
Hello World 3515
Hello World 3516
Hello World 3517
Hello World 3518
Hello World 3519
Hello World 3520
Hello World 3521
Hello World 3522
Hello World 3523
Hello World 3524
Hello World 3525
Hello World 3526
Hello World 352

```

6.5.- Learning javax.microedition.lcdui

The package javax.microedition.lcdui is nice package to manage the display. This package manages the following classes:

1. Alert
2. ChoiceGroup
3. Command
4. Display
5. Displayable
6. Font
7. Form
8. Gauge
9. Graphics
10. Image
11. Item
12. List
13. Screen
14. Spacer
15. StringItem
16. TextBox
17. TextField
18. Ticker

6.5.1.- Using the class Graphics

In some projects, it is possible that use the class Graphics.

```
import java.util.*;
import lejos.nxt.*;
import javax.microedition.lcdui.Graphics;

public class RADAR1_8 {

    //Information about NXT brick LCD
    static int XCENTER = 50;
    static int YCENTER = 32;

    static Graphics gObj;

    /**
     * Draw a Radar Interface
     */
    private static void drawBackground(){
        gObj.clear();
        gObj.drawArc(50-10, 32-10, 20,20,0,360);
        gObj.drawArc(50-20, 32-20, 40,40,0,360);
        gObj.drawArc(50-30, 32-30, 60,60,0,360);

        //X Axis
        for(int i=15;i<=85;i++){
            gObj.setPixel(1,i,YCENTER);
        }

        //Y Axis
        for(int j=0;j<=64;j++){
            gObj.setPixel(1,XCENTER,j);
        }
        gObj.refresh();
    }

    /**
     * Radar Class Constructor.
     */
    public RADAR1_8(Graphics gObj2) throws Exception{
        gObj = gObj2;
    }
}
```

6.6.- Creating Textmenu for your GUI

In leJOS, exist a easy way to develop Textmenus. The technique is easy:

```
String[] UTMOptions = new String[]{"-3 UTM", "-2 UTM", "-1 UTM", " 0
UTM", "+1 UTM", "+2 UTM", "+3 UTM"};
TextMenu UTMSettings = new TextMenu(UTMOptions, 0, "UTM Settings");
int UTMOption = UTMSettings.select();
```

If you study the code, you will see that you define the options and define the menu. To show the menu in your logic you use the method select()

6.7.- Summary

In this chapter you learnt how to use leJOS features to use NXT Display.

7.- Communications

7.1.- Introduction

LeJOS project has a rich support for communications and it supports the following protocols:

1. System integrations
 - a. Bluetooth
 - b. USB
2. NXT integrations
 - a. RS485
 - b. I2C

7.1.1.- Communications for System integrations

If you need to develop a project which uses Bluetooth or USB then leJOS is your solution. LeJOS offers solutions to create software in your PC and NXT to exchange data. Bluetooth support is available if you need to exchange among NXT bricks.

7.1.2.- Communications for NXT integrations

In the other side, with leJOS exist the possibility to communicate 2 NXT brick using RS485. RS485 is the fastest way to exchange information between 2 NXT bricks.

Currently every new sensors use I2C to send and receive information so if the NXT provider write a nice documentation about I2C registers it is so easy to write a new class to incorporate to leJOS project

7.2.- Strategies

If you are going to develop a system to exchange data between 2 NXT or PC with a NXT for example, the best way to develop this area is using a Multithreading solutions.

7.3.- Summary

In this chapter and following chapters you will learn how to use Bluetooth, USB, RS485 & I2C

8.- Communications I: Bluetooth

8.1.- Introduction

Bluetooth technology has achieved global acceptance such that any Bluetooth enabled device, almost everywhere in the world, can connect to other Bluetooth enabled devices in proximity. Bluetooth enabled electronic devices connect and communicate wirelessly through short-range, ad hoc networks known as piconets. Each device can simultaneously communicate with up to seven other devices within a single piconet. Each device can also belong to several piconets simultaneously. Piconets are established dynamically and automatically as Bluetooth enabled devices enter and leave radio proximity.

Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec. The 2.4 GHz ISM band is available and unlicensed in most countries.

The operating range depends on the device class:

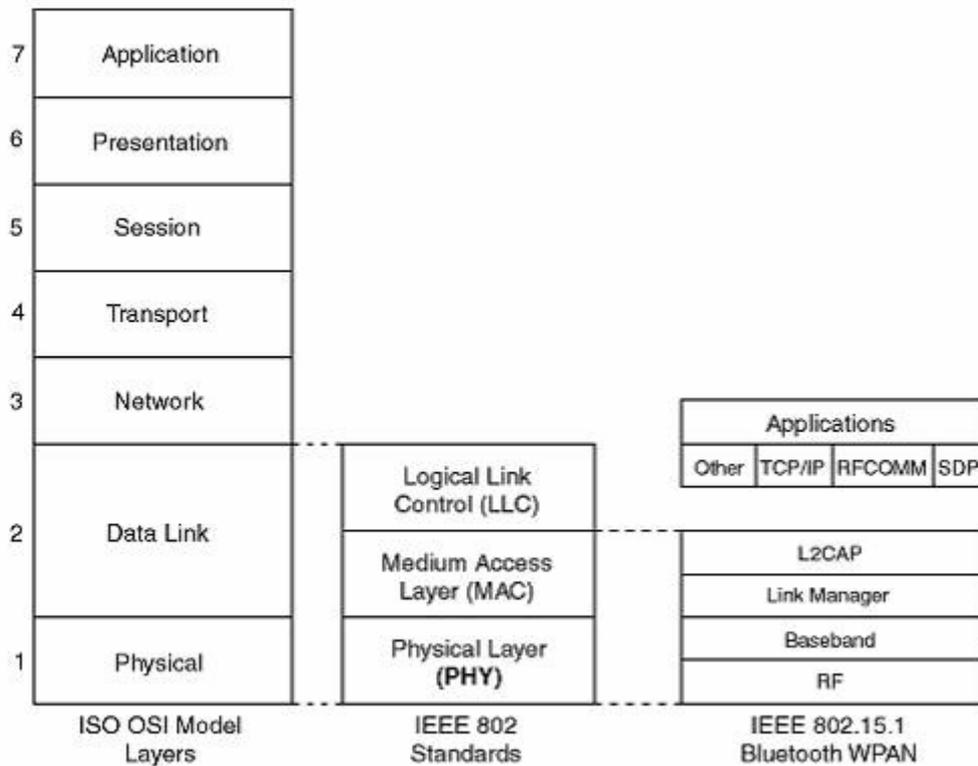
- Class 3 radios – have a range of up to 1 meter or 3 feet
- Class 2 radios – most commonly found in mobile devices – have a range of 10 meters or 33 feet
- Class 1 radios – used primarily in industrial use cases – have a range of 100 meters or 300 feet

8.2.- History

Originally invented in Scandinavia, the Bluetooth technology was named after the Danish Viking king Harold Bluetooth. However, when the technology was launched in 1998, it was very much an international initiative. A handful of leading companies within the computer and telecommunications industry formed the Bluetooth Special Interest Group (SIG). The goal was for devices from different manufacturers to be able to communicate with each other. Today, a great number of companies have joined the SIG as adopters of the Bluetooth technology, and the number is increasing all the time. The magnitude of industry involvement should ensure that Bluetooth becomes a widely adopted technology.

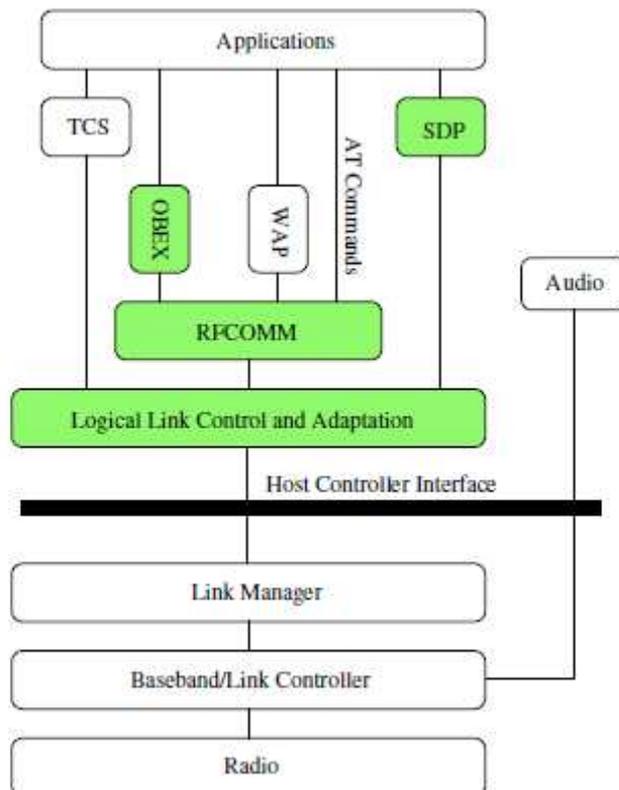
8.3.- Bluetooth Architecture

The Bluetooth architecture and its mapping to OSI model is shown below:



8.4.- Bluetooth Protocols

The Bluetooth protocols contain the standard procedures for connections and data exchange between Bluetooth devices.



Radio: Modulates and demodulates data for transmission and reception on air

Baseband/Link Controller: Controls the physical links via the radio, assembling packets and controlling frequency hopping

Link Manager: Responsible for security and link set-up between Bluetooth devices

Host Controller Interface (HCI): Handles communication between a separate host and a Bluetooth module (also referred to as the Bluetooth controller)

Logical Link Control and Adaptation (L2CAP): Multiplexes data from higher layers, converts between different packet sizes

RFCOMM: Emulates an RS-232 like serial interface

WAP and OBEX Adopted protocols

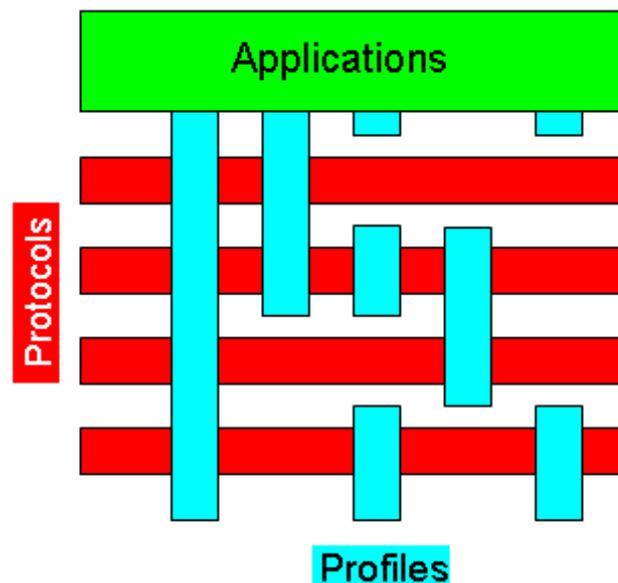
SDP (Service Discovery Protocol): Lets Bluetooth devices discover what services other Bluetooth devices support

OBEX: The Object Exchange Protocol (OBEX) is a specification for object data exchange.

TCS (Telephony Control Protocol Specification): Provides telephony services. It defines call control signaling for establishing speech and data calls between Bluetooth devices, providing them with telephony services.

8.5.- Bluetooth profiles

In order to use Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles. The profiles define the possible applications. Bluetooth profiles are general behaviors through which Bluetooth enabled devices communicate with other devices. Bluetooth technology defines a wide range of profiles that describe many different types of use cases.

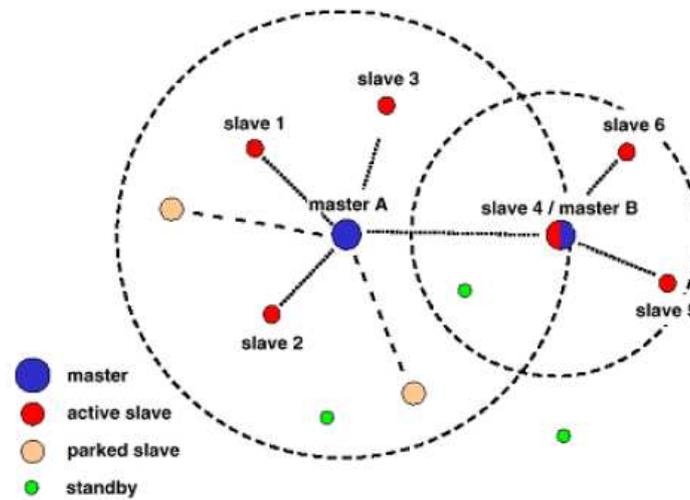


Devices do not need to implement all the profiles. A device only needs to implement the ones needed to support its applications. An exception however, is the Generic Access Profile, which is required in all devices.

8.6.- Bluetooth Networks

8.6.1.- Piconet

A piconet network is a group of Bluetooth devices joined together into a short range network by Bluetooth links. The group is synchronized to the timing and hopping sequence of the Master. Piconet is the heart of the Bluetooth technology. When a Bluetooth device has established a link to one or more other devices, a piconet has been formed. The device that initiates a connection acts as the master. The other devices are slaves. The master controls all traffic in the piconet. Communication between slaves can only take place via the master.



In any piconet network there can only be one master. Furthermore, up to seven slaves can be active. However, there can be additional slaves which are not active but remain synchronized to the piconet. Such slaves are referred to as parked. A parked device can very quickly become active and begin communicating in the piconet. By swapping active and parked slaves, you can increase the number of slaves virtually connected to the piconet from seven to 255 devices.

The master is the central entity that decides transmit/receive resource allocation to different slaves and thus controls bandwidth usage among slaves

The Slaves in a piconet only have links to the Master; there are no direct links between Slaves in a piconet:

- Master may become bottleneck for data transfer
- One piconet can be split into two piconets by one Slave becoming

A Slave have three power saving modes for idle devices. In order of power savings these modes are: PARK, HOLD, SNIFF.

The master can command slaves to go quite and then wake them up. In park these devices need only about 2 mW to stay operational. This makes them well suited for battery operations.

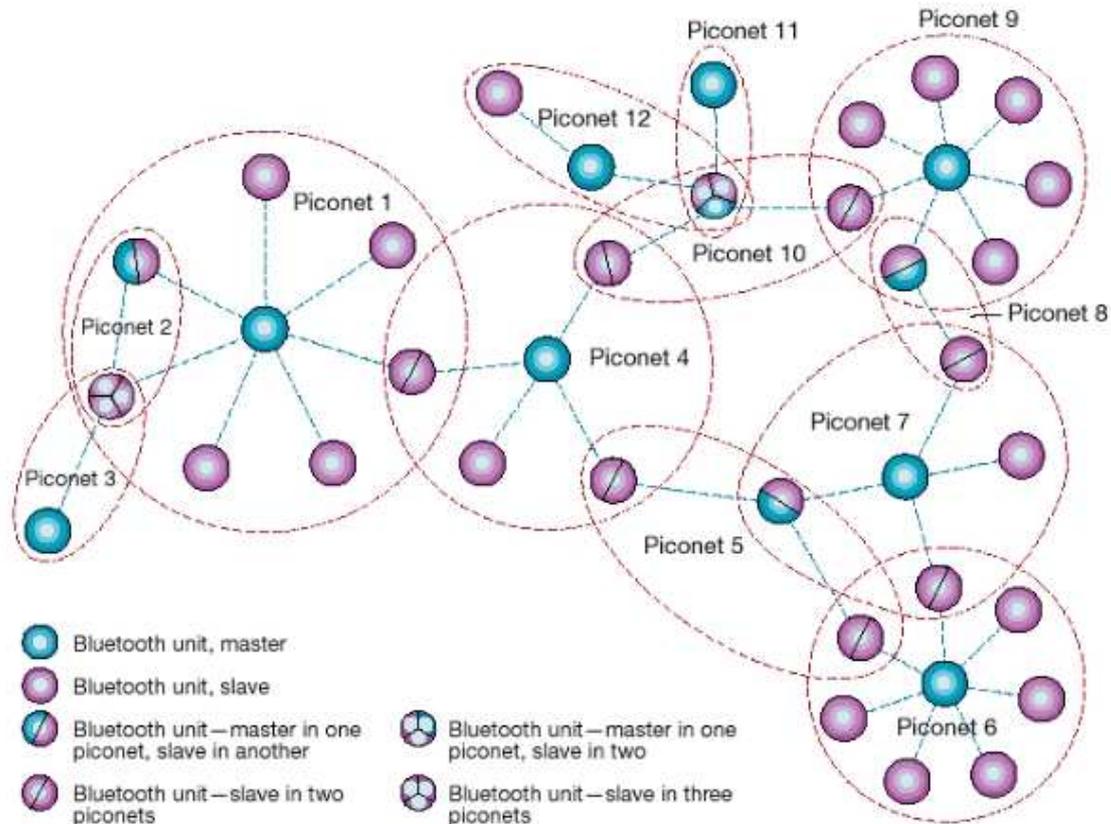
A Sniff mode, the slave listens within its Piconet at a reduced scan rate (programmable).

A Park mode, the slave is still synchronized within the network, but doesn't participate in data transmission. Devices in this mode only occasionally listen to maintain synchronization and for a wake up call. This is the maximum power saving mode.

A Hold mode, only an internal timer circuit keeps on working. Slaves can request the master to allow them to go into hold mode.

8.6.2.- Scatternet

A Scatternet is a group of Bluetooth piconets joined together by devices that are in more than one piconet.



Scatternet network consist on two piconets networks with a different power class devices.

8.7.- Bluetooth Connections

Two types of connections are possible: Synchronous Connection Orientated (SCO), or Asynchronous Connection Less (ACL).

The SCO (Synchronous Connection Orientated) channels are used for voice transfer by reserving slots, they are symmetric between the master and a slave. There is a maximum of three in a piconet, a slave being able to control two originating from different masters. One can use the voice packets or the mixed voice/data packets.

An ACL (Asynchronous Connection Less) channel supplies an asynchronous access between master and slave (a single channel per couple), with the slot as base. The data packets only are used. In addition, a slave can only transmit after having received a packet from the master (the following slot). For this purpose, the master can send polling packets to the slaves (when there is nothing more asynchronous to transmit).

8.8.- How to use Bluetooth with leJOS

Once you have understood Bluetooth technology, this document explains how to use LeJOS technology to manage Bluetooth connections and exchange data with others Bluetooth Device.

LeJOS packages which you have to use when you have to manage Bluetooth communications are:

- java.io.*
- Javax.bluetooth.*
- lejos.nxt.comm.*
- lejos.nxt.remote.*
- lejos.devices.*

The steps to connect and manage a Bluetooth device are the following:

1. Discovery Bluetooth Device
2. Connect with a Bluetooth device
3. Exchange Data between a NXT Brick with another one or a Bluetooth device
4. Listen a Bluetooth connection

8.8.1.- Discovery Bluetooth Device

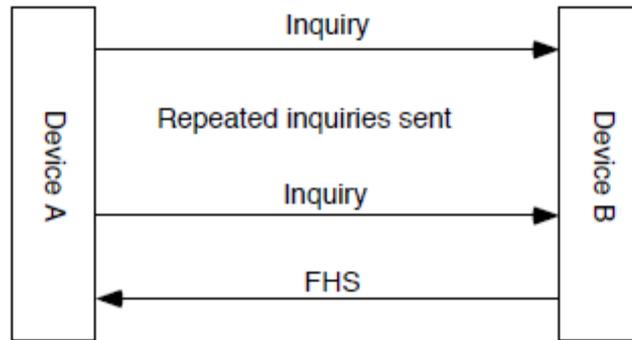
If you want to connect with another NXT brick or another Bluetooth Device, it is necessary to know 2 things:

1. What device do you want to connect?
2. Is available that device to connect?

NXT Brick has a memory where store information about Bluetooth devices. You can use the static method: **Bluetooth.getKnownDevicesList()**; to extract values from NXT brick.

```
//Extract data from NXT brick
Vector curList = Bluetooth.getKnownDevicesList();
if (curList == null)
{
    RConsole.println("getKnownDeviceList returns null\n");
    return false;
}
else{
    //Show remote devices stored on NXT brick
    for(int i = 0; i < curList.size(); i++){
        RConsole.println("Current device " +
            ((RemoteDevice)curList.elementAt(i)).getFriendlyName(false) + "\n");
    }
}
```

In case of your NXT brick doesn't have any data about your environment, it is necessary to make an inquire using **Bluetooth.inquire()**



This piece of code, explains the concept:

```

byte[] cod = {0,0,0,0}; // Any
RConsole.println("Searching...\n");
Vector devList = Bluetooth.inquire(5, 10,cod);
if (devList == null)
{
    RConsole.println("Inquire returns null\n");
    return false;
}
if (devList.size() > 0) {
    String[] names = new String[devList.size()];
    for (int i = 0; i < devList.size(); i++) {
        RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
        names[i] = btrd.getFriendlyName(false);
        RConsole.println("Got device " + names[i] + "\n");
    }
}
  
```

Once you discover others Bluetooth devices, it is necessary to add into NXT brick:

```

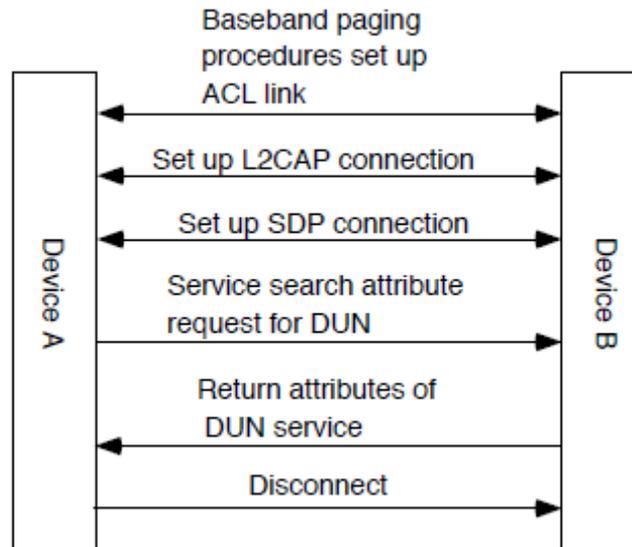
for(int i = 0; i < devList.size(); i++){
    RemoteDevice btrd = ((RemoteDevice) devList.elementAt(i));
    Bluetooth.addDevice(btrd);
}
  
```

Conclusions

See your list about known devices then if you don't have any device then search your environment to detect new devices. Using inquire methods. Remember that if you want to connect with them, then you need to add into your list.

8.8.2.- Connect with a Bluetooth device

Once NXT brick has added NXT brick or Bluetooth device, the connection it is very easy.



The first step is create a **RemoteDevice** object using the name of the target:

```

RemoteDevice btrd = Bluetooth.getKnownDevice("NXT");
if (btrd == null) {
    return false;
}
  
```

Once you have a **RemoteDevice** Object then connect with it:

```

BTConnection btc = Bluetooth.connect(btrd);
if (btc == null) {
    return false;
}
byte [] status = Bluetooth.getConnectionStatus();
if (status == null)
{
    btc.close();
    return false;
}
  
```

If the Bluetooth device has a pin then when have to connect, indicate it:

```

final byte[] pin = {(byte) '0', (byte) '0', (byte) '0', (byte)
'0'};
BTConnection btGPS = null;
btGPS = Bluetooth.connect(btrd.getDeviceAddr(),
NXTConnection.RAW, pin);
  
```

8.8.3.- Exchange Data between a NXT Brick with another one or a Bluetooth device

To exchange data between a NXT Device with another, exist 2 ways:

- Using setIOMode
- Using DataInputStream

The first way **setIOMode** to **NXTConnection.RAW**. This piece of code explain the idea:

```

btc.setIOMode(NXTConnection.RAW);
for(int i = 0; i < 100; i++)
{
    byte [] b = strToByte("Hello world " + i + "\r\n");
    if (btc != null) btc.write(b, b.length);
    try{Thread.sleep(delay);}catch(Exception e){}
}
if (btc != null) btc.close();

```

Another example using this way:

```

btc.setIOMode(NXTConnection.RAW);
byte []inBuf = new byte[255];
while (true)
{
    int cnt;
    if (userp)
        cnt = bt.readPacket(inBuf, inBuf.length);
    else
        cnt = bt.read(inBuf, inBuf.length);
    if (cnt == 0) continue;
    if (cnt == -2)
    {
        RConsole.println("Lost data, resync\n");
        bt.read(null, 256);
        continue;
    }
    int val = ((int)inBuf[2] & 0xff) + (((int)inBuf[3] & 0xff)
<< 8);
    RConsole.println("Read len " + cnt + " val " + val + "\n");
    if (val == 0xffff) break;
    byte [] b = strToByte("Hello from bt " + val + "\r\n");
    if (btc != null) btc.write(b, b.length);
}
if (btc != null) btc.close();

```

The second way to manage a Bluetooth connection, is the following:

Understand this piece of code:

```

InputStream in = null;
in = btGPS.openInputStream();
try {
    in.read(segment);
} catch (IOException e) {
    // How to handle error?
}

```

This second example is used for bidirectional communications:

```

DataInputStream dis = btc.openDataInputStream();
DataOutputStream dos = btc.openDataOutputStream();

for(int i=0;i<100;i++) {
    try {
        LCD.drawInt(i*30000, 8, 0, 2);
        LCD.refresh();
    }
}

```

```

        dos.writeInt(i*30000);
        dos.flush();
    } catch (IOException ioe) {
        LCD.drawString("Write Exception", 0, 0);
        LCD.refresh();
    }

    try {
        LCD.drawInt(dis.readInt(),8, 0,3);
        LCD.refresh();
    } catch (IOException ioe) {
        LCD.drawString("Read Exception ", 0, 0);
        LCD.refresh();
    }
}

try {
    LCD.drawString("Closing... ", 0, 0);
    LCD.refresh();
    dis.close();
    dos.close();
    btc.close();
} catch (IOException ioe) {
    LCD.drawString("Close Exception", 0, 0);
    LCD.refresh();
}
}

```

8.8.4.- Listen a Bluetooth connection

If your goal is connect with another NXT brick then the second NXT brick should listen a Bluetooth connection to exchange data or receive commands.

To listen a Bluetooth connection use: **Bluetooth.waitForConnection()**

```
BluetoothConnection btc = Bluetooth.waitForConnection();
```

Once you have connected with a NXT brick sender then exchange data using this way:

```

DataStream dis = btc.openDataInputStream();
DataStream dos = btc.openDataOutputStream();

for(int i=0;i<100;i++) {
    int n = dis.readInt();
    LCD.drawInt(n,7,0,1);
    LCD.refresh();
    dos.writeInt(-n);
    dos.flush();
}

dis.close();
dos.close();
Thread.sleep(100); // wait for data to drain
LCD.clear();
LCD.drawString(closing,0,0);
LCD.refresh();
btc.close();

```

8.9.- LeJOS examples using Bluetooth

The following examples have been extracted from current leJOS NXJ release.

- Bluetooth Concepts
 - BTConnectTest.java
 - BTReceive.java
 - Bttest.java
 - Bluestats.java
 - SignalTest.java
 - StartUpText.java
- Connection with a GPS
 - BTGPS.java
 - GPS.java
- Connection with a BT Keyboard
 - KeyboardTest.java
 - Keyboard.java
 - KeyListener.java
 - KeyEvent.java

This source code is very useful to understand the concepts about Bluetooth technology and leJOS.

8.10.- Summary

In this chapter you learnt how to use Bluetooth protocol with leJOS.

9.- Communications II: USB

9.1.- Introduction

LeJOS support USB to send and receive data from a PC to NXT. In this chapter you will learn how to use USB.

9.2.- Send data from PC to NXT

```
public class USBSend {
    public static void main(String[] args) {
        NXTConnector conn = new NXTConnector();
        if (!conn.connectTo("usb://")){
            System.err.println("No NXT find using USB");
            System.exit(1);
        }

        DataInputStream inDat = conn.getDataIn();
        DataOutputStream outDat = conn.getDataOut();

        int x = 0;
        for(int i=0;i<100;i++)
        {
            try {
                outDat.writeInt(i);
                outDat.flush();

            } catch (IOException ioe) {
                System.err.println("IO Exception writing
bytes");
            }

            try {
                x = inDat.readInt();
            } catch (IOException ioe) {
                System.err.println("IO Exception reading reply");
            }
            System.out.println("Sent " + i + " Received " + x);
        }

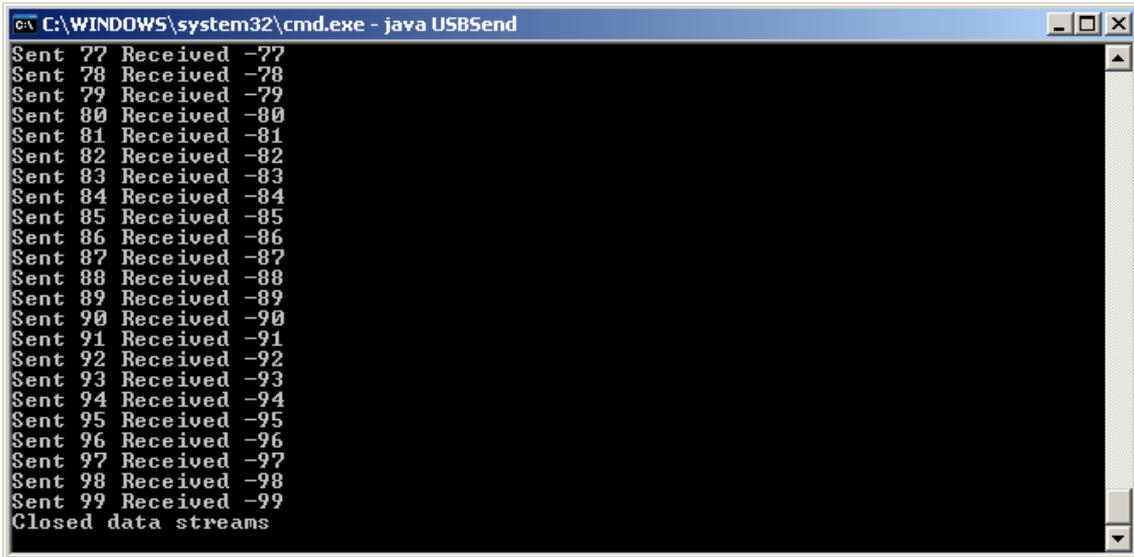
        try {
            inDat.close();
            outDat.close();
            System.out.println("Closed data streams");
        } catch (IOException ioe) {
            System.err.println("IO Exception Closing
connection");
        }

        try {
            conn.close();
            System.out.println("Closed connection");
        } catch (IOException ioe) {
            System.err.println("IO Exception Closing
connection");
        }
    }
}
```

```

    }
}
}

```



```

C:\WINDOWS\system32\cmd.exe - java USBSend
Sent 77 Received -77
Sent 78 Received -78
Sent 79 Received -79
Sent 80 Received -80
Sent 81 Received -81
Sent 82 Received -82
Sent 83 Received -83
Sent 84 Received -84
Sent 85 Received -85
Sent 86 Received -86
Sent 87 Received -87
Sent 88 Received -88
Sent 89 Received -89
Sent 90 Received -90
Sent 91 Received -91
Sent 92 Received -92
Sent 93 Received -93
Sent 94 Received -94
Sent 95 Received -95
Sent 96 Received -96
Sent 97 Received -97
Sent 98 Received -98
Sent 99 Received -99
Closed data streams

```

9.3.- Receive data from PC

This example show how to use a USB Connection to receive data from a PC.

```

import lejos.nxt.*;
import java.io.*;
import lejos.nxt.comm.*;

public class USBReceive {

    public static void main(String [] args) throws Exception
    {
        LCD.drawString("waiting", 0, 0);
        USBConnection conn = USB.waitForConnection();
        DataOutputStream dOut = conn.openDataOutputStream();
        DataInputStream dIn = conn.openDataInputStream();

        while (true)
        {
            int b;
            try
            {
                b = dIn.readInt();
            }
            catch (EOFException e)
            {
                break;
            }

            dOut.writeInt(-b);
            dOut.flush();
            LCD.drawInt((int)b,8,0,1);
        }
    }
}

```

9.4.- Summary

In this chapter, you learnt how to use USB in your leJOS projects.

10.- Communications III: RS485

10.1.- Introduction

NXTBrick has a surprise in the input port 4. The port4 has the support for RS485. LeJOS project has implemented Bitbus protocol for RS485. To use this feature, connect 2 NXT brick with the same wire plugging the wire in the port4.

The way to use RS485 is similar to others protocols:

```
String name = "NXT";
NXTCommConnector[] connectors = {Bluetooth.getConnector(),
RS485.getConnector()};
int[] modes = {NXTConnection.PACKET, NXTConnection.RAW};
NXTConnection con = connectors[connectionType].connect(name,
modes[mode]);
NXTConnection con = RS485.getConnector().connect(name,modes[0]);
DataInputStream dis = con.openDataInputStream();
DataOutputStream dos = con.openDataOutputStream();
```

In the following examples you will learn how to send and receive data with RS485.

10.2.- Send data with RS485

```
import lejos.nxt.*;
import lejos.nxt.comm.*;
import java.io.*;

public class NXTConnectTest
{
    public static void main(String[] args) throws Exception
    {
        String name = "NXT";
        String[] connectionStrings = {"Bluetooth", "RS485"};
        TextMenu connectionMenu = new TextMenu(connectionStrings, 0,
"Connection");
        String[] modeStrings = {"Packet", "Raw"};
        TextMenu modeMenu = new TextMenu(modeStrings, 0, "Mode");
        NXTCommConnector[] connectors = {Bluetooth.getConnector(),
RS485.getConnector()};
        int[] modes = {NXTConnection.PACKET, NXTConnection.RAW};

        int connectionType = connectionMenu.select();
        LCD.clear();
        int mode = modeMenu.select();

        LCD.clear();
        LCD.drawString("Name: " + name, 0, 0);
        LCD.drawString("Type: " + connectionStrings[connectionType],
0, 1);
        LCD.drawString("Mode: " + modeStrings[mode], 0, 2);
        LCD.drawString("Connecting...", 0, 3);
```

```

    NXTConnection con = connectors[connectionType].connect(name,
modes[mode]);

    if (con == null)
    {
        LCD.drawString("Connect fail", 0, 5);
        Thread.sleep(2000);
        System.exit(1);
    }

    LCD.drawString("Connected          ", 0, 3);
    LCD.refresh();

    DataInputStream dis = con.openDataInputStream();
    DataOutputStream dos = con.openDataOutputStream();

    for (int i = 0; i < 100; i++)
    {
        try
        {
            LCD.drawString("write: ", 0, 6);
            LCD.drawInt(i * 30000, 8, 6, 6);
            dos.writeInt(i * 30000);
            dos.flush();
        }
        catch (IOException ioe)
        {
            LCD.drawString("Write Exception", 0, 5);
        }
        try
        {
            LCD.drawString("Read: ", 0, 7);
            LCD.drawInt(dis.readInt(), 8, 6, 7);
        }
        catch (IOException ioe)
        {
            LCD.drawString("Read Exception ", 0, 5);
        }
    }

    try
    {
        LCD.drawString("Closing...      ", 0, 3);
        dis.close();
        dos.close();
        con.close();
    }
    catch (IOException ioe)
    {
        LCD.drawString("Close Exception", 0, 5);
        LCD.refresh();
    }
    LCD.drawString("Finished          ", 0, 3);
    Thread.sleep(2000);
}
}

```

10.3.- Receive data with RS485

```

import lejos.nxt.*;
import lejos.nxt.comm.*;
import java.io.*;

public class NXTReceive
{
    public static void main(String[] args) throws Exception
    {
        String[] connectionStrings = new String[]{"Bluetooth", "USB",
"RS485"};
        TextMenu connectionMenu = new TextMenu(connectionStrings, 0,
"Connection");
        String[] modeStrings = new String[] {"Packet", "Raw"};
        TextMenu modeMenu = new TextMenu(modeStrings, 0, "Mode");
        NXTCommConnector[] connectors = {Bluetooth.getConnector(),
USB.getConnector(), RS485.getConnector()};
        int[] modes = {NXTConnection.PACKET, NXTConnection.RAW};

        int connectionType = connectionMenu.select();
        LCD.clear();
        int mode = modeMenu.select();
        while (true)
        {
            LCD.clear();
            LCD.drawString("Type: " + connectionStrings[connectionType], 0, 0);
            LCD.drawString("Mode: " + modeStrings[mode], 0, 1);
            LCD.drawString("Waiting...", 0, 2);

            NXTConnection con =
connectors[connectionType].waitForConnection(0, modes[mode]);

            LCD.drawString("Connected...", 0, 2);

            DataInputStream dis = con.openDataInputStream();
            DataOutputStream dos = con.openDataOutputStream();

            while(true)
            {
                int n;
                try{
                    n = dis.readInt();
                } catch (EOFException e)
                {
                    break;
                }
                LCD.drawString("Read: ", 0, 4);
                LCD.drawInt(n, 7, 6, 4);
                dos.writeInt(-n);
                dos.flush();
            }

            LCD.drawString("Closing... ", 0, 2);
            dis.close();
            dos.close();
            con.close();
        }
    }
}

```

10.4.- Summary

In this chapter, you learnt how to use RS485 in your leJOS projects.

11.- Communications IV: I2C

11.1.- Introduction

I2C (Inter-Integrated Circuit) is a multi-master serial computer bus invented by Philips that is used to attach low-speed peripherals to a motherboard, embedded system, or cellphone.

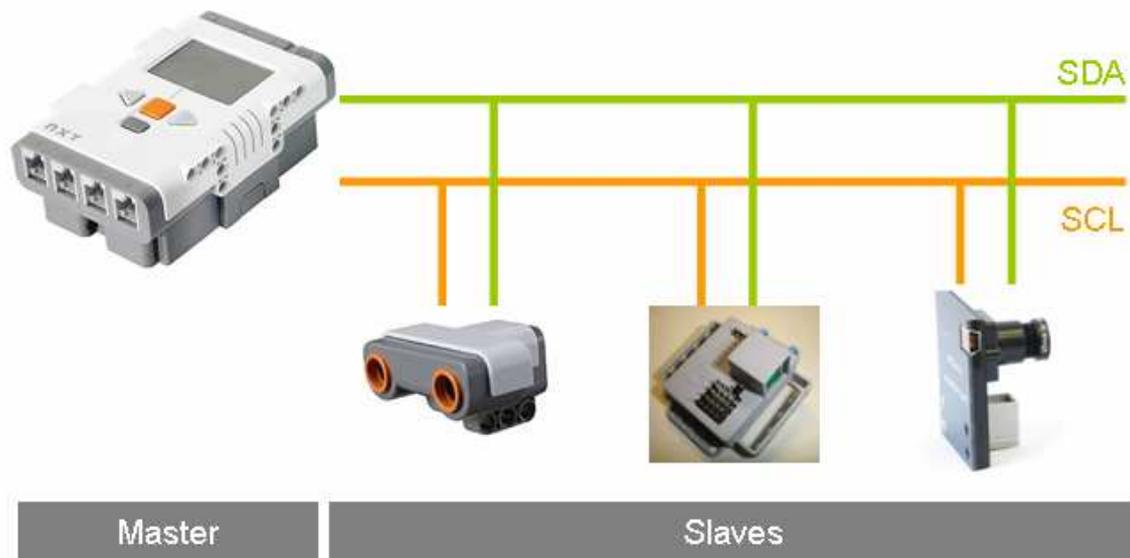
I2C uses only two bidirectional open-drain lines, Serial Data (SDA) and Serial Clock (SCL), pulled up with resistors.

Every device hooked up to the bus has its own unique address.

The bus has two roles for nodes: master and slave:

- Master node: node that issues the clock and addresses slaves
- Slave node: node that receives the clock line and address.

In NXT world the I2C Diagrama should be the following:



There are four potential modes of operation for a given bus device, although most devices only use a single role and its two modes:

- **Master transmit:** master node is sending data to a slave
- **Master receive:** master node is receiving data from a slave
- **Slave transmit:** slave node is sending data to a master
- **Slave receive:** slave node is receiving data from the master

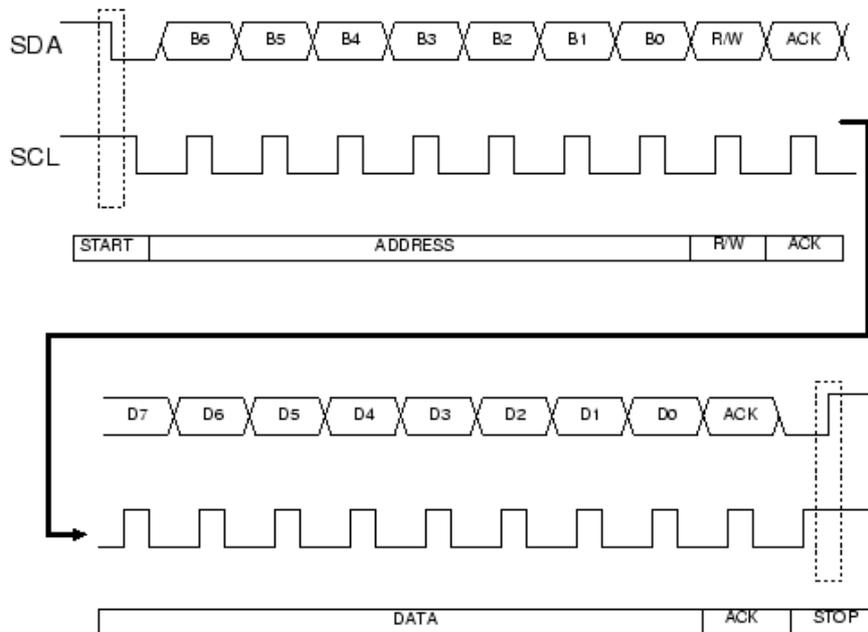
11.2.- I2C Bus terminology

The terminology used when you work with I2C is the following:

- **Transmitter:** The device that sends data to the bus. A transmitter can either be a device that puts data on the bus on its own accord (a 'master-transmitter'), or in response to a request from the master (a 'slave-transmitter').
- **Receiver:** the device that receives data from the bus. A receiver can either be a device that receives data on its own request (a 'master-receiver'), or in response to a request from the master (a 'slave-receiver').
- **Master:** the component that initializes a transfer (Start command), generates the clock (SCL) signal and terminates the transfer (Stop command). A master can be either a transmitter or a receiver.
- **Slave:** the device addressed by the master. A slave can be either receiver or transmitter.
- **Multi-master:** the ability for more than one master to co-exist on the bus at the same time without collision or data loss. Typically "bit-banged" software implemented masters are not multi-master capable. Parallel to I²C bus controllers provide an easy way to add a multi-master hardware I²C port to DSPs and ASICs.
- **Arbitration:** the prearranged procedure that authorizes only one master at a time to take control of the bus.
- Synchronization - the prearranged procedure that synchronizes the clock signals provided by two or more masters.
- **SDA:** data signal line (Serial DAta)
- **SCL:** clock signal line (Serial CLock)

11.3.- Terminology for bus transfer

- **F (FREE):** the bus is free or idle; the data line SDA and the SCL clock are both in the high state.
- **S (START) or R (RESTART):** data transfer begins with a Start condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes 'busy'.
- **C (CHANGE):** while the SCL clock line is low, the data bit to be transferred can be applied to the SDA data line by a transmitter. During this time, SDA may change its state, as long as the SCL line remains low.
- **D (DATA):** a high or low bit of information on the SDA data line is valid during the high level of the SCL clock line. This level must be kept stable during the entire time that the clock remains high to avoid misinterpretation as a Start or Stop condition.
- **P (STOP):** data transfer is terminated by a Stop condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free once again.



11.4.- LeJOS API

LeJOS project supports I2C devices connected to NXT brick. Every object which uses I2C protocol inherits from **I2CSensor**

The NXT devices which use I2C are:

- ColorSensor
- CompassSensor
- IRSeeker
- NXTe
- NXTCam
- OpticalDistanceSensor
- PSPNXController
- RCXLink
- RCXMotorMultiplexer
- RCXSensorMultiplexer
- TiltSensor
- UltrasonicSensor

The class I2CSensor has the following I2C methods:

- setAddress
- sendData
- getData

Further information about leJOS API here:

<http://lejos.sourceforge.net/nxt/nxj/api/index.html>

When it is necessary to write a class to manage a new leJOS device the question to do are:

- What is the I2C address to write and read data?
- What is the list of I2C registers to write and read data?
- How to interpret the values from I2C registers?

11.5.- I2C Examples with leJOS

To explain the concepts, I will use a NXT I2C Device, Mindsensors NXTServo. This device has been developed to manage RC Servos.

If we want to read the battery connected to that device, it is necessary to know the following parameters:

1. NXTServo I2C Address: **0xb0**
2. NXTServo I2C Register to read battery level: **0x41**

Now I will write a simple example which read the battery from NXTServo:

```
public class NXTServoTest{

    public static void main(String[] args){
        DebugMessages dm = new DebugMessages();
        dm.setLCDLines(6);
        dm.echo("Testing NXT Servo");

        MSC msc = new MSC(SensorPort.S1);
        msc.addServo(1,"Mindsensors RC Servo 9Gr");

        while(!Button.ESCAPE.isPressed()){
            dm.echo(msc.getBattery());
        }
        dm.echo("Test finished");
    }
}
```

The class MSC, Mindsensors Servo Controller, manages until 8 RC Servos. I will show 2 internal methods in the class MSC:

The constructor:

```
public static final byte NXTSERVO_ADDRESS = (byte)0xb0;

public MSC(SensorPort port){
    super(port);
    port.setType(TYPE_LOWSPEED_9V);
    this.setAddress(MSC.NXTSERVO_ADDRESS);

    this.portConnected = port;
    arrServo = new ArrayList();
}
```

If you observe the code, all I2C operation will use the address 0xb0

The method getBattery:

```
public int getBattery(){
    int I2C_Response;
    byte[] bufReadResponse;
    bufReadResponse = new byte[8];
    byte kSc8_Vbatt = 0x41;//The I2C Register to read the battery
    I2C_Response = this.getData(kSc8_Vbatt, bufReadResponse, 1);
}
```

```

return(37*(0x00FF & bufReadResponse[0])); // 37 is calculated
from
    //supply from NXT =4700 mv /128
}

```

In this example we read the I2C register 0x41 which store battery level. Every I2C action has a response. If the response is 0 then it is a success if the result is not 0 then it was a failure. Besides when you read a I2C register, you have to use a buffer, in this case bufReadResponse.

11.6.- Migrating code I2C from others platforms

When you develop NXT software, it is a usual that you get ideas from others developers who likes others platforms. In this section I will explain how to migrate I2C RobotC and NXC code

11.6.1.- Migrating I2C Code from RobotC to Java leJOS

RobotC has the following I2C functions to read and write registers.

Function	Description
sendI2CMsg(nPort, sendMsg, nReplySize);	Send an I2C message on the specified sensor port.
nI2CBytesReady[]	This array contains the number of bytes available from a I2C read on the specified sensor port.
readI2CReply(nPort, replyBytes, nBytesToRead);	Retrieve the reply bytes from an I2C message.
nI2CStatus[]	Currents status of the selected sensor I2C link.
nI2CRetries	This variable allows changing the number of message retries. The default action tries to send every I2C message three times before giving up and reporting an error. Unfortunately, this many retries can easily mask any faults that can exist.
SensorType[]	This array is used to configure a sensor for I2C operation. It also indicates whether 'standard' or 'fast' transmission should be used with this sensor.

Now I will show an example with the same method getBattery:

```

/*=====
**
** Read the battery voltage data from
** NXTServo module (in mili-volts)
**
=====*/
int Get_Batt_V()
{
    byte sc8Msg[5];
    const int kMsgSize      = 0;
    const int kSc8Address   = 1;
    const int kReadAddress  = 2;
    byte replyMsg[2];

    // Build the I2C message

```

```

sc8Msg[kMsgSize]          = 2;
sc8Msg[kSc8Address]      = kSc8ID ;
sc8Msg[kReadAddress]     = kSc8_Vbatt ;

while (nI2CStatus[kSc8Port] == STAT_COMM_PENDING);
{
// Wait for I2C bus to be ready
}
// when the I2C bus is ready, send the message you built
sendI2CMsg(kSc8Port, sc8Msg[0], 1);

while (nI2CStatus[kSc8Port] == STAT_COMM_PENDING);
{
// Wait for I2C bus to be ready
}
// when the I2C bus is ready, send the message you built
readI2CReply(kSc8Port, replyMsg[0], 1);

return(37*(0x00FF & replyMsg[0])); // 37 is calculated from

//supply from NXT =4700 mv /128
}

```

Now the migration to Java leJOS:

```

/**
 * Read the battery voltage data from
 * NXTServo module (in mili-volts)
 *
 * @return
 */
public int getBattery(){
    int I2C_Response;
    byte[] bufReadResponse;
    bufReadResponse = new byte[8];
    byte kSc8_Vbatt = 0x41;//The I2C Register to read the battery
    I2C_Response = this.getData(kSc8_Vbatt, bufReadResponse, 1);
    return(37*(0x00FF & bufReadResponse[0]));// 37 is calculated
from
    //supply from NXT =4700 mv /128
}

```

11.6.2.- Migrating I2C Code from NXC to Java leJOS

NXC Programming Language has a set of functions to manage I2C:

Function	Description
LowspeedWrite(port, returnlen, buffer)	This method starts a transaction to write the bytes contained in the array buffer to the I2C device on the specified port. It also tells the I2C device the number of bytes that should be included in the response. The maximum number of bytes that can be written or read is 16. The port may be specified using a constant (e.g., IN_1, IN_2, IN_3, or IN_4) or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.

	<pre>x = LowspeedWrite(IN_1, 1, inbuffer);</pre>
LowspeedStatus(port, out bytesready)	<p>This method checks the status of the I2C communication on the specified port. If the last operation on this port was a successful LowspeedWrite call that requested response data from the device then bytesready will be set to the number of bytes in the internal read buffer. The port may be specified using a constant (e.g., IN_1, IN_2, IN_3, or IN_4) or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.</p> <p>If the return value is 0 then the last operation did not cause any errors. Avoid calls to LowspeedRead or LowspeedWrite while LowspeedStatus returns STAT_COMM_PENDING.</p> <pre>x = LowspeedStatus(IN_1, nRead);</pre>
LowspeedCheckStatus(port)	<p>This method checks the status of the I2C communication on the specified port. The port may be specified using a constant (e.g., IN_1, IN_2, IN_3, or IN_4) or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads. If the return value is 0 then the last operation did not cause any errors. Avoid calls to LowspeedRead or LowspeedWrite while LowspeedStatus returns STAT_COMM_PENDING.</p> <pre>x = LowspeedCheckStatus(IN_1);</pre>
LowspeedBytesReady(port)	<p>This method checks the status of the I2C communication on the specified port. If the last operation on this port was a successful LowspeedWrite call that requested response data from the device then the return value will be the number of bytes in the internal read buffer. The port may be specified using a constant (e.g., IN_1, IN_2, IN_3, or IN_4) or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads.</p> <pre>x = LowspeedBytesReady(IN_1);</pre>
LowspeedRead(port, buflen, out buffer)	<p>Read the specified number of bytes from the I2C device on the specified port and store the bytes read in the array buffer provided. The maximum number of bytes that can be written or read is 16. The port may be specified using a constant (e.g., IN_1, IN_2, IN_3, or IN_4) or a variable. Constants should be used where possible to avoid blocking access to I2C devices on other ports by code running on other threads. If the return value is negative then the output buffer will be empty.</p>

	<code>x = LowspeedRead(IN_1, 1, outbuffer);</code>
<code>I2CWrite(port, returnlen, buffer)</code>	This is an alias for <code>LowspeedWrite</code> . <code>x = I2CWrite(IN_1, 1, inbuffer);</code>
<code>I2CStatus(port, out bytesready)</code>	This is an alias for <code>LowspeedStatus</code> . <code>x = I2CStatus(IN_1, nRead);</code>
<code>I2CCheckStatus(port)</code>	This is an alias for <code>LowspeedCheckStatus</code> . <code>x = I2CCheckStatus(IN_1);</code>
<code>I2CBytesReady(port)</code>	This is an alias for <code>LowspeedBytesReady</code> . <code>x = I2CBytesReady(IN_1);</code>
<code>I2CRead(port, buflen, out buffer)</code>	This is an alias for <code>LowspeedRead</code> . <code>x = I2CRead(IN_1, 1, outbuffer);</code>
<code>I2CBytes(port, inbuf, in/out count, out outbuf)</code>	This method writes the bytes contained in the input buffer (inbuf) to the I2C device on the specified port, checks for the specified number of bytes to be ready for reading, and then tries to read the specified number (count) of bytes from the I2C device into the output buffer (outbuf). The port may be specified using a constant (e.g., <code>IN_1</code> , <code>IN_2</code> , <code>IN_3</code> , or <code>IN_4</code>) or a variable. Returns true or false indicating whether the I2C read process succeeded or failed. This is a higher-level wrapper around the three main I2C functions. It also maintains a "last good read" buffer and returns values from that buffer if the I2C communication transaction fails. <code>x = I2CBytes(IN_4, writebuf, cnt, readbuf);</code>

To explain the concepts, will show an example from Lattebox NXTe which has leJOS and NXC support:

```
byte bufConfigureSPI[] = {0x50, 0xF0, 0x0C};

void LowspeedWait()
{
  while(true){
    if (LowspeedCheckStatus(IN_3) == NO_ERR) break;
  }
}

void nxt_init()
{
  ResetSensor(IN_3);
  Wait(100);
  SetSensorType(IN_3, IN_TYPE_LOWSPEED);
  SetSensorMode(IN_3, IN_MODE_RAW);
  ResetSensor(IN_3);
  Wait(100);
}
```

```

        LowSpeedWait();
        LowSpeedWrite(IN_3,0,bufConfigureSPI);
        LowSpeedWait();
    }

    public static final byte NXTE_ADDRESS = 0x28;
    private final byte REGISTER_IIC = (byte)0xF0;//NXTe IIC address

    /**
     * Constructor
     *
     * @param port
     */
    public NXTe(SensorPort port){
        super(port);

        port.setType(TYPE_LOWSPEED_9V);
        port.setMode(MODE_RAW);

        portConnected = port;

        arrLSC = new ArrayList();

        this.setAddress((int) NXTE_ADDRESS);
        int I2C_Response;
        I2C_Response = this.sendData((int)this.REGISTER_IIC,
        (byte)0x0c);
    }

```

11.7.- Summary

In this chapter, you learnt how to use I2C protocol to develop new leJOS classes to support new sensors or actuators.

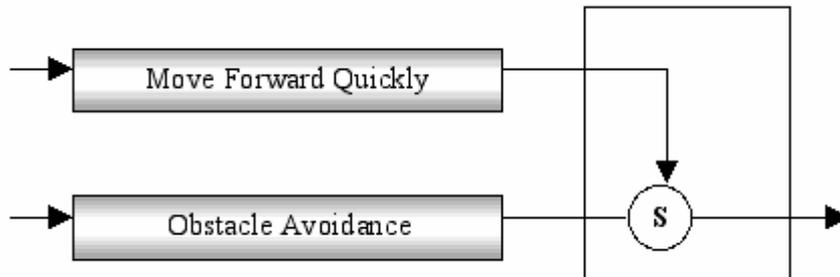
12.- Subsumption architecture

12.1.- 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.



12.2.- 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

12.3.- 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, BumperCar 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.

13.- Multithreading with Java leJOS

13.1.- Introduction

Multithreading is a Java feature which allow executing multiple tasks at the same time. If you develop robots, you should consider this programming feature as the basis of your architecture.

Normally when you begin to develop with Java and leJOS you develop programs which execute the operations in sequence:

```
Program:
  Task1
  Task2
  Task3
  ...
  Taskn
```

But if you use Java Multithreading then you could execute robot's tasks in parallel if it is necessary.

```
Program:
  Task1      Task2
  Task3
```

If you notice, the human body executes several operations in parallel for example: Respiration, blood circulation, digestion, thinking and walking, besides the human brain process many data from our senses in parallel: sight, touch, smell, taste and hearing.

Most programming languages do not enable programmers to specify concurrent activities. Rather, the languages generally provide control statements that only enable programmers to perform one action at a time, proceeding to the next action after the previous one has finished. Historically, concurrency has been implemented with operating system primitives available only to experienced systems programmers.

13.2.- The Thread concept

A thread in Java is the minimum process's unit in parallelism terms. When you design a robot using Threads, you could think in the following ideas:

1. A thread to manage the locomotion subsystem (For a wheeled robot, biped robot, hexapod robot, etc...)
2. A thread to manage GPS
3. A thread to manage Bluetooth communications
4. A thread to manage the robot arm
5. A thread to manage the robot's senses (Ultrasonic sensors for example)

The robot's design process is iterative, because it is a complex task, but it is so useful because when in the future you want to incorporate a new feature then add a new task, a thread, or select the old thread and update this one with new capabilities.

In the following example, we will develop our first Thread which is managed by a main program.

HelloWorldThread.java

This is the first example using Threads. This thread prints in NXT LCD the message "Hello World" and the value of the variable i.

```
import lejos.nxt.*;

public class HelloWorldThread extends Thread{
    private int i = 0;

    public HelloWorldThread(){
    }

    public void run(){
        while(true){
            LCD.drawString("Hello World:", 0, 0);
            LCD.drawInt(i, 0, 1);
            LCD.refresh();
            i++;
        }
    }
}
```

ThreadTest1.java

This program executes the thread HelloWorldThread until you push the button ESCAPE in your NXT brick.

```
import lejos.nxt.*;

public class ThreadTest1 {
    private static HelloWorldThread hwt;

    public static void main(String[] args){
        hwt = new HelloWorldThread();
        hwt.start();

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

        System.exit(0);
    }
}
```

ThreadTest2.java

This program executes the thread HelloWorldThread until you push the button ESCAPE in your NXT brick.

```
import lejos.nxt.*;

public class ThreadTest2 {
    private static HelloWorldThread hwt;

    public static void main(String[] args){
        hwt = new HelloWorldThread();
    }
}
```

```

    hwt.start();
    hwt.setDaemon(true);

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

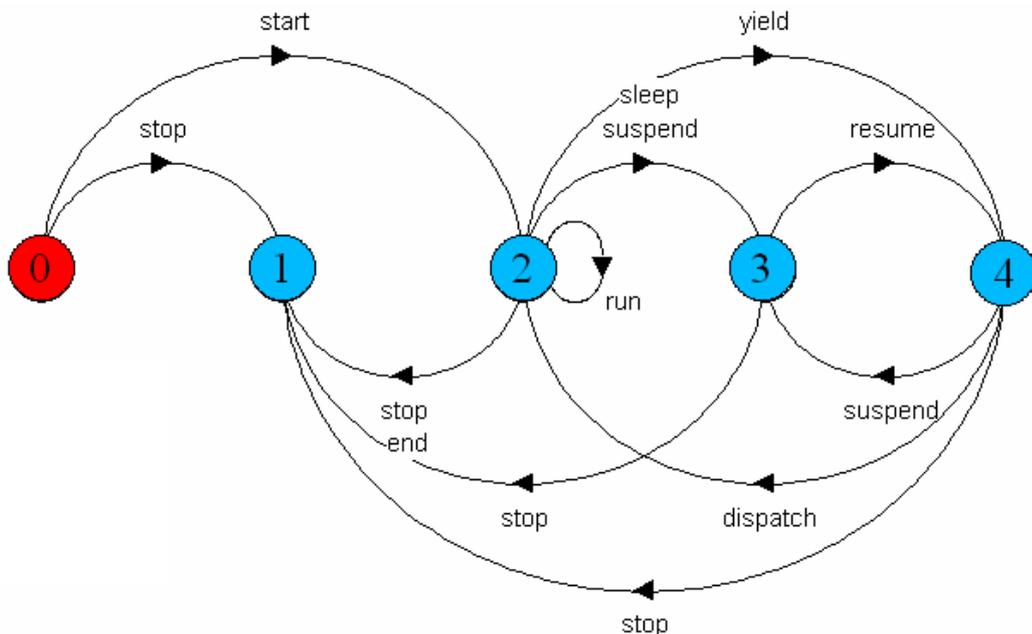
```

13.3.- The Thread life cycle

It is very important to know the life cycle of a Thread.

The possible states are the following:

- Start
- Sleep
- Interrupted
- Yield
- Join
- Interrupt



13.4.- Using the class Thread

Thread class in leJOS project has the following methods:

- Start
- IsAlive
- Sleep

13.4.1.- Method start

When you define a Thread in your robot to execute a task in some moment it is necessary to say the Thread when it is necessary to begin to run.

```
import lejos.nxt.*;

public class StartExample {
    private static HelloWorldThread hwt;

    public static void main(String[] args){
        hwt = new HelloWorldThread();
        hwt.start();
        hwt.setDaemon(true);

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

        }
    }
}
```

13.4.2.- Method isAlive

In several scenarios when you work with multiple threads, it is necessary to know if a thread is alive or not.

```
import lejos.nxt.*;

public class HelloWorldThread extends Thread{
    private int i = 0;

    public HelloWorldThread(){

    }

    public void run(){
        while(true){
            LCD.drawString("Hello World:", 0, 0);
            LCD.drawInt(i, 0, 1);
            LCD.drawString("" + this.isAlive(), 0, 2);
            LCD.refresh();
            i++;
        }
    }
}

import lejos.nxt.*;

public class IsAliveExample {
    private static HelloWorldThread hwt;

    public static void main(String[] args){
        hwt = new HelloWorldThread();
        hwt.start();
        hwt.setDaemon(true);

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

```

    }
  }
}

```

13.4.3.- Method sleep

In some scenarios, it is necessary to generate a time space to make others tasks. The method sleep is really useful.

```

import lejos.nxt.*;

public class SleepExample {
    private static String leJosMessages[] = {
        "Java leJOS NXJ",
        "Java leJOS PC API",
        "Java leJOS Mobile API",
        "Java leJOS Icommand"
    };

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

        for(int i=0;i< leJosMessages.length; i++){
            Thread.sleep(1000);
            System.out.println(leJosMessages[i]);
        }
    }
}

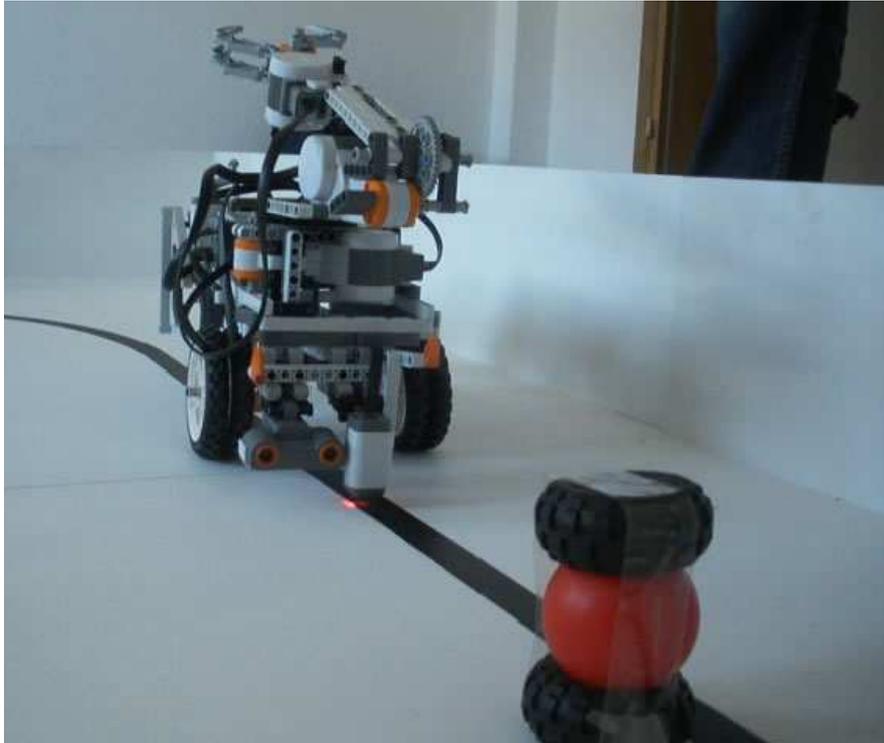
```

13.5.- Examples

This section will explain Multithreading using real examples.

13.5.1.- Example1: LineFollower

Imagine if you have a NXT Robot which has the mission to follow a black line and if it discovers an obstacle then it stops and use its arm.



The original code which was coded without any Thread:

```
import lejos.nxt.*;

public class LineFollowerOld {

    public static void main(String[] args) {
        LightSensor ss = new LightSensor(SensorPort.S3);
        ss.setFloodlight(true);
        UltrasonicSensor us = new UltrasonicSensor(SensorPort.S1);

        while (us.getDistance()>25){
            LCD.drawInt(ss.readValue(), 3, 9, 0);
            LCD.refresh();

            while (ss.readValue()< 45){
                MotorPort.C.controlMotor(0, 3);
                MotorPort.B.controlMotor(80, 1);
            }

            while(ss.readValue()>=45){
                MotorPort.B.controlMotor(0, 3);
                MotorPort.C.controlMotor(80, 1);
            }
        }

        LCD.drawString("Object found!", 0,1);
        Sound.twoBeeps();
        Sound.twoBeeps();
    }
}
```

Once you analyze the tasks:

1. Follow black lines
2. Discover obstacles in the route

It is very easy to design and develop an scalable solutions with Threads:

The main Program:

```
import lejos.nxt.*;

public class HarisRobot {
    private static DataExchange DE;
    private static LineFollower LFObj;
    private static ObstacleDetector ODObj;

    public static void main(String[] args){
        DE = new DataExchange();
        ODObj = new ObstacleDetector(DE);
        LFObj = new LineFollower(DE);
        ODObj.start();
        LFObj.start();

        while(!Button.ESCAPE.isPressed()){
            //Empty
        }
        LCD.drawString("Finished",0, 7);
        LCD.refresh();
        System.exit(0);
    }
}
```

The thread used to follow a black line:

```
import lejos.nxt.*;

public class LineFollower extends Thread {
    DataExchange DEObj;

    private LightSensor ss;
    private UltrasonicSensor us;
    private final int colorPattern = 45;

    public LineFollower(DataExchange DE){
        DEObj = DE;

        ss = new LightSensor(SensorPort.S3);
        us = new UltrasonicSensor(SensorPort.S1);

        ss.setFloodlight(true);
    }

    public void run(){
        //Infinite Task
        while(true){
            if(DEObj.getCMD() == 1){
                if(ss.readValue() < colorPattern){
                    MotorPort.C.controlMotor(0, 3);
                    MotorPort.B.controlMotor(80, 1);
                }else{
                    MotorPort.B.controlMotor(0, 3);
                    MotorPort.C.controlMotor(80, 1);
                }
            }
            LCD.drawInt(ss.readValue(), 3, 9, 0);
            LCD.refresh();
        }
    }
}
```

```

        }else{
            //Stop
            MotorPort.B.controlMotor(0, 3);
            MotorPort.C.controlMotor(0, 3);
        }
    }
}

```

The Thread used to detect Obstacles

```

import lejos.nxt.*;

public class ObstacleDetector extends Thread{
    private DataExchange DEObj;

    private UltrasonicSensor us;
    private final int securityDistance = 25;

    public ObstacleDetector(DataExchange DE){
        DEObj = DE;
        us = new UltrasonicSensor(SensorPort.S1);
    }

    public void run(){
        while(true){
            if(us.getDistance() > securityDistance){
                DEObj.setCMD(1);
            }else{
                DEObj.setCMD(0);

                //LCD Output
                LCD.drawString("Object found!", 0,1);
                LCD.refresh();
                Sound.twoBeeps();
                Sound.twoBeeps();
            }
        }
    }
}

```

Finally the Class used to exchange data among Threads:

```

public class DataExchange {
    //ObstacleDetector
    private boolean obstacleDetected = false;

    //Robot has the following commands: Follow Line, Stop
    private int CMD = 1;

    public DataExchange(){
    }

    /*
     * Getters & Setters
     */

    public void setObstacleDetected(boolean status){
        obstacleDetected = status;
    }
}

```

```

public boolean getObstacleDetected(){
    return obstacleDetected;
}

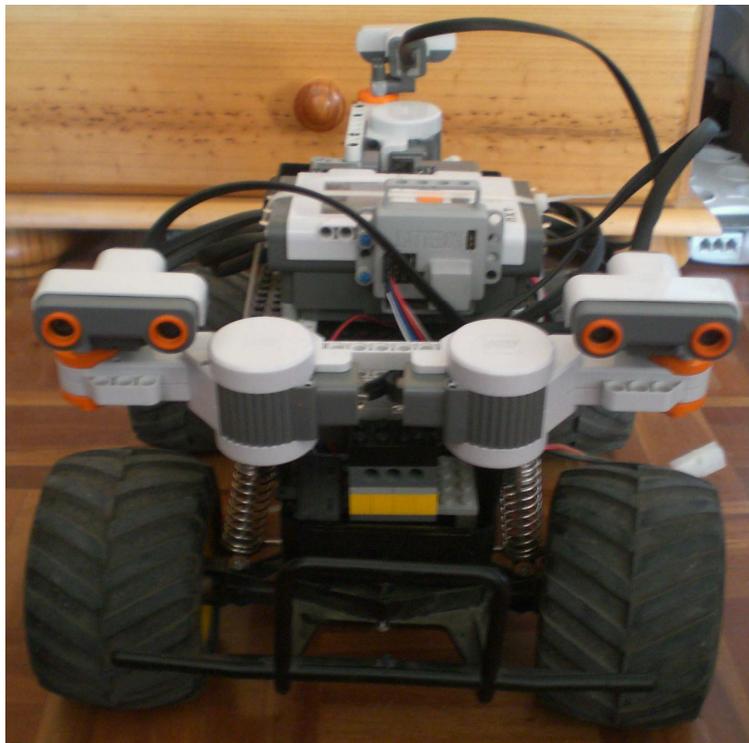
public void setCMD(int command){
    CMD = command;
}

public int getCMD(){
    return CMD;
}
}

```

13.5.2.- Example2: Stereo US

In the project RC Car managed by NXT Brick, <http://www.juanantonio.info/jab cms.php?id=228> exists 2 US Sensors which detect obstacles.



This subsystem runs in parallels with other subsystems as Navigation or GPS for example.

A tes program to test new subsystem for RC Car Project:

```

import lejos.nxt.*;

public class SUSETest {
    private static StereoUSEyes SUSEObj;

    public static void main(String[] args) throws Exception {
        TLBDataBridge TLBDB = new TLBDataBridge();
        SUSEObj = new StereoUSEyes(TLBDB);
    }
}

```

```

        SUSEObj.start();

        while(!Button.ESCAPE.isPressed()){
        }
        LCD.drawString("Finished",0, 7);
        LCD.refresh();
        System.exit(0);
    }
}

```

Note: In a future I will code a Test Case for this subsystem using JUnit for leJOS

If notice StereoUSEye is based on 2 parts: Left Eye and Righth Eye. The code to manage any eye is the following:

```

import lejos.nxt.*;

public class USEye extends Thread{
    private UltrasonicSensor US;
    private Motor motor;
    private boolean eyeSide = false;
    private boolean turnFlag = false;
    private int angles[] = {0,10,70,80};
    private int oldDistances[] = {0,0,0,0};
    private int distances[] = {0,0,0,0};

    static final boolean LEFTSIDE = true;
    static final boolean RIGHTSIDE = false;

    public USEye(UltrasonicSensor usObj, Motor mObj, boolean side){
        US = usObj;
        motor = mObj;
        eyeSide = side;
        motor.resetTachoCount();
        motor.setPower(100);
    }

    public void run(){
        //It is a infinite task
        while(true){
            SICKMode();
        }
    }

    public void SICKMode(){
        if(turnFlag){
            motor.rotateTo(0);
            oldDistances[0] = distances[0];
            distances[0] = US.getDistance();
            motor.rotateTo(10);
            oldDistances[1] = distances[1];
            distances[1] = US.getDistance();
            motor.rotateTo(70);
            oldDistances[2] = distances[2];
            distances[2] = US.getDistance();
            motor.rotateTo(80);
            oldDistances[3] = distances[3];
            distances[3] = US.getDistance();
            turnFlag = false;
        }else{

```

```

        motor.rotateTo(80);
        oldDistances[3] = distances[3];
        distances[3] = US.getDistance();
        motor.rotateTo(70);
        oldDistances[2] = distances[2];
        distances[2] = US.getDistance();
        motor.rotateTo(10);
        oldDistances[1] = distances[1];
        distances[1] = US.getDistance();
        motor.rotateTo(0);
        oldDistances[0] = distances[0];
        distances[0] = US.getDistance();
        turnFlag = true;
    }
}

public int[] getDistances(){
    return distances;
}

public int getDistance(int index){
    int distance = 0;
    if((index >= 0) && (index <= distances.length)){
        distance = distances[index];
    }else{
        distance = -1;
    }
    return distance;
}
}

```

The class designed to manage 2 USEye object is the following:

```

import lejos.nxt.*;

public class StereoUSEyes extends Thread{
    private TLBDataBridge TLBDB;

    private USEye leftEye;
    private USEye rightEye;
    private int[] leftDistances;
    private int[] rightDistances;
    private final int[] angles = {0,10,70,80};

    public StereoUSEyes(TLBDataBridge _TLBDB){
        TLBDB = _TLBDB;

        UltrasonicSensor leftUS = new
UltrasonicSensor(SensorPort.S1);
        UltrasonicSensor rightUS = new
UltrasonicSensor(SensorPort.S2);
        Motor leftMotor = Motor.A;
        Motor righthMotor = Motor.B;
        leftEye = new USEye(leftUS, leftMotor, USEye.LEFTSIDE);
        rightEye = new USEye(rightUS, righthMotor, USEye.RIGHTSIDE);
    }

    public void run(){
        leftEye.start();
        rightEye.start();
    }
}

```

```

//Set Enabled the subsystem
TLBDB.setEyesEnabled(true);

while(true){

    leftDistances = leftEye.getDistances();
    rightDistances = rightEye.getDistances();
    TLBDB.setLeftEyedistances(leftDistances);
    TLBDB.setRightEyedistances(rightDistances);

    /*
    for(int i=0; i< leftDistances.length;i++){
        LCD.drawInt(angles[i], 0, i);
        LCD.drawString("  ", 4, i);
        LCD.drawInt(leftDistances[i], 4, i);
        LCD.drawString("  ", 10, i);
        LCD.drawInt(rightDistances[i], 10, i);
    }
    LCD.refresh();
    //Exchange data with other subsystems
    *
    */
}
}
}

```

If you want to observe SUSE in action, see the following video on Youtube:
<http://www.youtube.com/watch?v=3hXTek8IFMc>



14.- LeJOS and mobile phones

14.1.- Introduction

If you need to develop Java software to be used on a mobile device, you need to install the following software on your PC Computer:

- SDK
 - J2SE SDK
 - Java ME Wireless Toolkit
- IDE
 - Eclipse
 - Eclipseme plugin
- Optional software
 - Proguard
 - Antenna
- Software to deliver Java ME software
 - Nokia PC Suite (If you use Nokia Mobile Phones)

14.2.- Install SDKs

14.2.1.- Installing J2SE SDK

Read the section Getting Started.

14.2.2.- Installing Java ME Wireless Toolkit

Java ME Wireless Toolkit is used to develop Java ME Software. Download latest release from: <http://java.sun.com/javame/downloads/index.jsp>

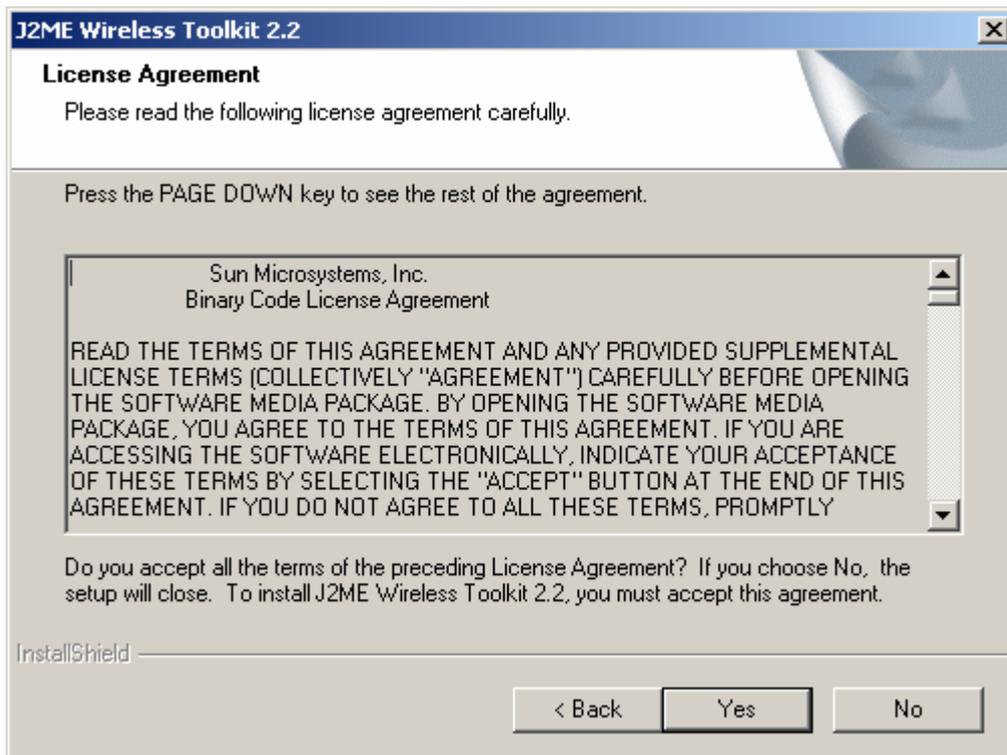
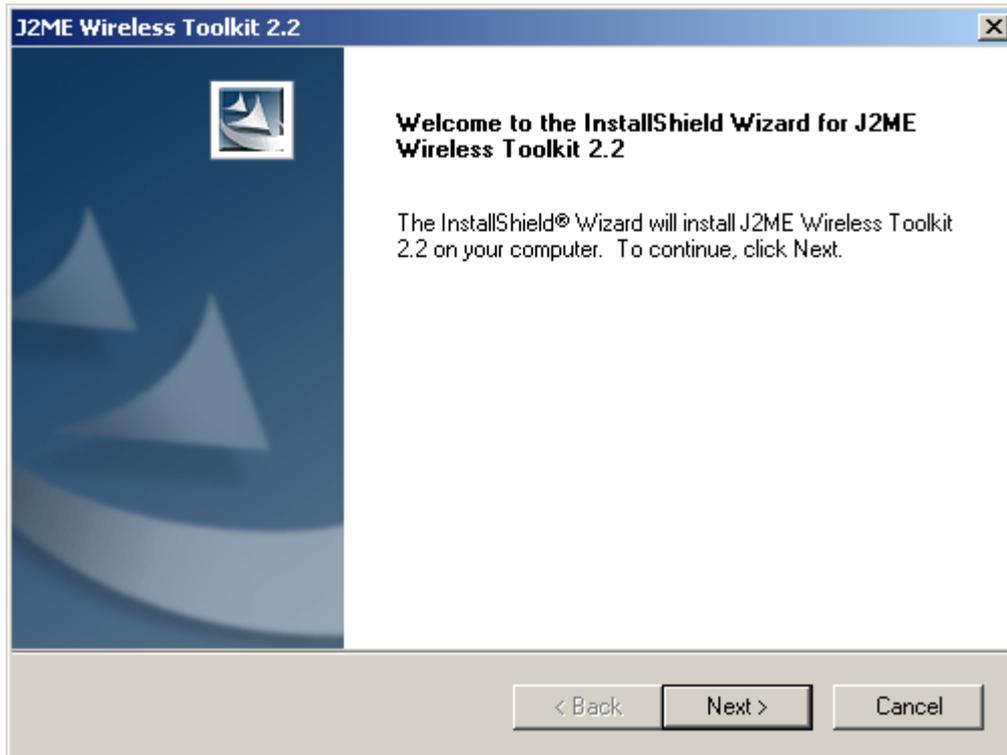
Note:

When I wrote this document I used: j2me_wireless_toolkit-2_2-windows.exe

Once you have saved the toolkit execute the installer and follow the steps.

14.2.2.1.- Licence agreement

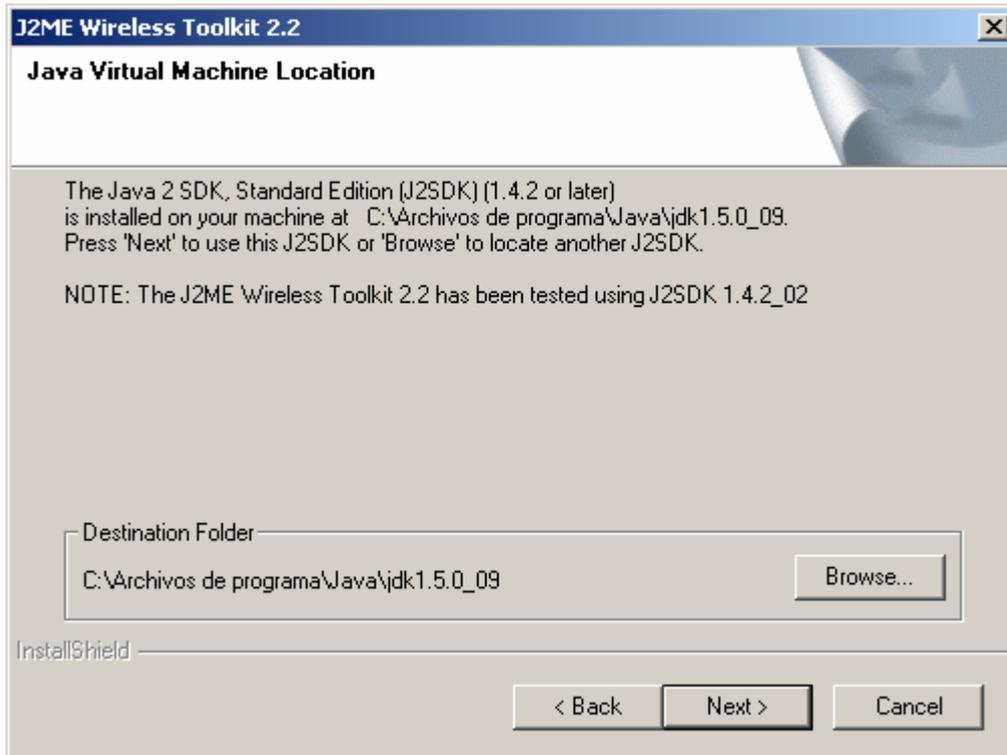
When you init the installer, click in next button to show the licence agreement.



Accept the agreement and click in yes button.

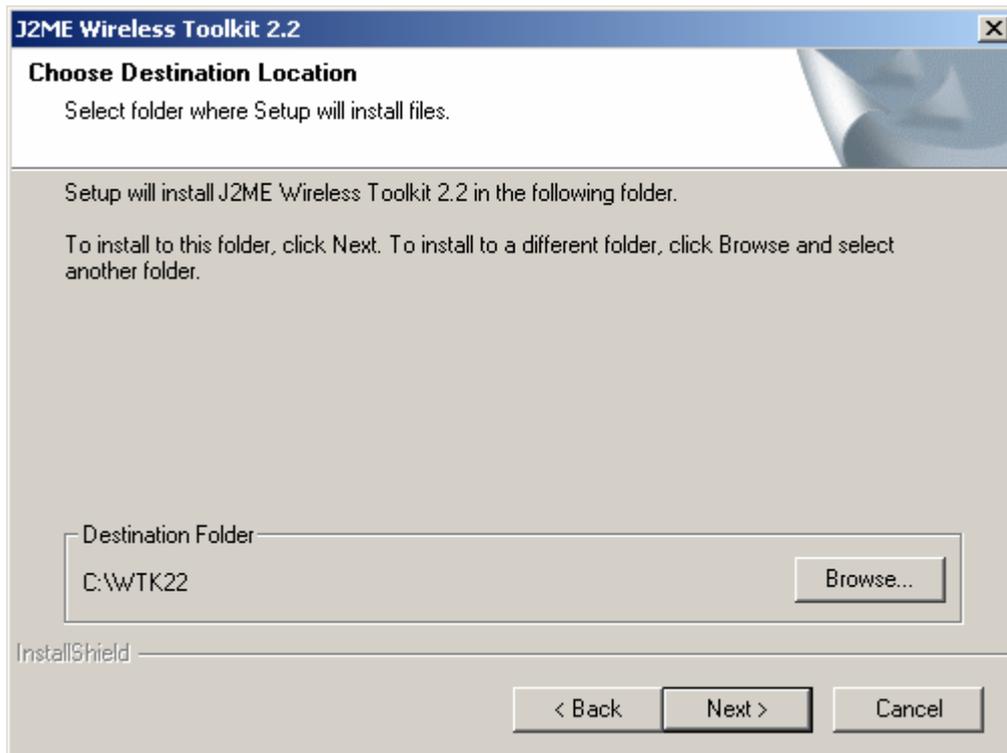
14.2.2.2.- JDK Detection

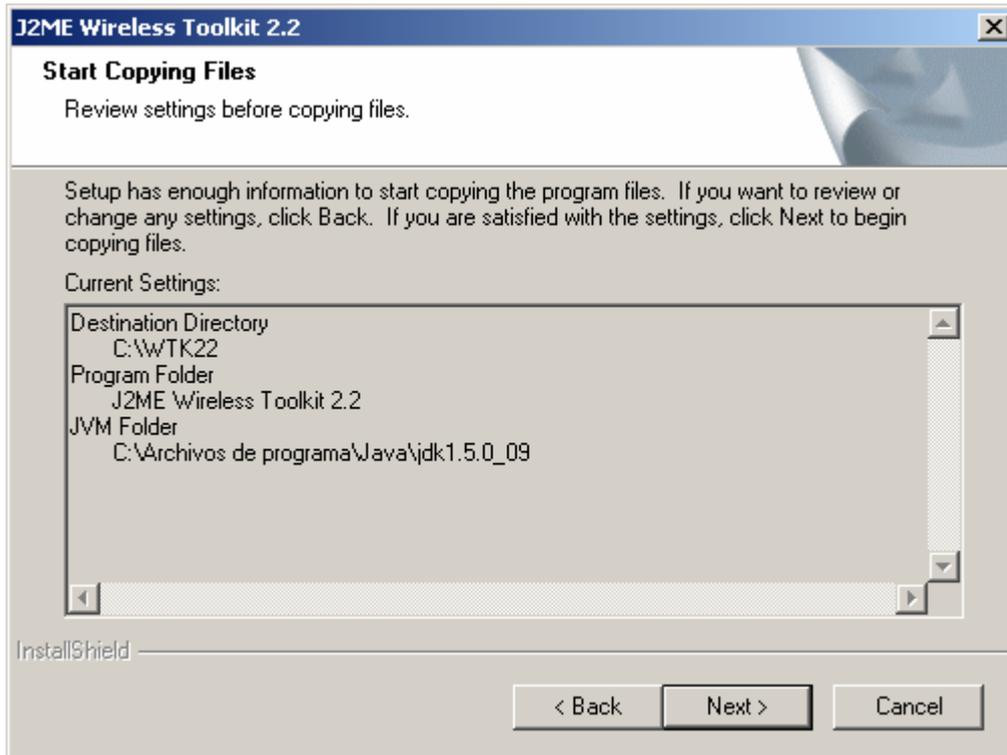
In some computers, it is possible that you have installed several J2SE SDK, then you have to select the SDK which you will link with the toolkit.



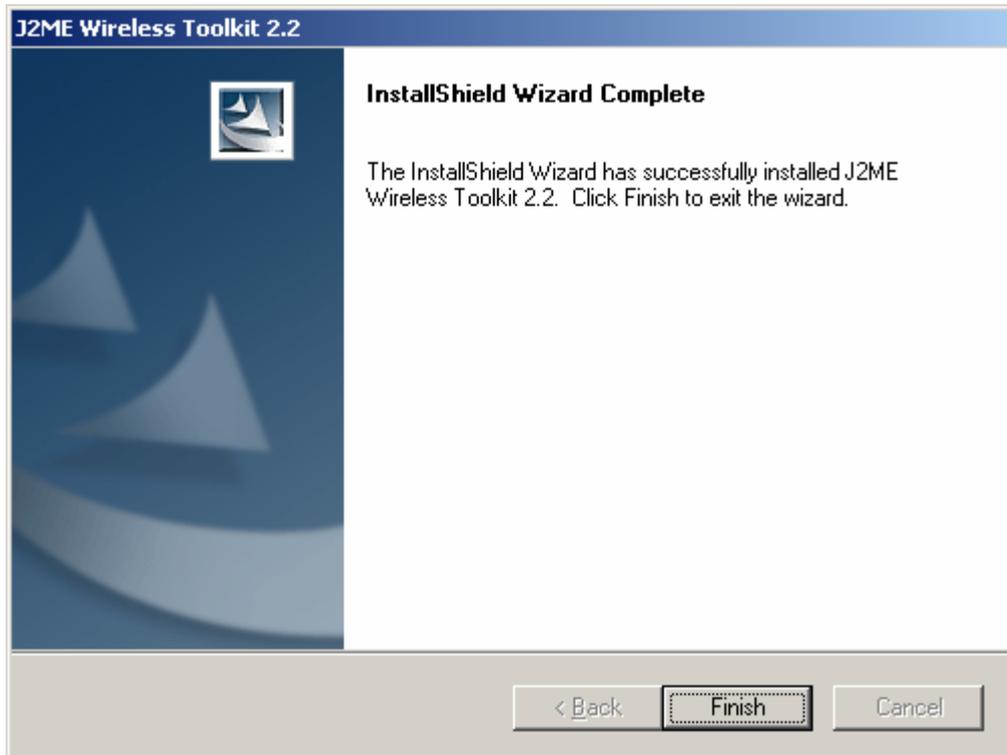
14.2.2.3.- Choose where you want to install the toolkit

Choose the place where you are going to install the toolkit. The installer gives you a recommendation if you have any suggestion, install the toolkit in the default path.





Check all parameters and click in next button to install the toolkit.



Now you have installed the Java ME Toolkit. In following points you will learn how to install a IDE to develop easily your Java ME Software

14.3.- Install IDE

14.3.1.- Installing Eclipse IDE

14.3.1.1.- 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:

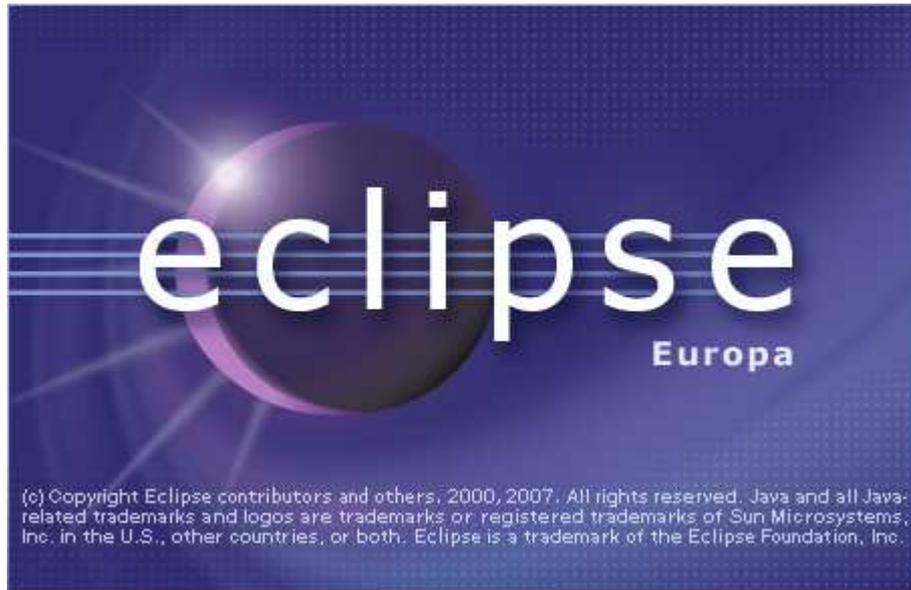
4. The Eclipse Project is responsible for developing the Eclipse IDE workbench (the "platform" for hosting Eclipse tools), the Java Development Tools (JDT), and the Plug-In Development Environment (PDE) used to extend the platform.
5. The Eclipse Tools Project is focused on creating best-of-breed tools for the Eclipse platform. Current subprojects include a Cobol IDE, a C/C++ IDE, and an EMF modeling tool.
6. 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/>

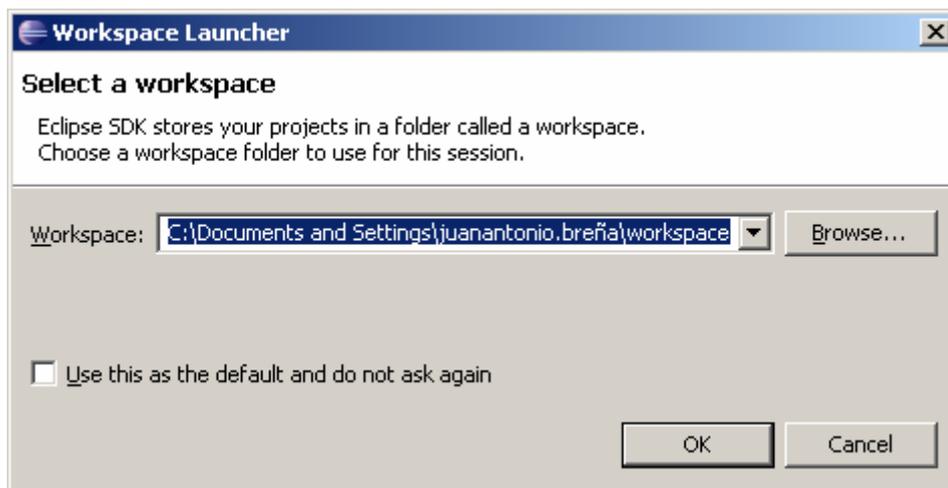
This software doesn't have any installer. Unzip latest release and execute eclipse.exe to run Eclipse IDE.

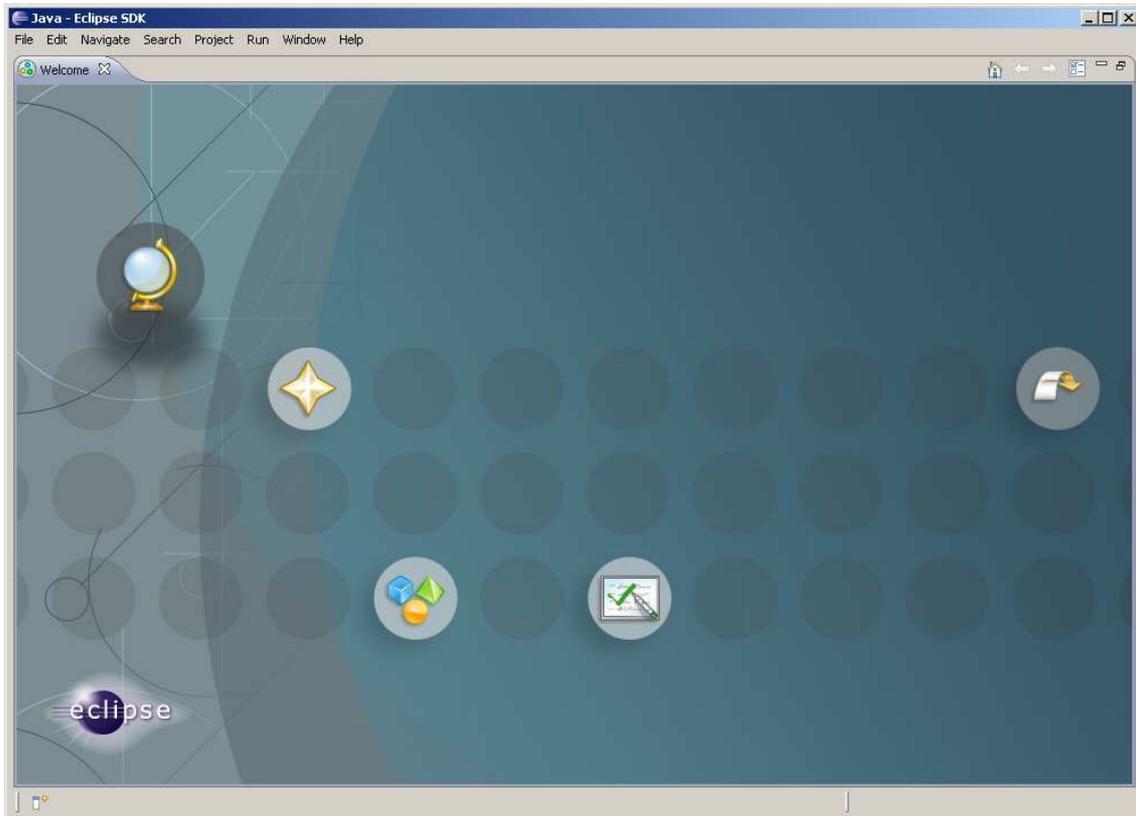


Every time that you execute Eclipse IDE, you will see the following window:



The first time that you use Eclipse, you have to determine your workspace directory:





14.3.2.- Installing Eclipse ME Plugin

EclipseME is an Eclipse plugin to help develop J2ME MIDlets. EclipseME does the "grunt work" of connecting Wireless Toolkits to the Eclipse development environment, allowing you to focus on developing your application, rather than worrying about the special needs of J2ME development.

Note: This document used the following release:

<http://download.eclipse.org/dsdp/mtj/updates/0.9/stable/>

If you need to install the plugin Eclipse ME, one way to install is using the update manager from eclipse.

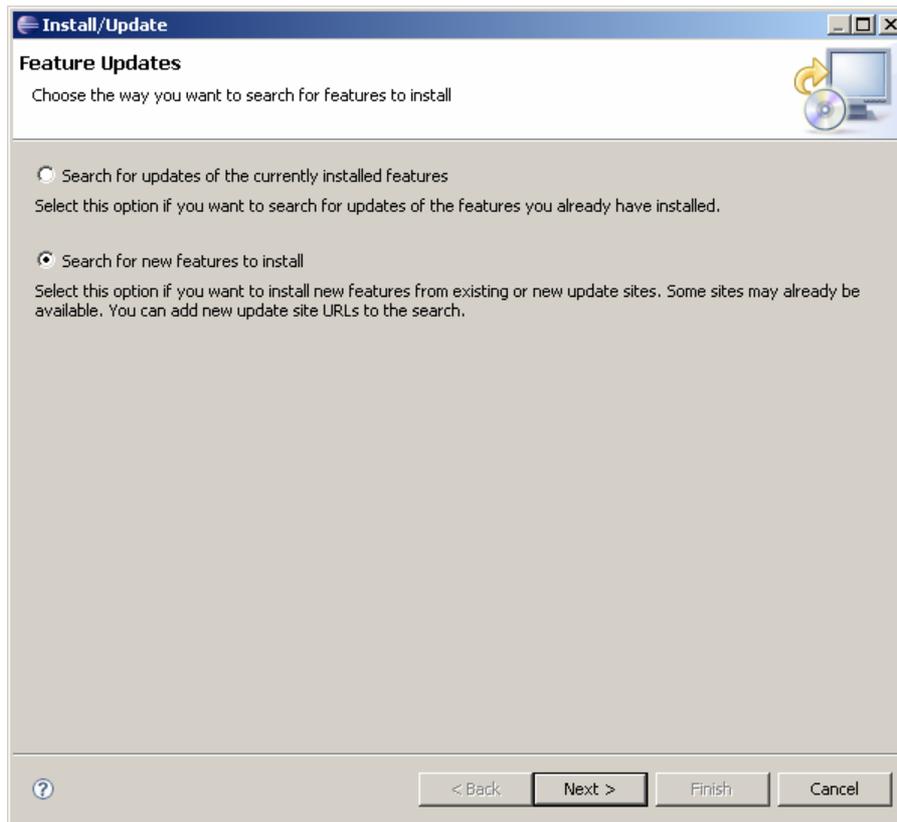
14.3.2.1.- Installation Process with Update Manager

Open Eclipse IDE and click in the main menu on the option:

Help > Software Updates > Find and Install



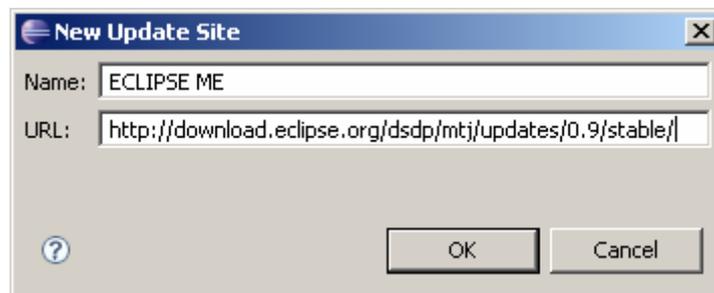
Then you will see the following assistant which will help you if you need to increase Eclipse's features. In this case you will use this assistant to install Eclipse ME Plugin.



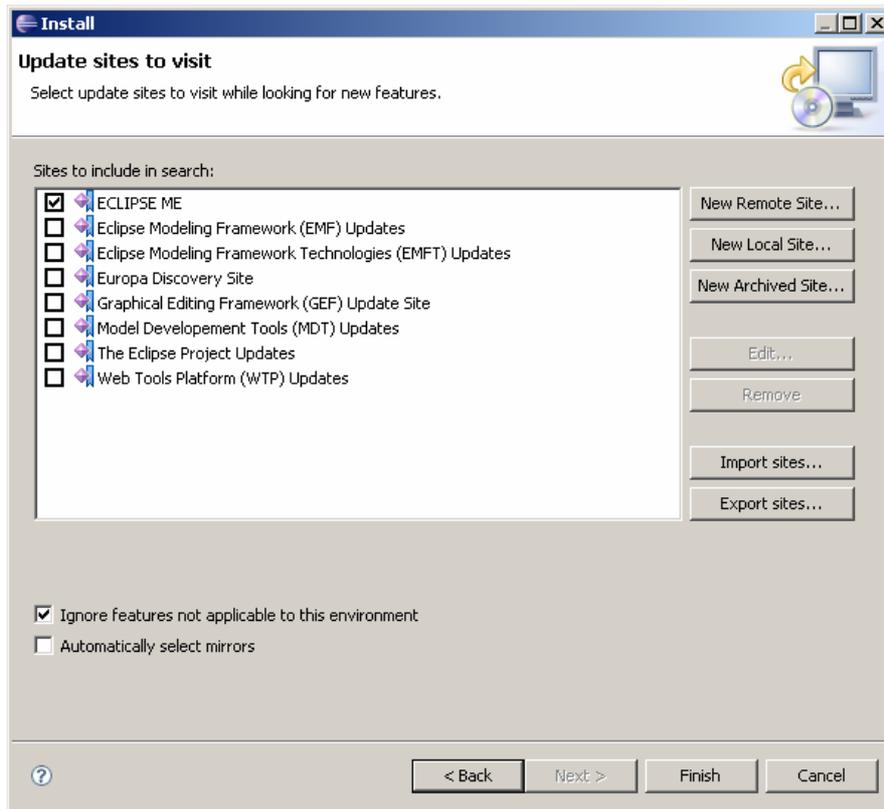
Select the option "Search for new features to install" and click in the button Next.

The URL where is located the latest release is:

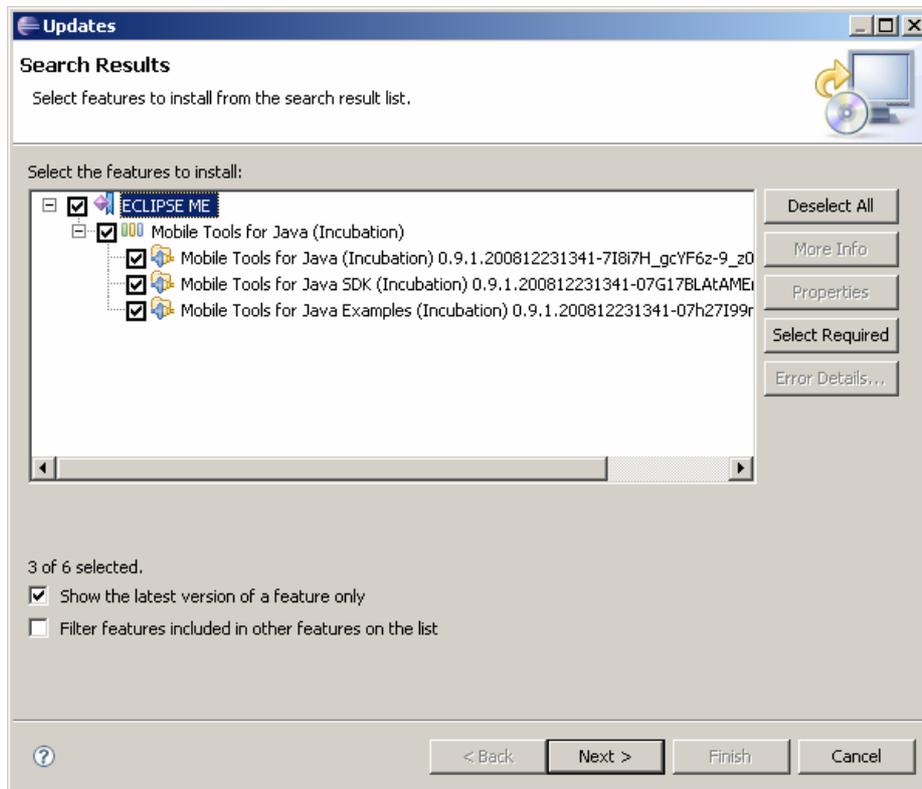
<http://download.eclipse.org/dsdp/mtj/updates/0.9/stable/>



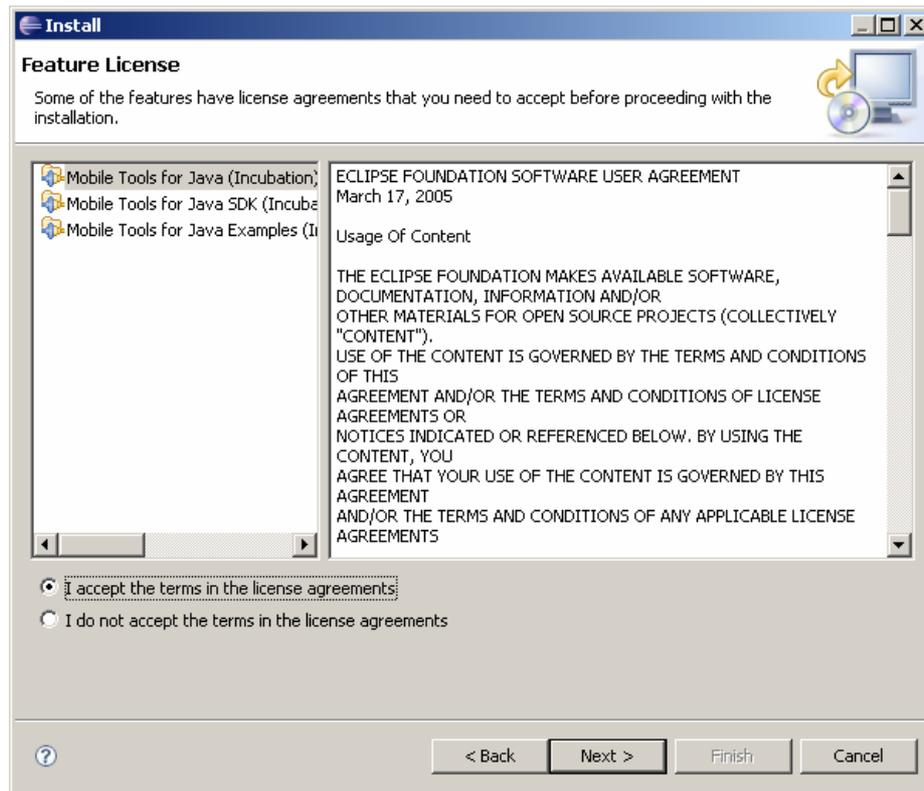
Once you have typed the Name and URL, click in the button Ok to show a new window with places where you can search to find plugins for Eclipse:



Click in the button Finish. In this moment Eclipse search in the selected options, in this case in the option Eclipse ME and Eclipse will find the following software to install:



Select all options and click in the button Next.



Accept the licence to install the plugin and click in the button Next. In this moment, Eclipse will install the plugin. Reboot eclipse to take effect the changes.

Now you have a Computer where you can develop Java ME Software.

14.4.- Install Optional Software

When you develop Java ME software, it is necessary to optimize the code and make Unit tests. Proguard and Antenna are a Open Source projects developed to cover this needs.

14.4.1.- Proguard

ProGuard is a free Java class file shrinker, optimizer, obfuscator, and preverifier. It detects and removes unused classes, fields, methods, and attributes. It optimizes bytecode and removes unused instructions. It renames the remaining classes, fields, and methods using short meaningless names. Finally, it preverifies the processed code for Java 6 or for Java Micro Edition.

Some uses of ProGuard are:

- Creating more compact code, for smaller code archives, faster transfer across networks, faster loading, and smaller memory footprints.
- Making programs and libraries harder to reverse-engineer.
- Listing dead code, so it can be removed from the source code.
- Retargeting and preverifying existing class files for Java 6, to take full advantage of Java 6's faster class loading.

You can visit the project in the following URL to download latest release
<http://proguard.sourceforge.net/>

http://sourceforge.net/project/showfiles.php?group_id=54750

Note:

This document used Proguard 4.3

Store this libraries in a unique location, for example:

D:\DATA\PROJECTS\ROBOTICS\J2ME\proguard4.3

When we create a new Java ME project, we will use Proguard.

14.4.2.- Antenna

Antenna provides a set of Ant tasks suitable for developing wireless Java applications targeted at the Mobile Information Device Profile (MIDP). With Antenna, you can compile, preverify, package, obfuscate, and run your MIDP applications (aka MIDlets), manipulate Java Application Descriptor (JAD) files, as well as convert JAR files to PRC files designed to run on the MIDP for PalmOS implementations from Sun and IBM. Deployment is supported via a deployment task and a corresponding HTTP servlet for Over-the-Air (OTA) provisioning. A small preprocessor allows to generate different variants of a MIDlet from a single source

You can visit the project in the following URL to download latest release

<http://antenna.sourceforge.net/>

http://sourceforge.net/project/showfiles.php?group_id=67420

Note:

This document used Antenna1.1

Store this libraries in a unique location, for example:

D:\DATA\PROJECTS\ROBOTICS\J2ME

When we create a new Java ME project, we will use Antenna.

14.5.- Creating your first Java ME project

In this chapter we will create our first Java ME project. We learn the following topics:

- Create a Java ME project with Eclipse
- Test the software with a simulator
- Deliver your project on your mobile

14.6.- Creating a new Java ME Project

Open Eclipse and click in the option:

File > New > Midlet Project



Then you will see an assistant to define some parameters:

New MIDlet Project

Create a MIDlet Project
Create a MIDlet project in the workspace or in an external location.

Project name: HelloWorld

Application Descriptor
Name to be used for the jad file, generated during the "Create Package" process:

- Use project name as filename
- Use custom jad file name

Jad filename:
HelloWorld.jad

Contents

- Create new project in workspace
- Create project from existing source

Directory: D:\DATA\PROJECTS\ROBOTICS\J2ME\ECLIPSEME\HelloWorld

Configurations
You can add more configurations here:

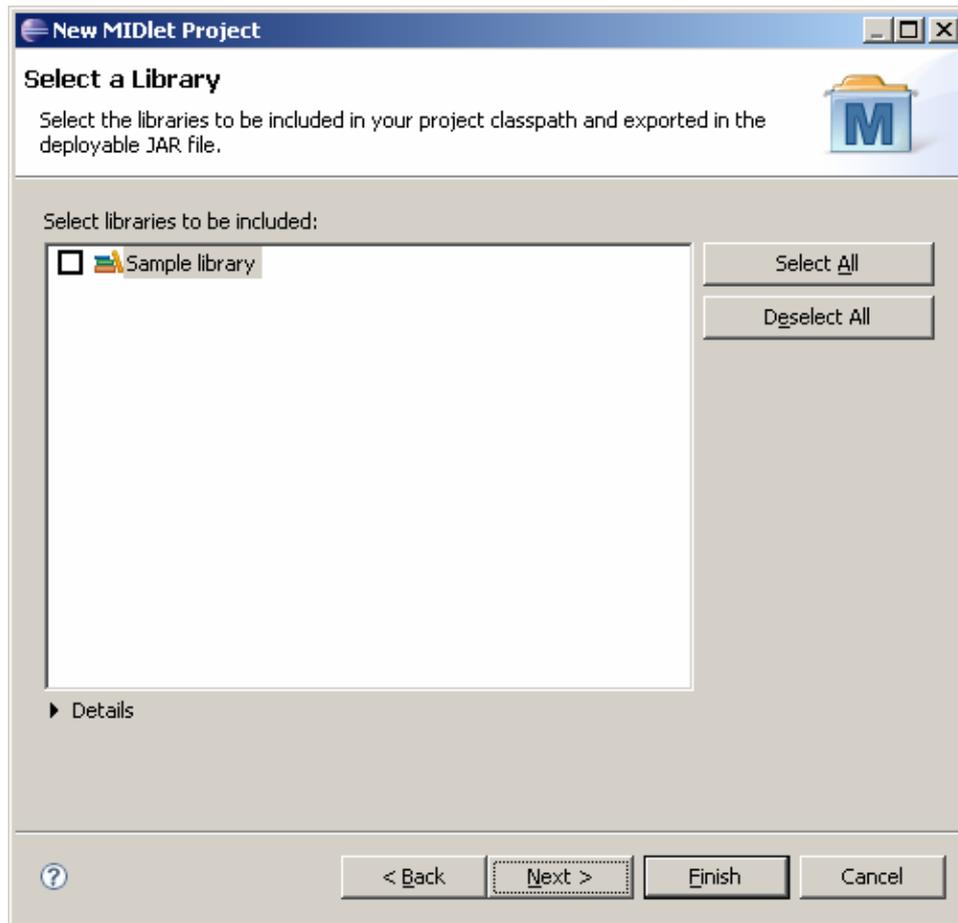
active	Configuration
<input checked="" type="checkbox"/>	DefaultColorPhone
<input type="checkbox"/>	

Preprocessor
 Enable Preprocessing Support

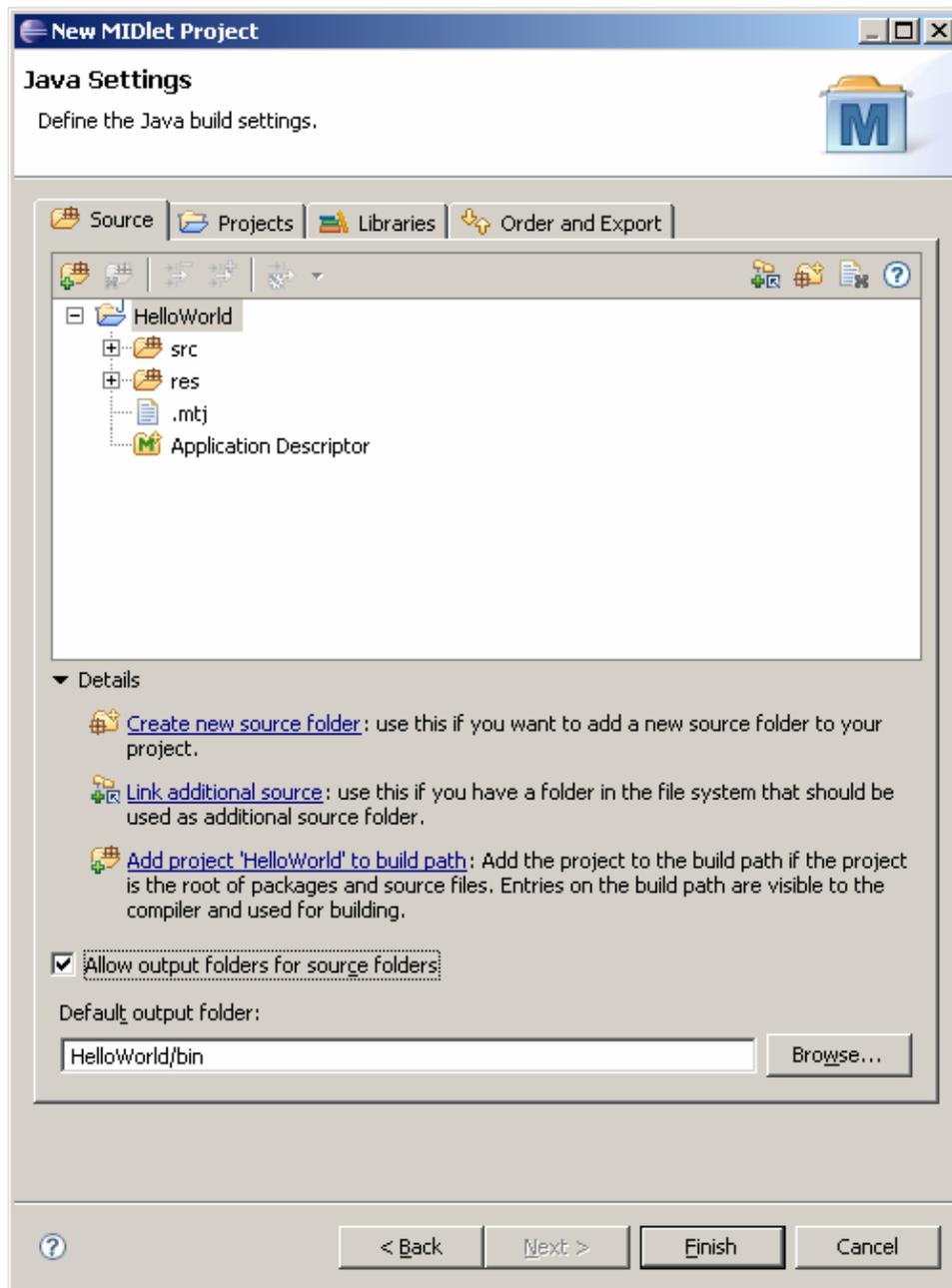
The parameters to define are:

- Project name
- Place where you are going to store the source code and bytecodes to send to your mobile phone

Click in the button Next to select a librarie used in the new project. In this example we will not include any library.



Click in next button to define others parameters:



Click in the option "Allow output folders for source folders" and click in the button Finish.

14.7.- Java ME Settings

Eclipse ME Plugin offers a control panel to define many parameters. This panel will save many hours when you have to debug your projects on your real mobile phones.

Develop leJOS programs Step by Step

Required Information

This section describes required information about this application.

MIDlet Jar URL:

MIDlet Name:

MIDlet Vendor:

MIDlet Version:

Microedition Configuration:

Microedition Profile:

Packaging

Package your application:

- Create package
- Create obfuscated package

Exporting

Generate your Buildfiles based on the configuration of the MIDlet Project:

- Export Antenna Buildfiles

Running

Run your application within a Java ME device:

- Launch as emulated Java ME MIDlet
- Launch as emulated Java ME JAD

Debugging

Debug your application within a Java ME device:

- Launch as emulated Java ME MIDlet in Debug mode
- Launch as emulated Java ME JAD in Debug mode

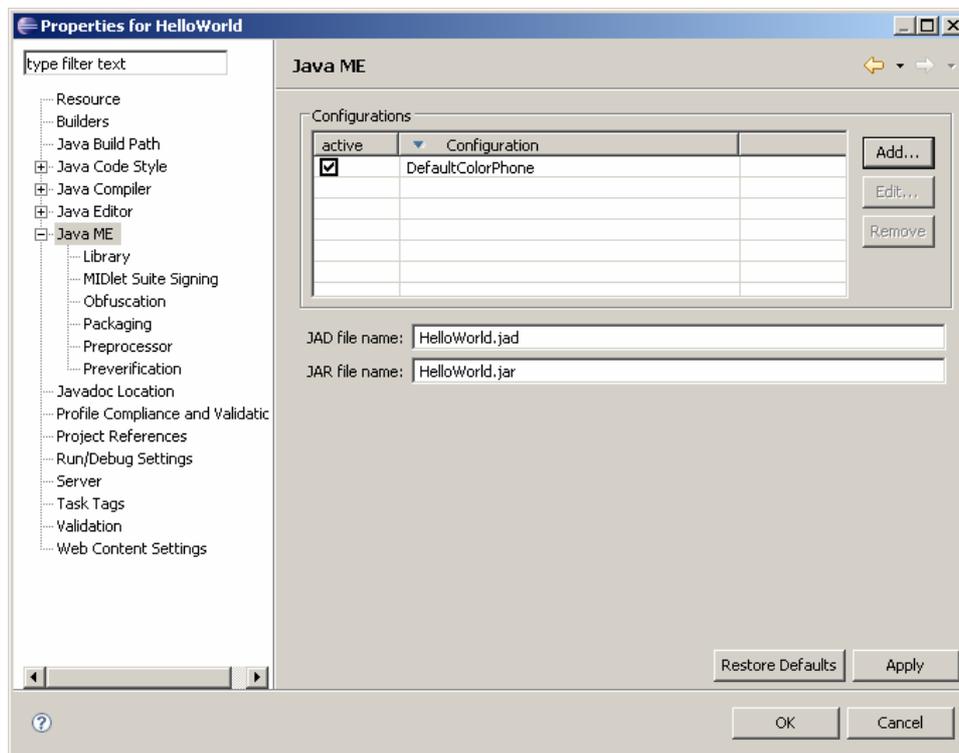
Runtime

Specify the execution environments to run this Project.

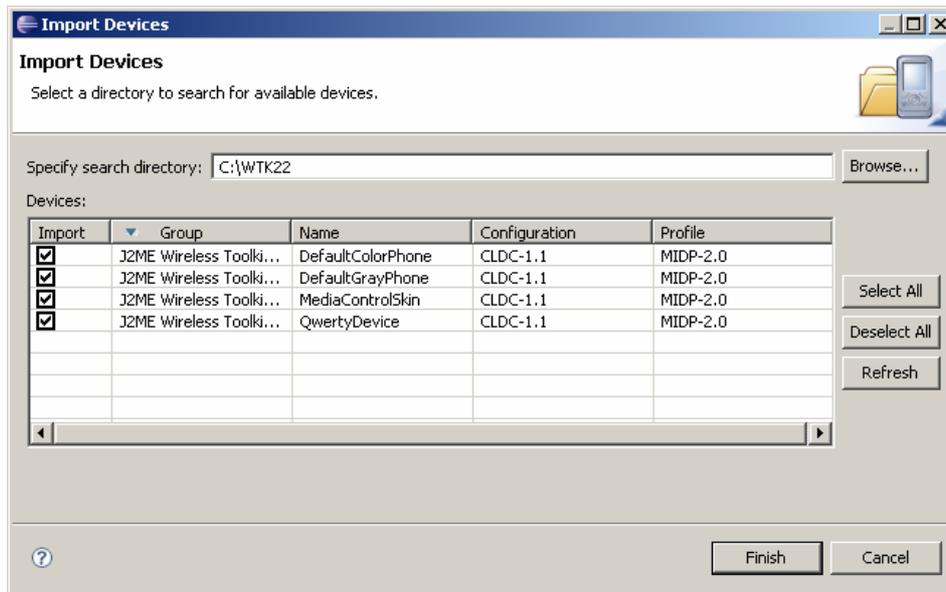
active	Configuration	
<input checked="" type="checkbox"/>	DefaultColorPhone	
<input type="checkbox"/>		

In general, you can configure Java ME Settings if you click:

Project > Properties

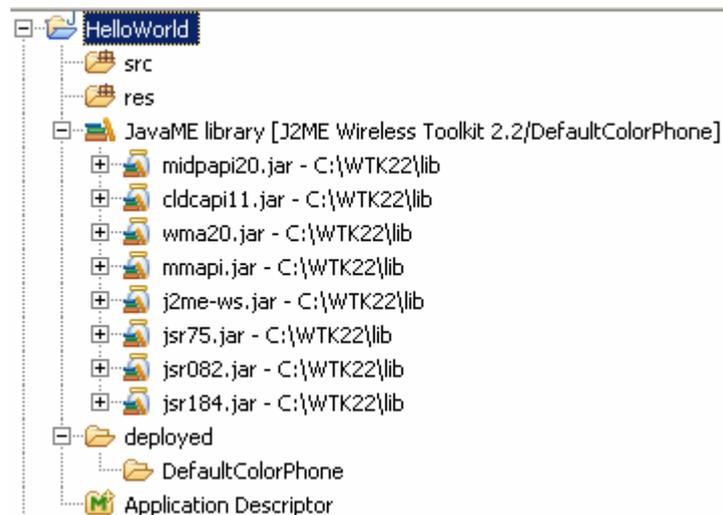


If you need to add Mobile profiles, click in Add to import profiles from Wireless Toolkit:



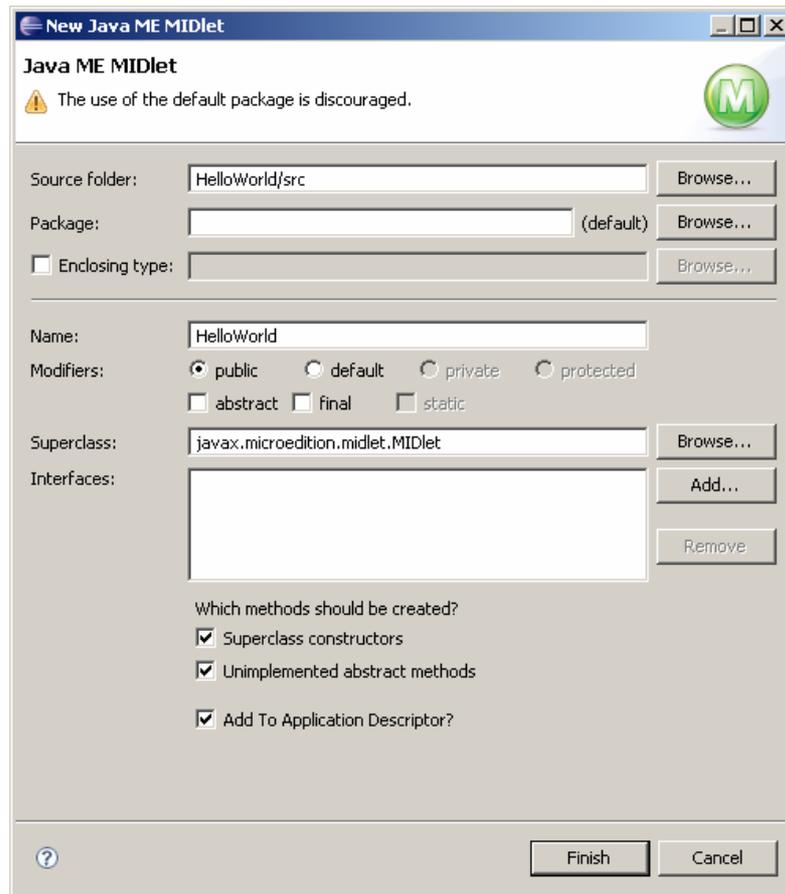
14.8.- Add a HelloWorld Middlet

For the moment, your project is empty. Add the middlet HelloWorld in the project.



select the folder src and right click to select:

new > new > Java ME Middlet



Click in the button Finish to create a new Middle in the project.

HelloWorld.java

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/**
 *
 */

public class HelloWorld extends MIDlet implements CommandListener {

    private Command exitCommand; // The exit command
    private Display display;     // The display for this MIDlet

    public HelloWorld() {
        display = Display.getDisplay(this);
        exitCommand = new Command("Exit", Command.EXIT, 0);
    }

    public void startApp() {
        TextBox t = new TextBox("Hello", "Hello, World!", 256, 0);
        t.addCommand(exitCommand);
        t.setCommandListener(this);
        display.setCurrent(t);
    }

    public void pauseApp() {
    }
}
```

```

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable s) {
    if (c == exitCommand) {
        destroyApp(false);
        notifyDestroyed();
    }
}
}

```

14.9.- Test your Middlet with a simulator

Once you have code your first Middlet, you need to test in your computer. When you defined the project, you indicated that you designed a Java ME solution to a Color Mobile phone but you can add more Devices.

Required Information
This section describes required information about this application.

MIDlet Jar URL:

MIDlet Name:

MIDlet Vendor:

MIDlet Version:

Microedition Configuration:

Microedition Profile:

Packaging
Package your application:

Exporting
Generate your Buildfiles based on the configuration of the MIDlet Project:

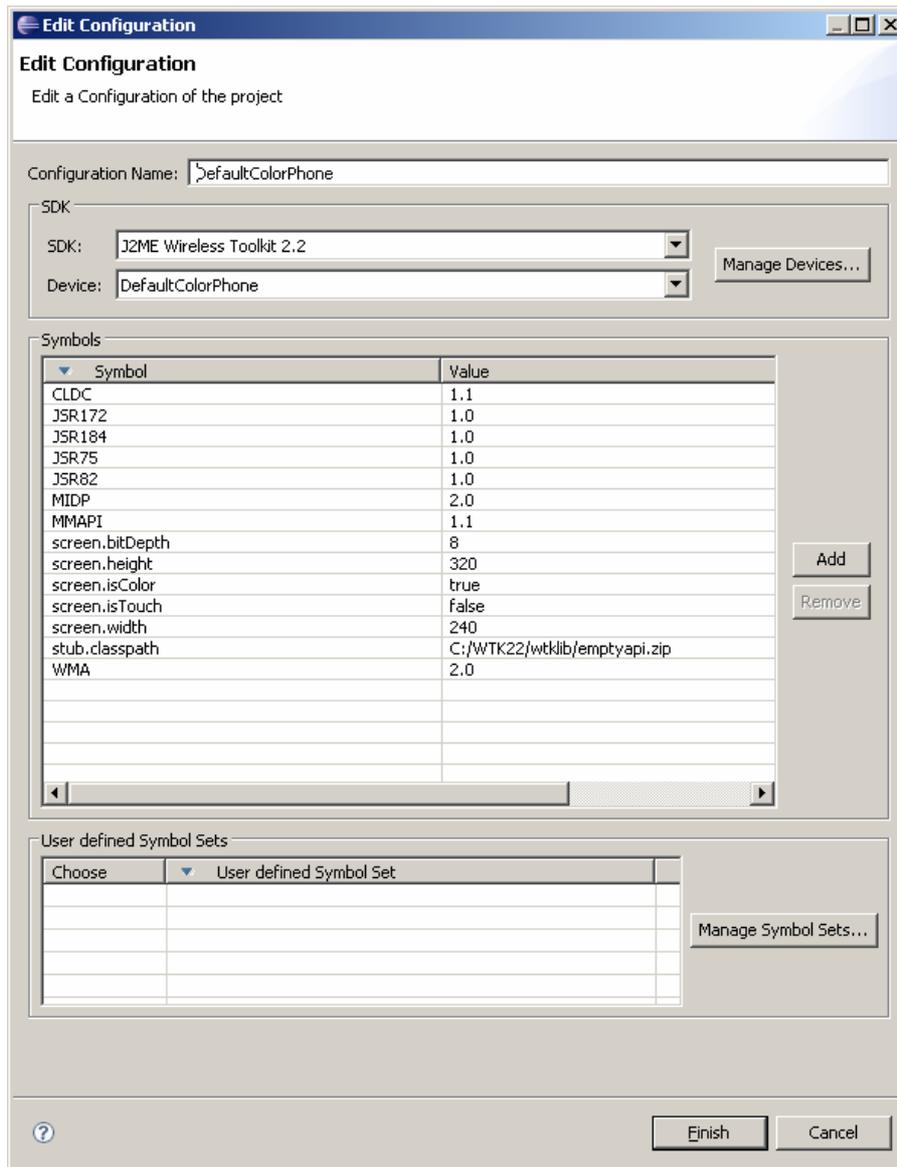
Running
Run your application within a Java ME device:

Debugging
Debug your application within a Java ME device:

Runtime
Specify the execution environments to run this Project.

active	Configuration	
<input checked="" type="checkbox"/>	DefaultColorPhone	<input type="button" value="Add..."/>
		<input type="button" value="Edit..."/>
		<input type="button" value="Remove"/>

If you select the Device, you can see the details:



To test your first Java ME project, go to the Control panel and click in the Link "Launch as emulated Java ME JAD"



Once you have clicked on the link then EclipseME will show a emulator running your program.



14.10.- Package your application

Once you consider that your Java ME application is finished, come back to the control panel and click in the link: Create package to create a package

Required Information
This section describes required information about this application.

MIDlet Jar URL:

MIDlet Name:

MIDlet Vendor:

MIDlet Version:

Microedition Configuration:

Microedition Profile:

Packaging

Package your application:

Create package

Create unscrubbed package

Exporting

Generate your Buildfiles based on the configuration of the MIDlet Project:

Export Antenna Buildfiles

Running

Run your application within a Java ME device:

Launch as emulated Java ME MIDlet

Launch as emulated Java ME JAD

Debugging

Debug your application within a Java ME device:

Launch as emulated Java ME MIDlet in Debug mode

Launch as emulated Java ME JAD in Debug mode

Runtime

Specify the execution environments to run this Project.

active	Configuration	
<input checked="" type="checkbox"/>	DefaultColorPhone	<input type="button" value="Add..."/>
		<input type="button" value="Edit..."/>
		<input type="button" value="Remove"/>

Eclipse ME create a package and descriptor file (.JAD)

14.11.- Deliver your Middlet into a Mobile Phone

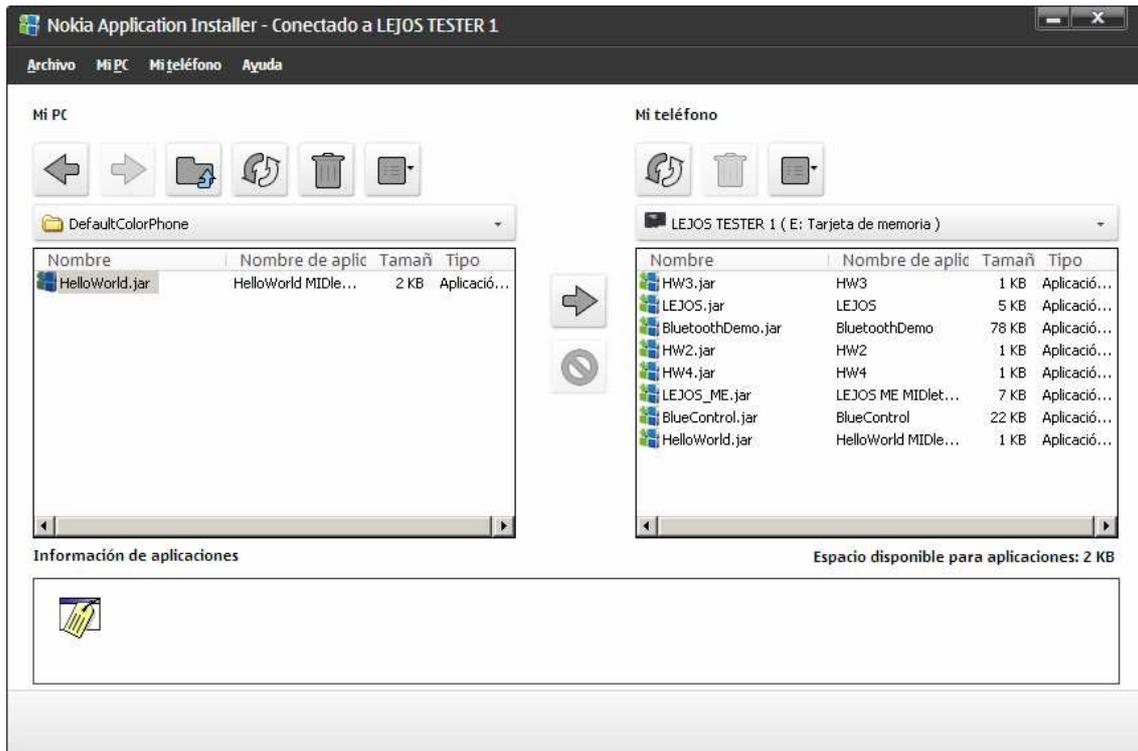
Once you have developed your Middlet application and you made several tests then you can deliver your software into a Mobile Phone.

14.11.1.- Deliver Java ME with Nokia PC Suite

If you have a Nokia Mobile Phone, Nokia has software to manage their mobiles. You can use PC Suite to send by Bluetooth your Java ME software. Open Nokia PC Suite. When PC Suite and click in the launcher for Nokia Application Installer.

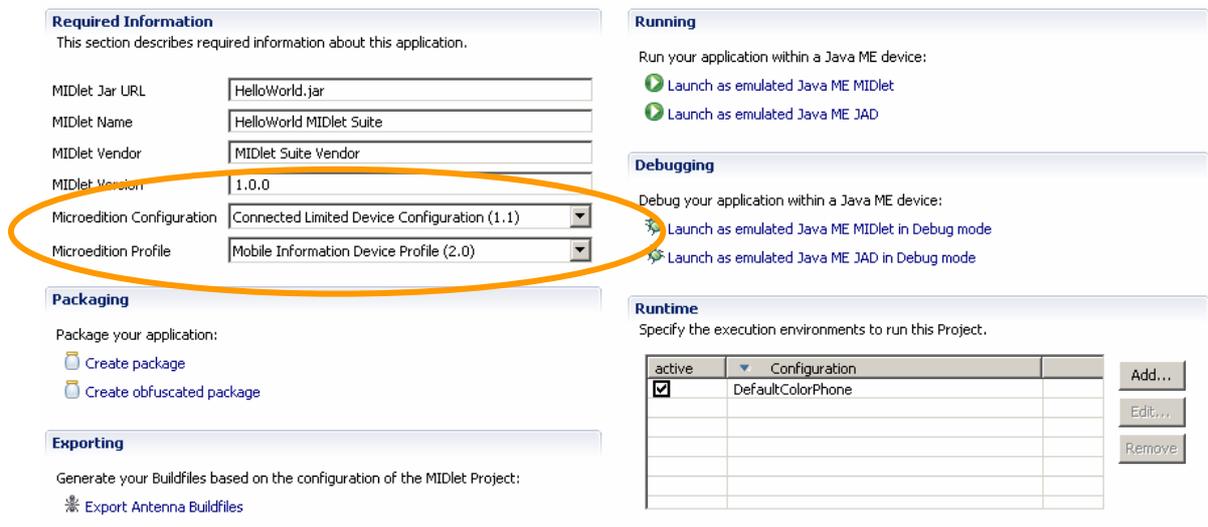


Once you have launched Nokia Application Manager, you will find your application and send into your mobile phone.



14.11.1.1.- Problems with the delivery

Along the time, Mobile phones has evolved then some mobile phones will not run with your software if this one use latest Java ME features. When you deliver a mobile application, check the following parameters from the Control Panel:



- Microedition Configuration:
 - CLDC 1.0
 - CLDC 1.1
- Microedition Profile:
 - MIDP 1.0
 - MIDP 2.0
 - MIDP 2.1
 - IMP 1.0
 - IMP NG

Check the technical specifications for your mobile phone on Internet.

15.- LeJOS Tools

15.1.- Introduction

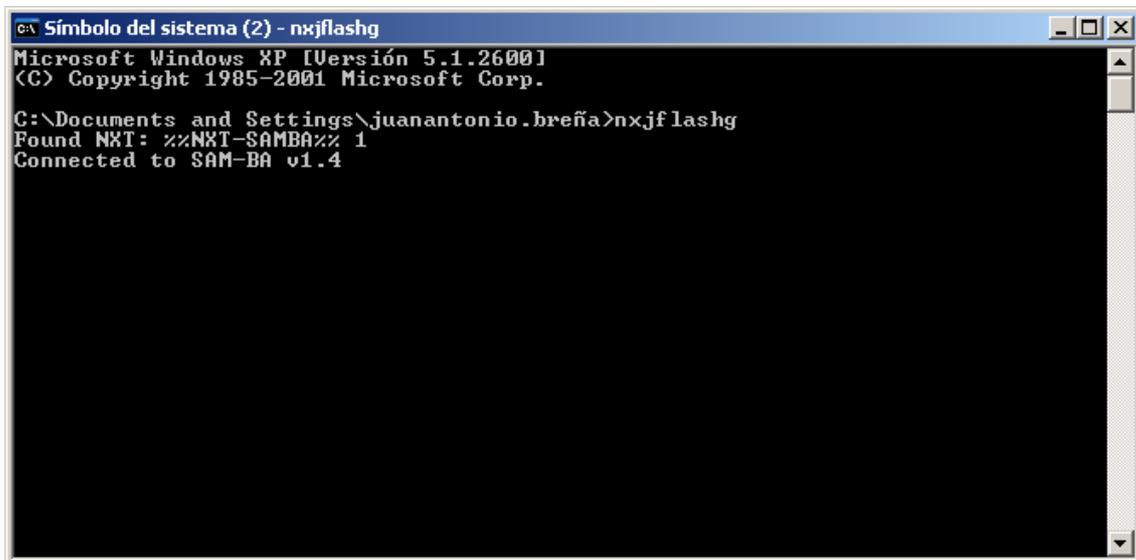
LeJOS project has developed some tools to help in your leJOS projects.

A summary about leJOS tools are:

Tool	Description
Nxjflash	A ghell command tool designed to install leJOS Firmware
Nxjflashg	A graphical tool designed to install leJOS Firmware
Nxjbrowse	A graphical tool designed to opeate over NXT's filesystem
Nxjconsole	A graphical tool designed to show messages from any NXJ program which use RConsole
Nxjmonitor	A graphical tool designed to show values from Motors, Sensors and Battery
Statemachine developer kit	A Eclipse plugin designed to model lejos application based on statemachines.

15.2.- NXJFlashg

Nxjflashg is a leJOS tool designed to install/reinstall leJOS firmware. To use the tool, open a shell window and type the command nxjflashg.



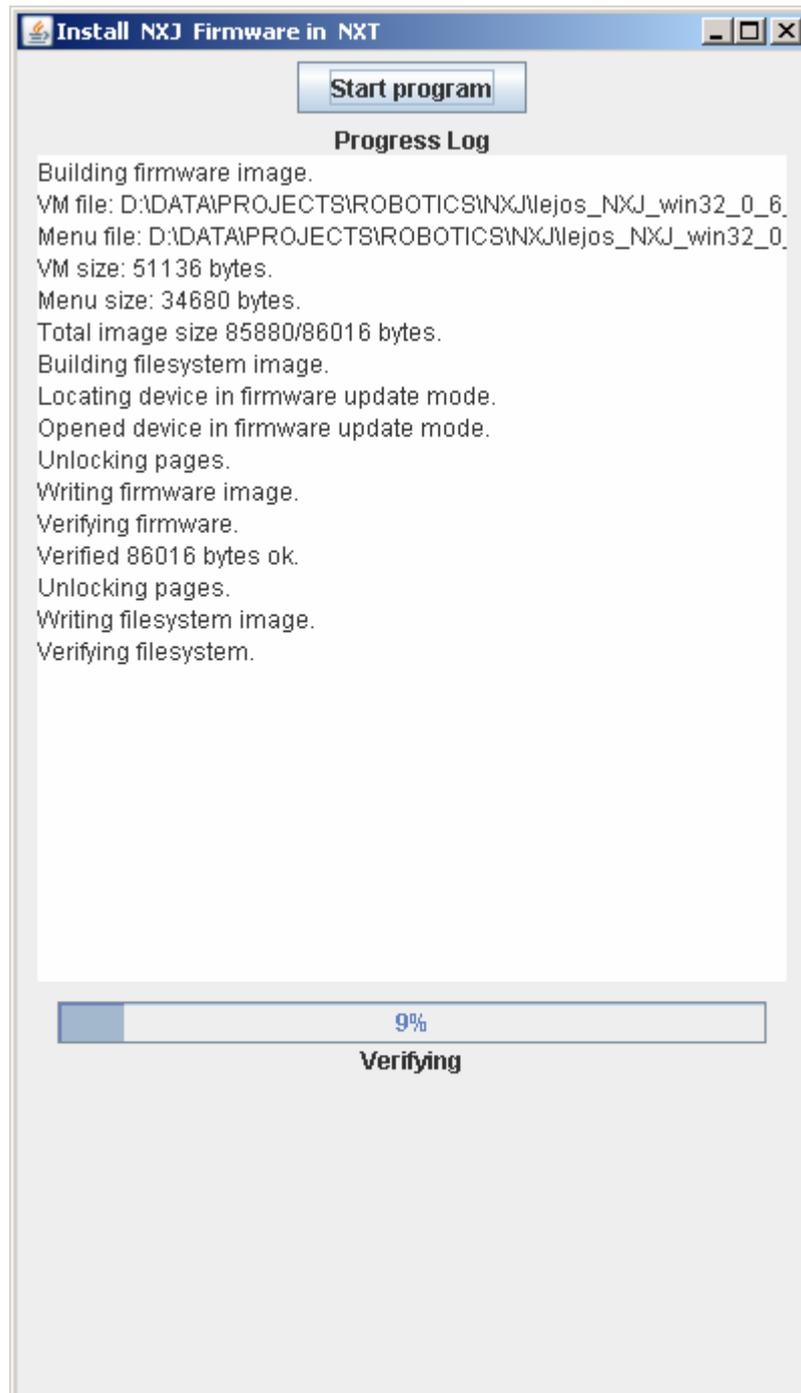
```

c:\ Símbolo del sistema (2) - nxjflashg
Microsoft Windows XP [Versión 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\juanantonio.breña>nxjflashg
Found NXT: %%NXT-SAMBA%% 1
Connected to SAM-BA v1.4

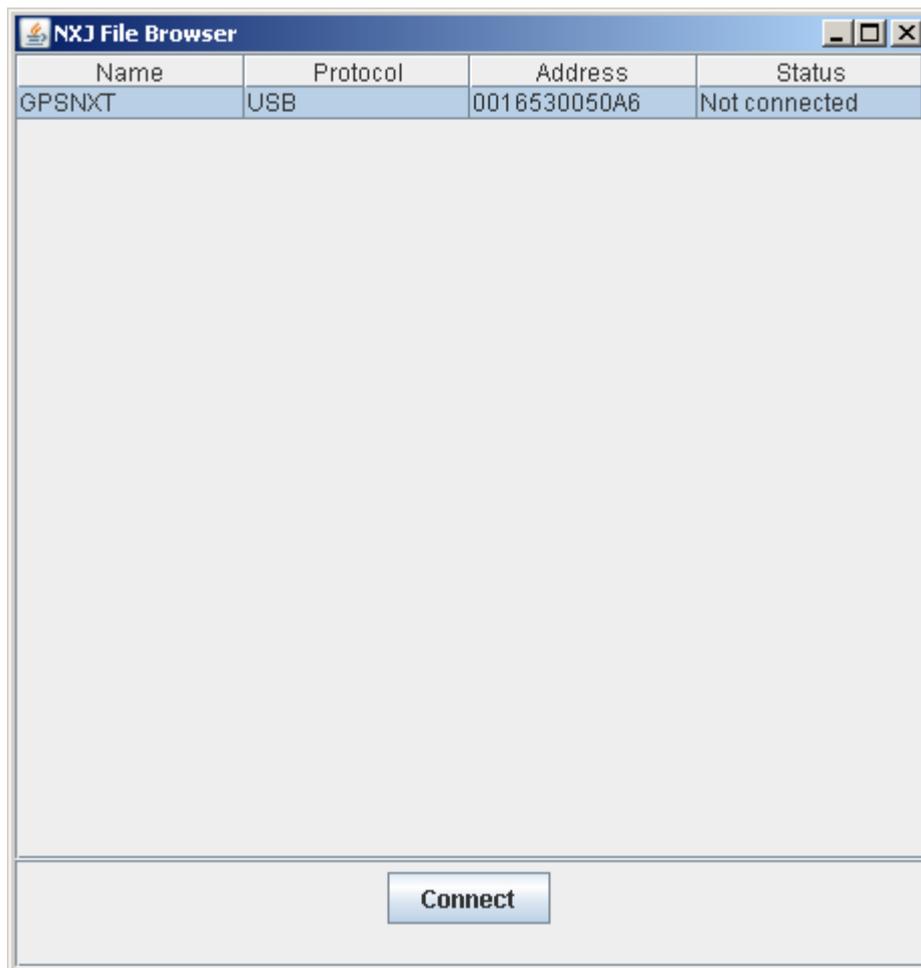
```

Once you have typed the command, you will see a window which will help you in the process to install leJOS firmware into NXT Brick.

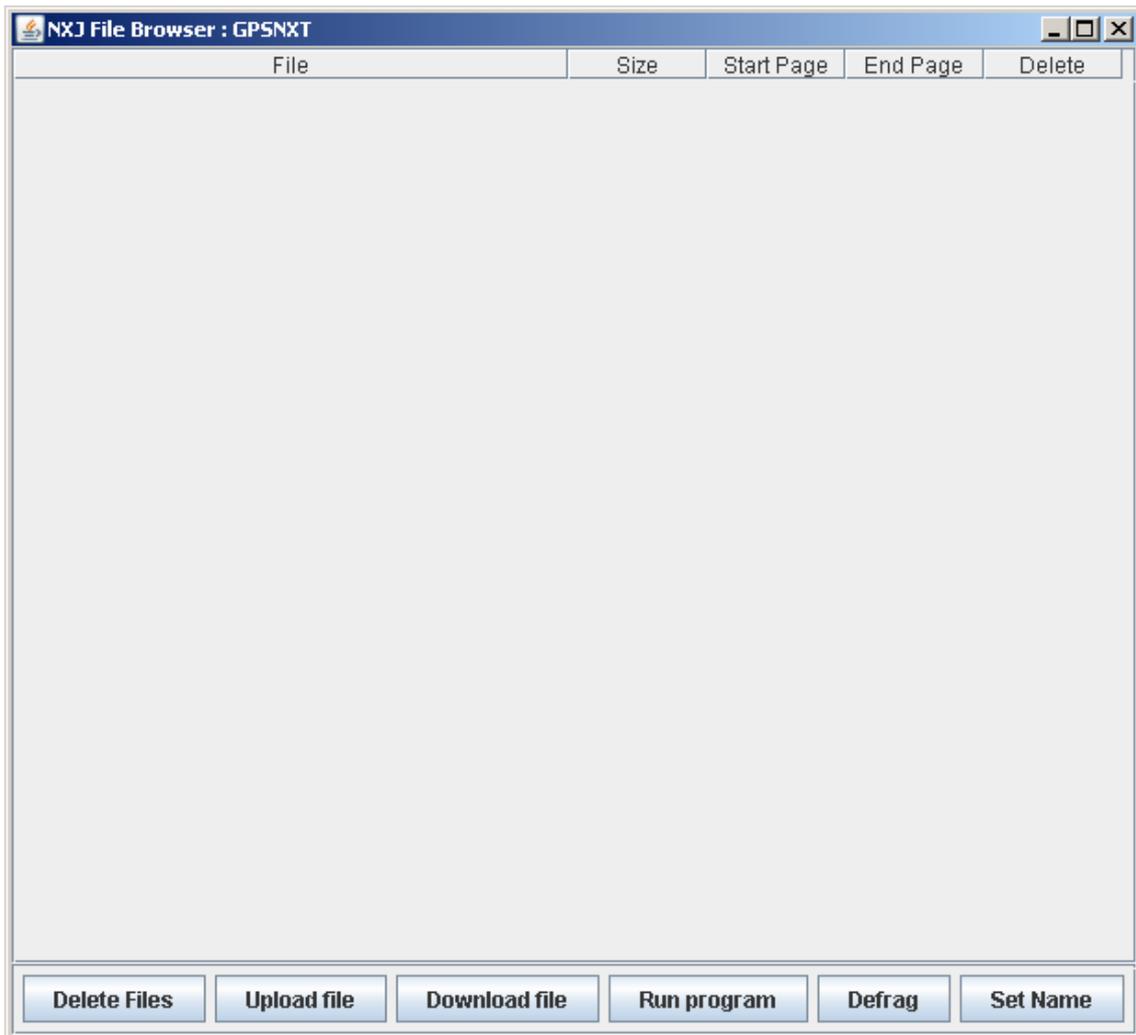


15.3.- NXJBrowse

NXJBrowse is an interesting tool to browse the filesystem. Open a shell window and type the command `nxjbrowse` then you will see a window where you will see the name of your NXT brick connected to your computer.



Click in the button "Connect" to show the filesystem.



Once you see this window, you can use this tool to make the following operations:

1. Delete files
2. Upload a NXJ program
3. Download files
4. Run a NXJ program
5. Defrag filesystem
6. Set NXT name

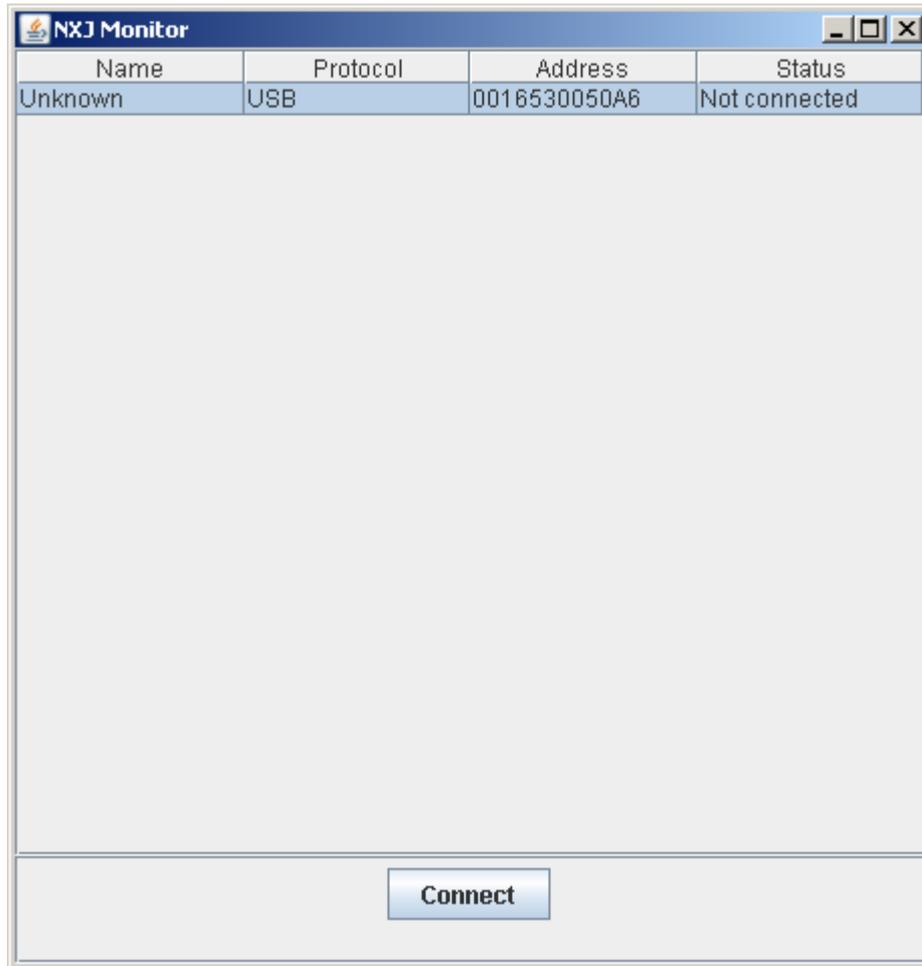
15.4.- NXJConsole

NXJConsole is a nice tool designed to debug NXJ programs. NXJConsole show messages from any NXJ program which uses RConsole.

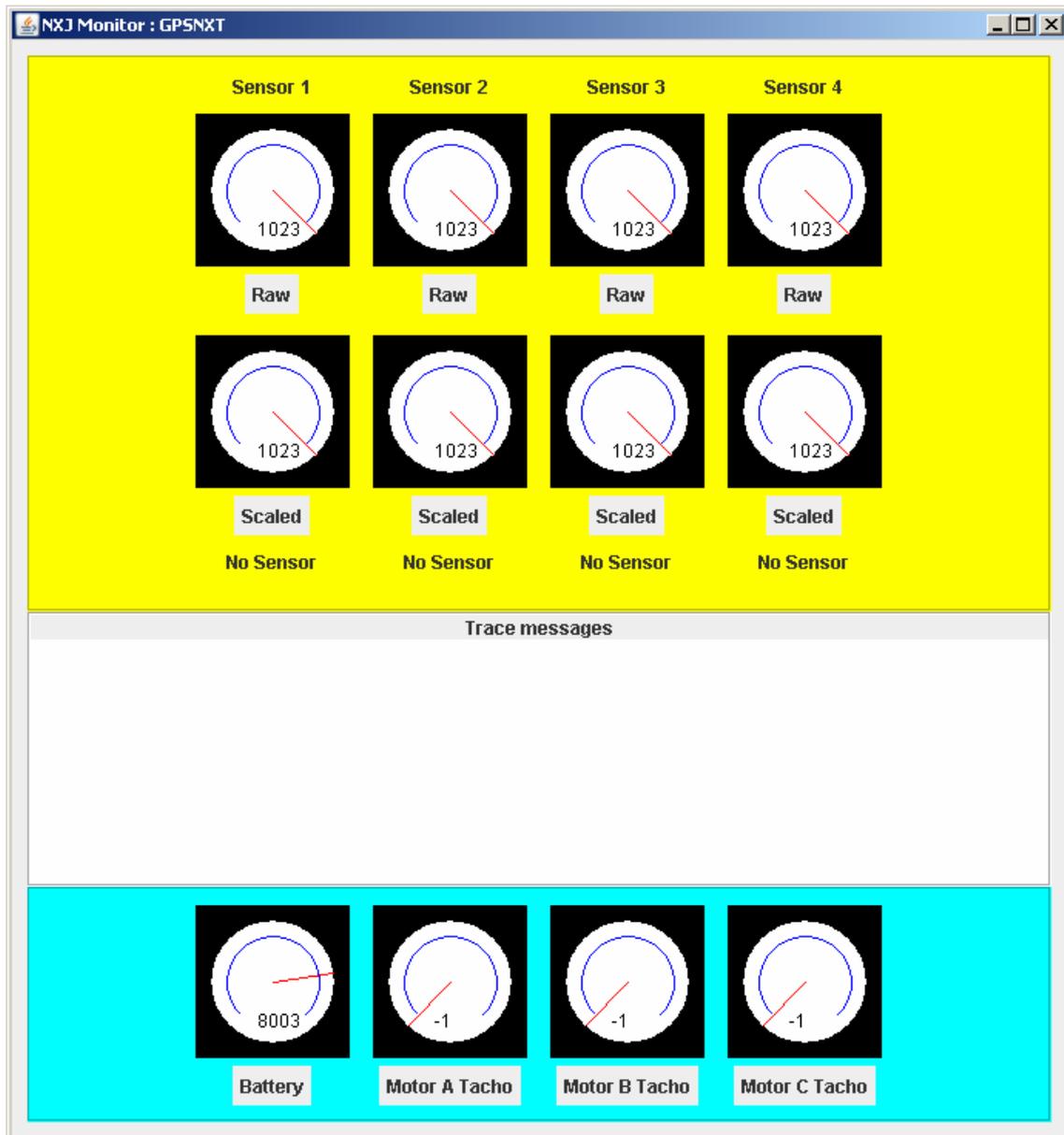


15.5.- NXJMonitor

NXJMonitor is a leJOS tool designed to read values from Motor ports, Sensor ports and Battery. To execute NXJMonitor, open a shell window and type `nxjmonitor`.



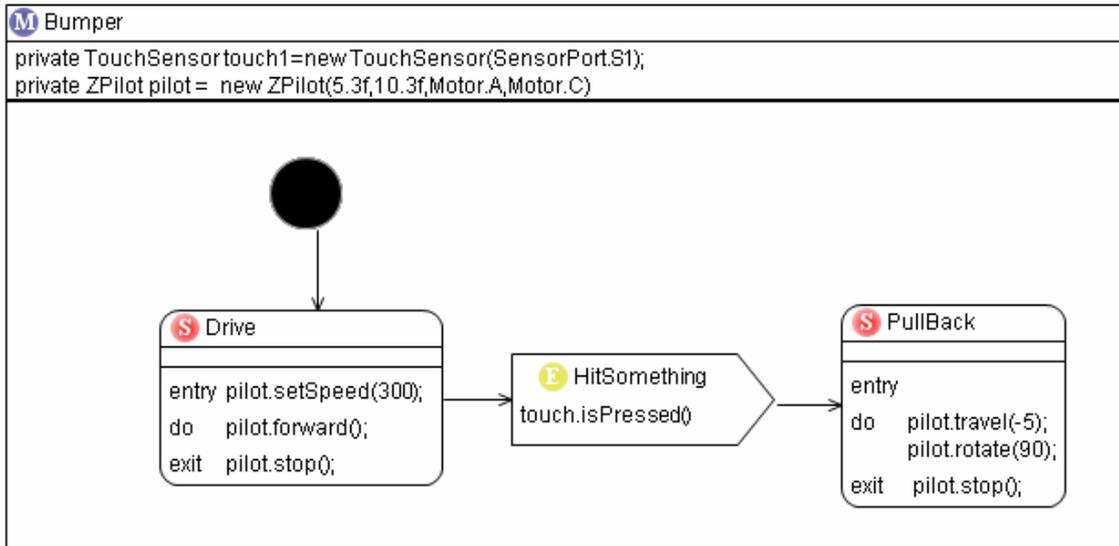
Click in the button "Connect" to show a new window to read values from Sensors, Motors and Battery.



15.6.- LeJOS statemachine developer toolkit

15.6.1.- 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.

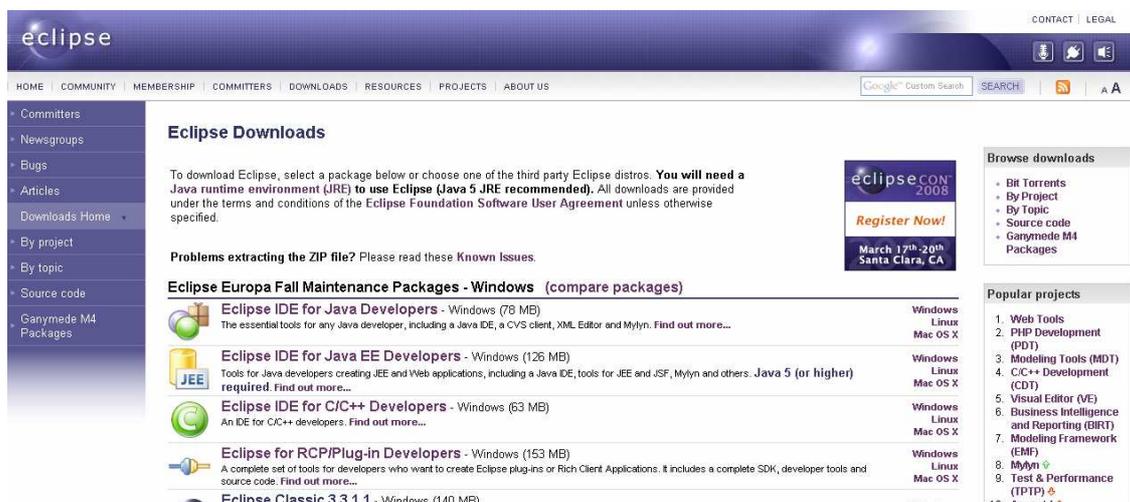
15.6.2.- Installing LeJOS statemachine developer toolkit

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

15.6.2.1.- 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/>



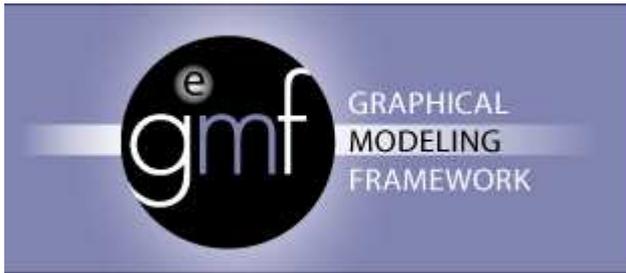
Once you have installed the IDE in your computer it is necessary to install the following features on eclipse:

- GMF, Graphics Modeling Framework
- OWA

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

15.6.2.2.- 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

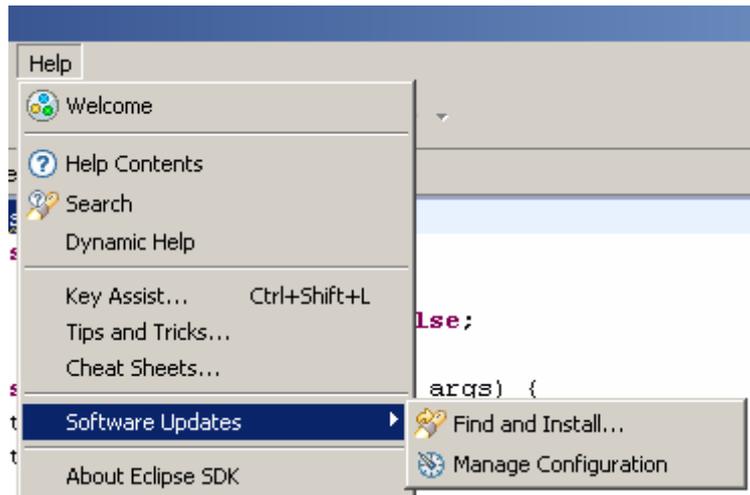


<http://www.eclipse.org/gmf/>

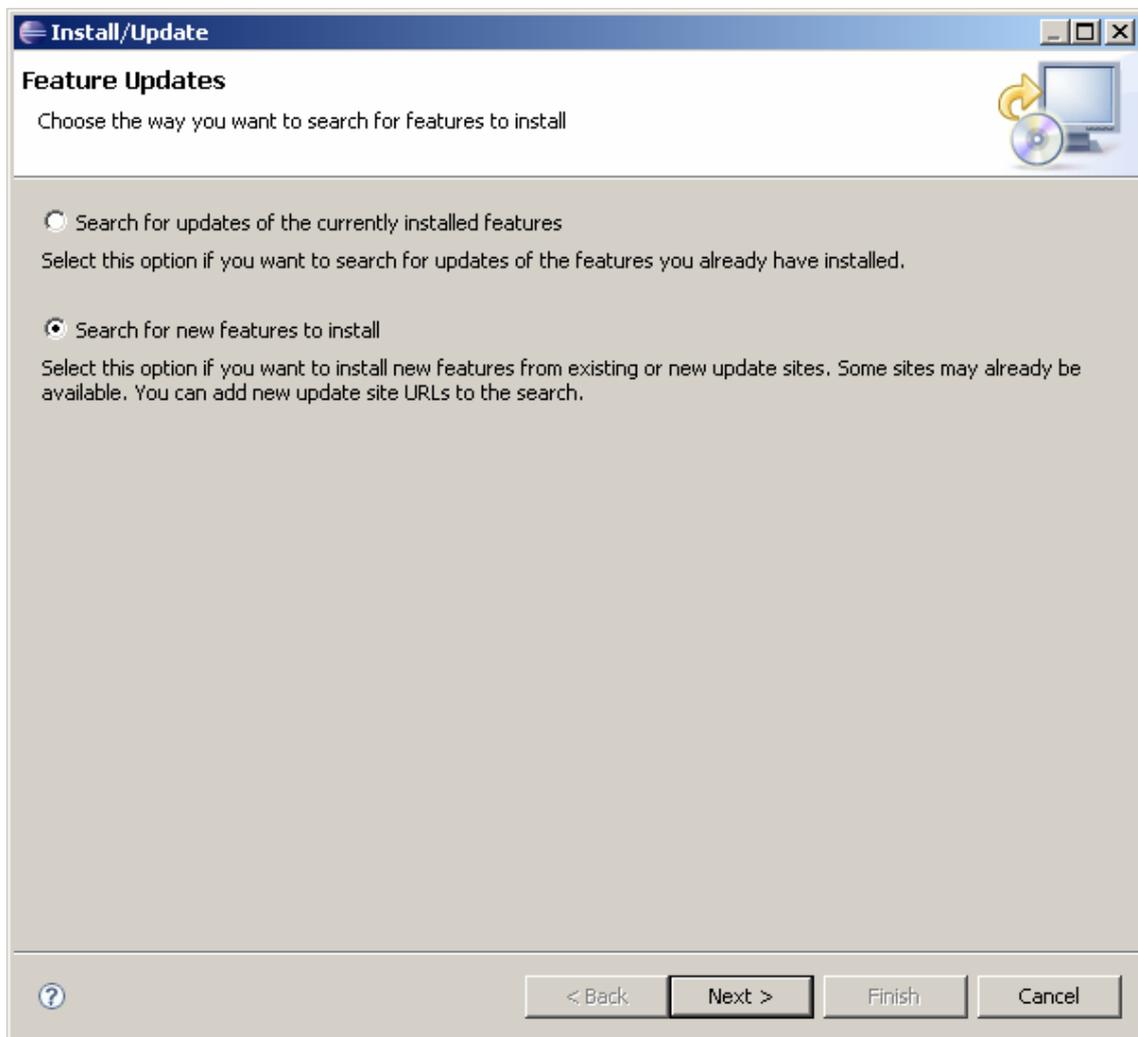
<http://download.eclipse.org/modeling/gmf/downloads/index.php>

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

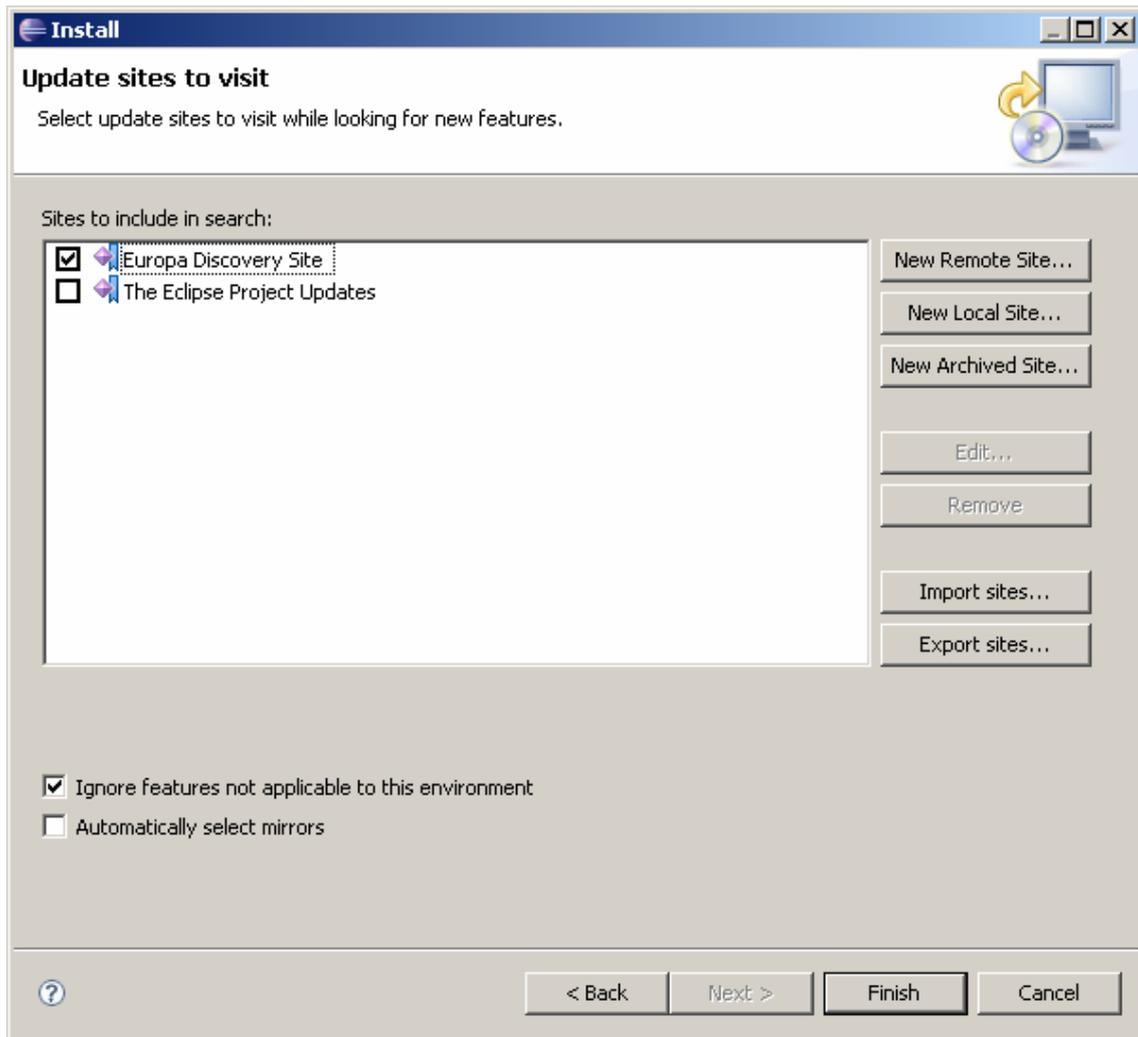




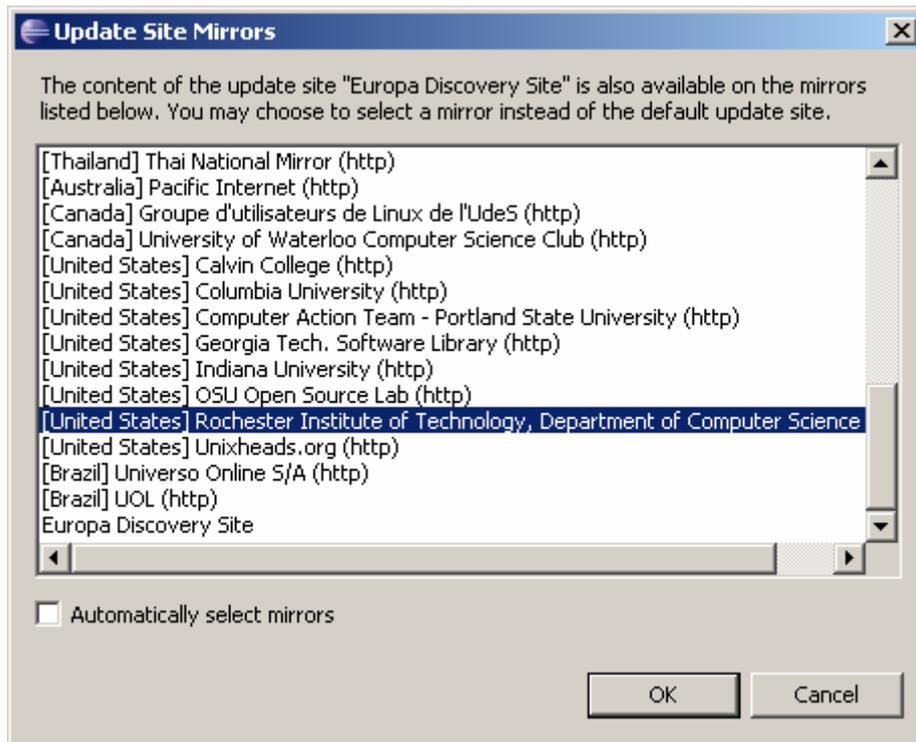
When you click in the option then, Eclipse will init an assistant to install new features.



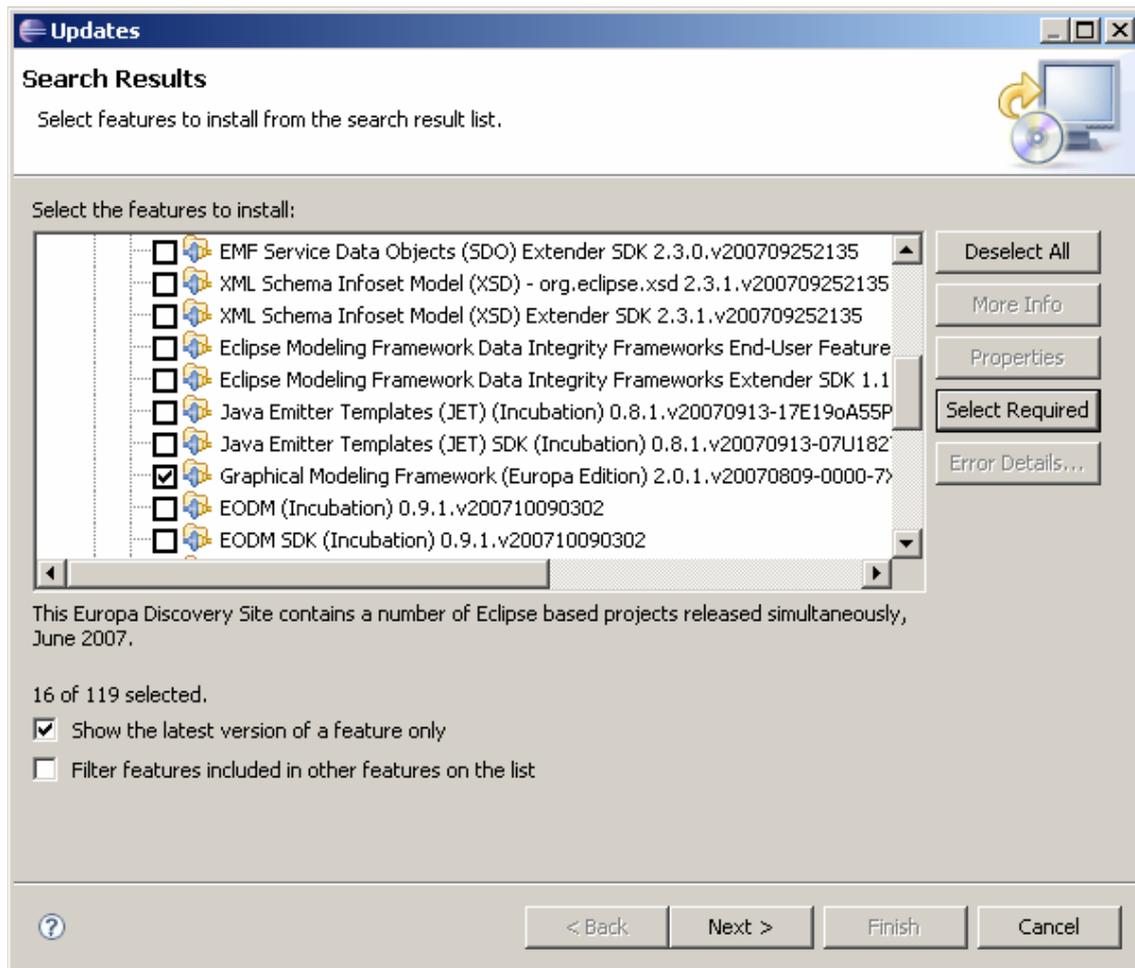
Select Search for new features to install and click in the button next



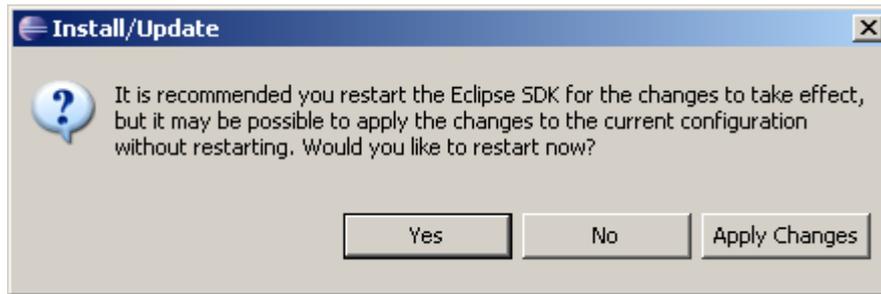
Select Europe Discovery Site and click in the button finish showing a list of server to search latest release of GMF



Select your favourite server and select the feature:



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

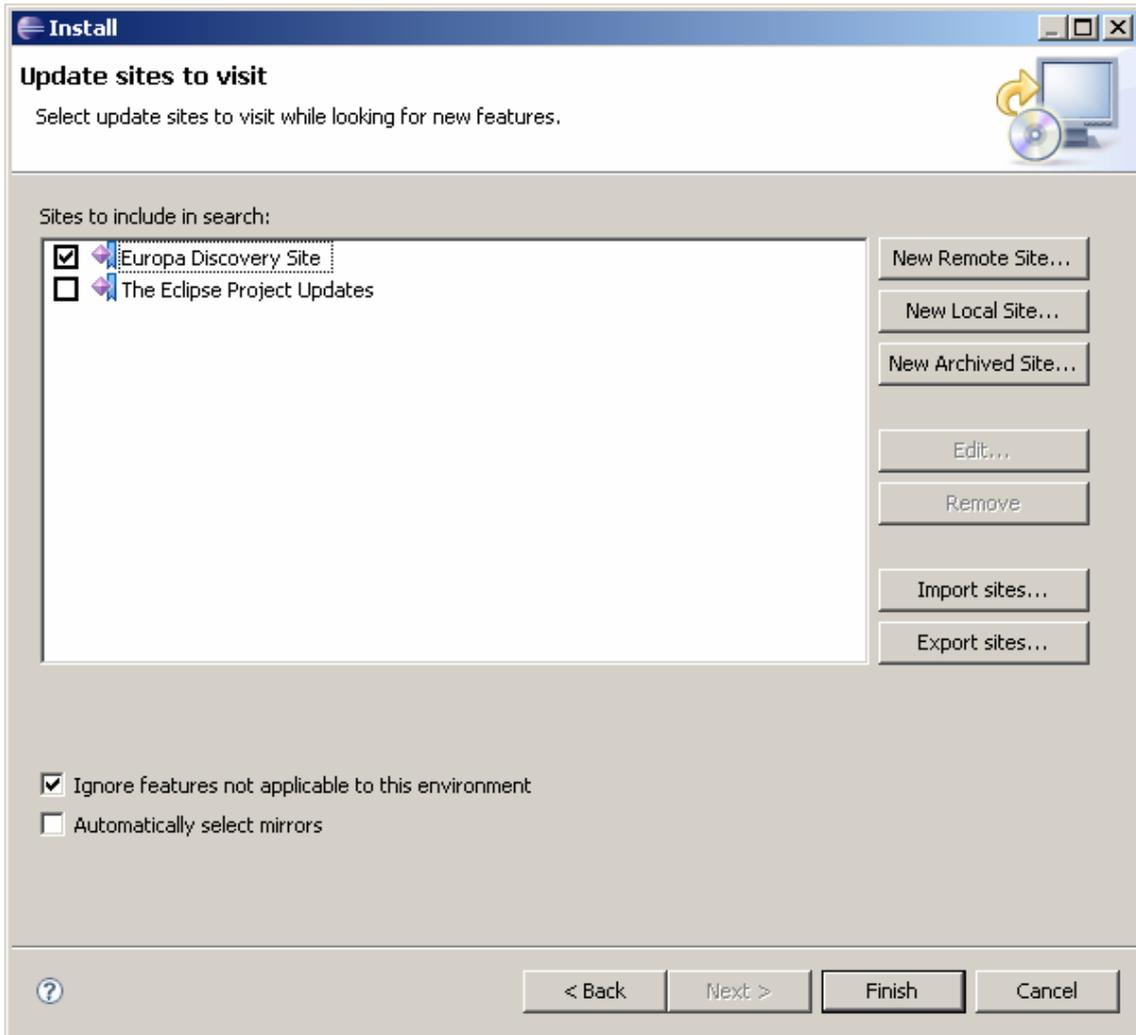


15.6.2.3.- OAW, Open Architecture Ware

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.

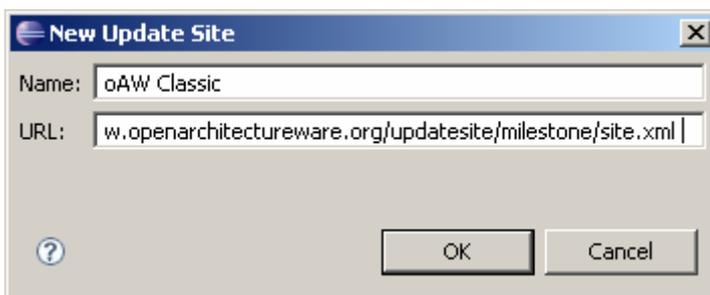


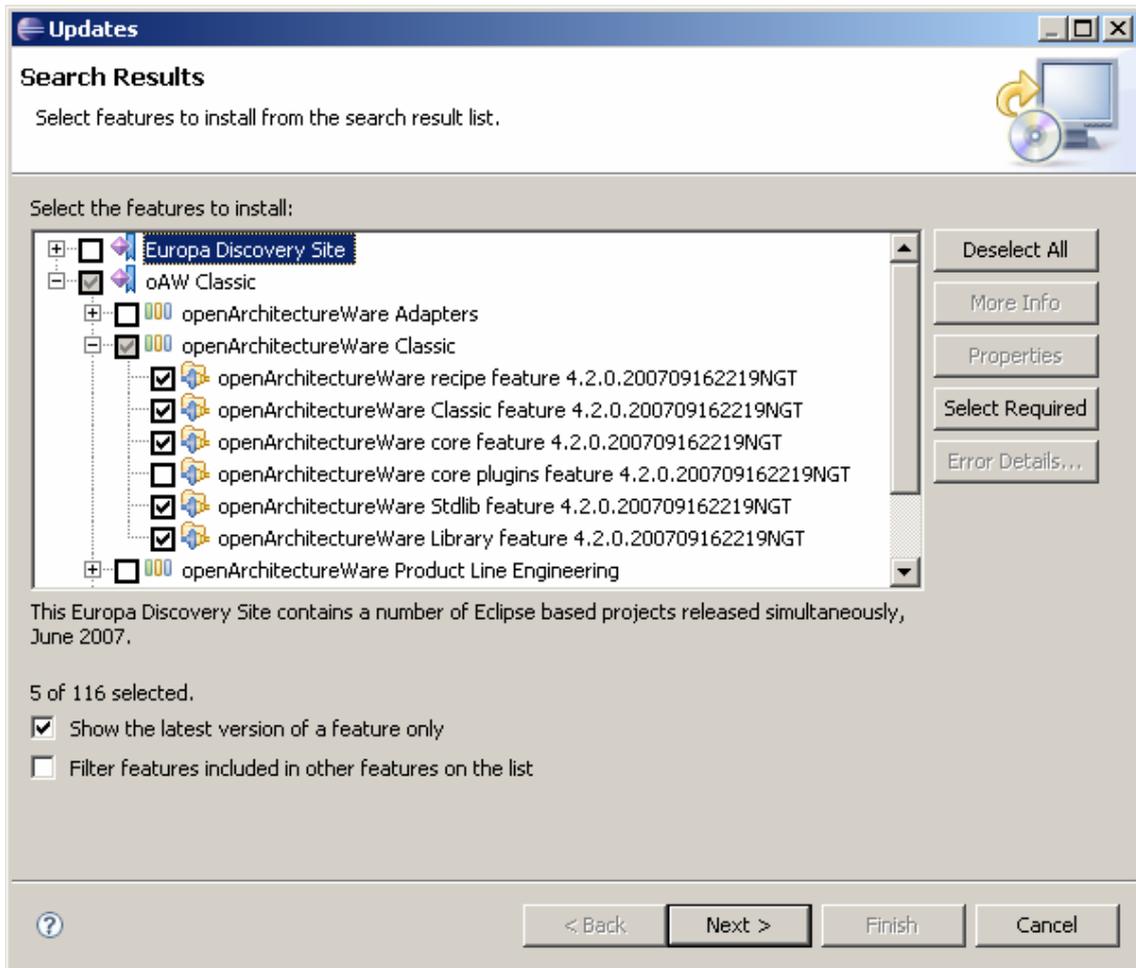
In this case, OWA should be installed with a different way in relation to GMF.



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

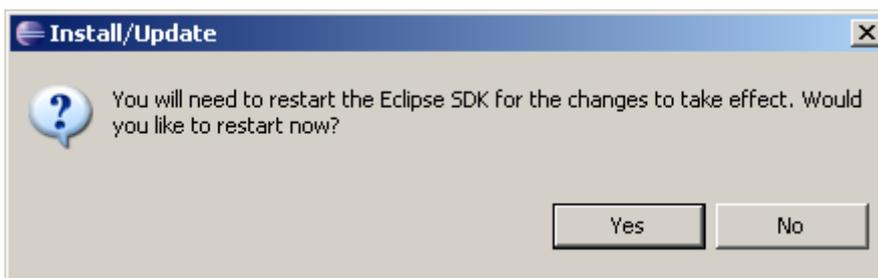
<http://www.openarchitectureware.org/updatesite/milestone/site.xml>





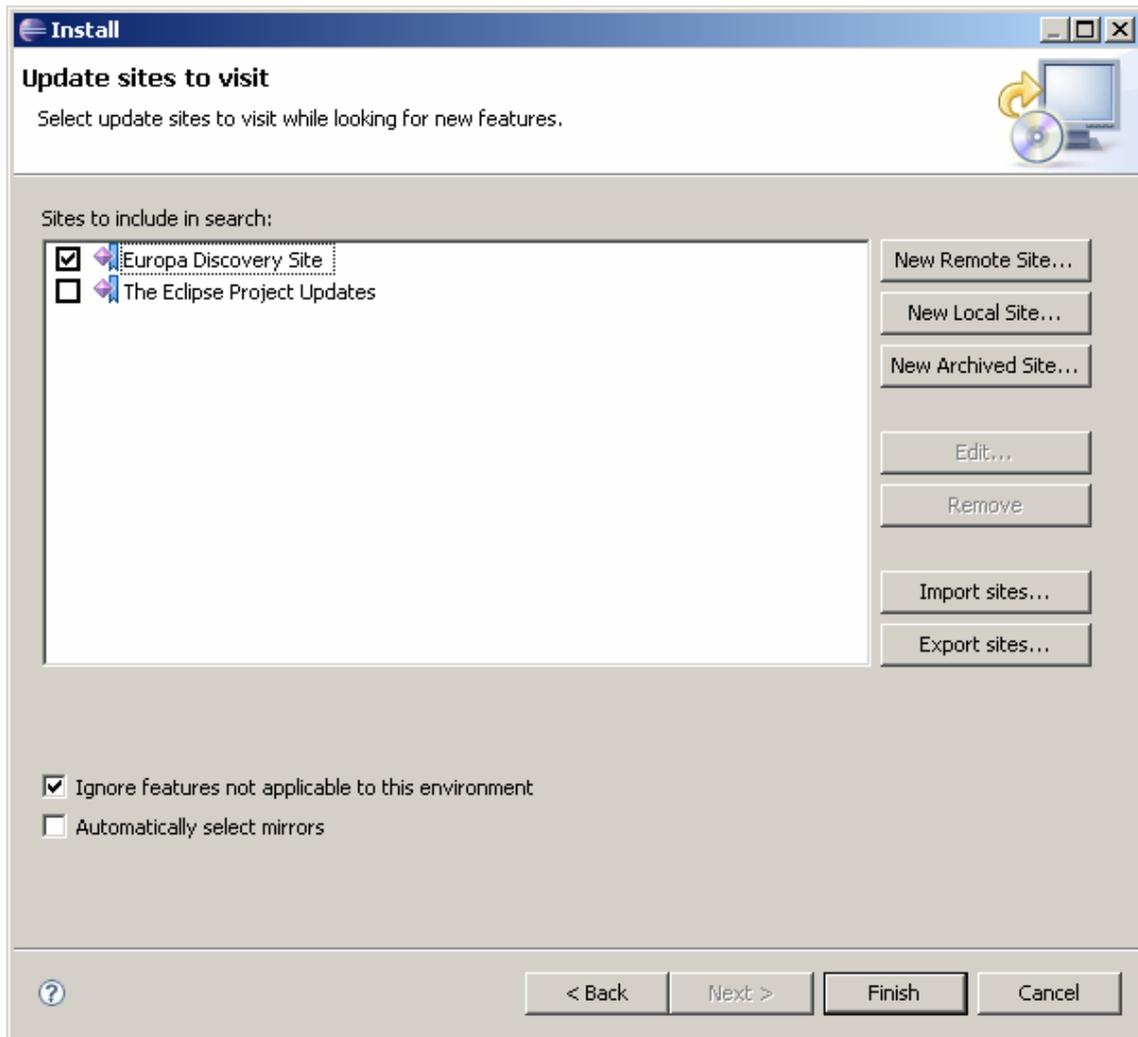
Expand the node and install the "oAW Classic feature" (you need to expand the oAW Classic one step further, to see the oAW Classic feature). When you click on select required, there will be two more features selected.

When you finish, restart Eclipse.



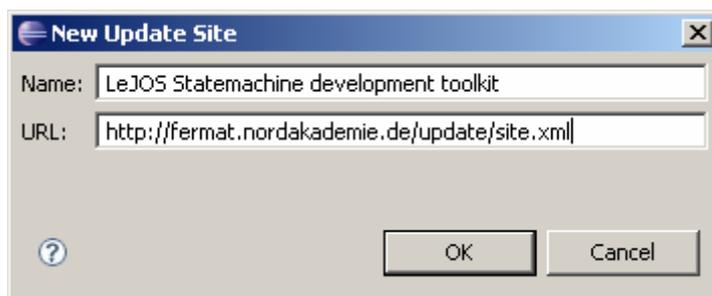
15.6.2.4.- LeJOS Statemachine development toolkit

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

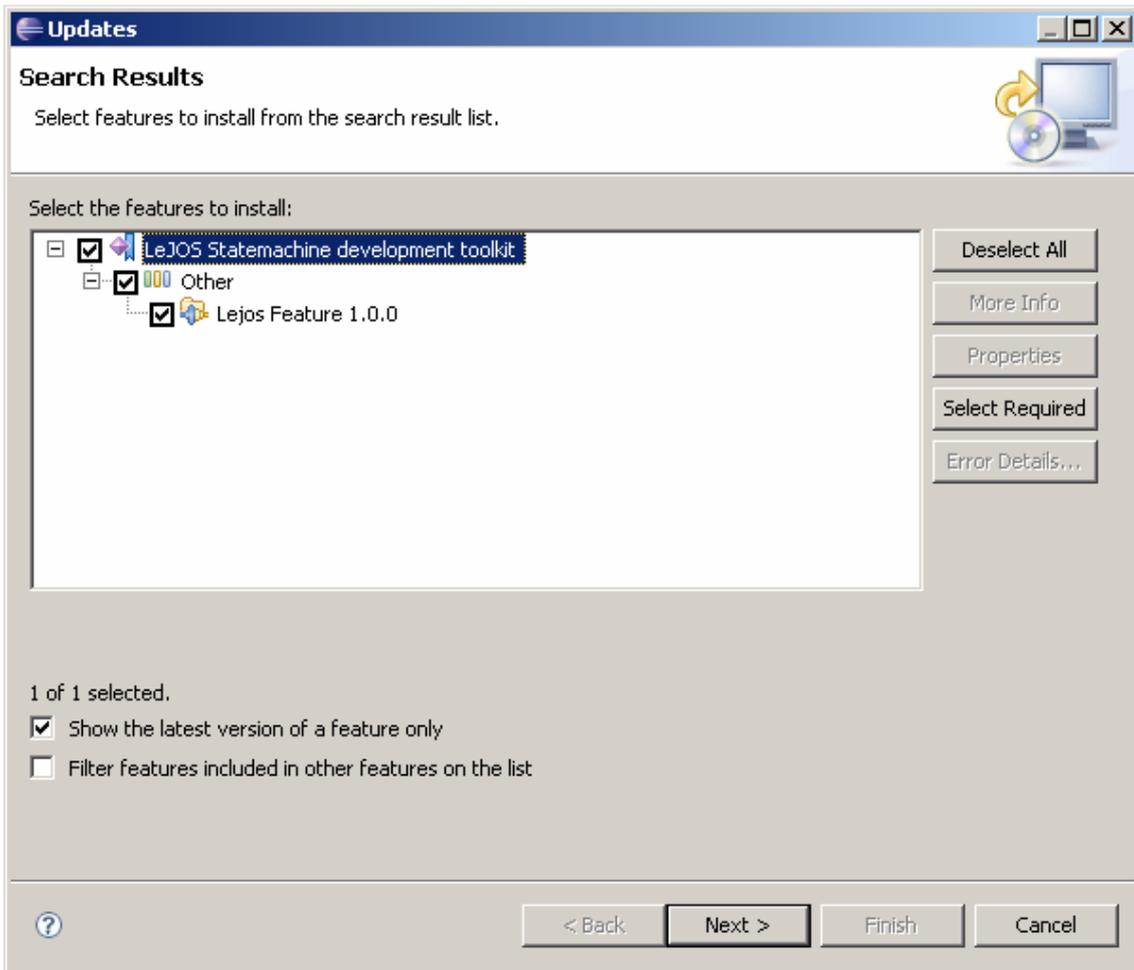


When you are in this window, click in "New remote site" button and edit an empty window with the following URL:

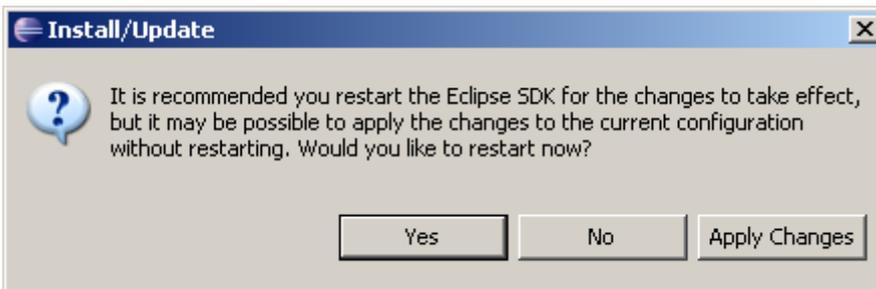
<http://fermat.nordakademie.de/update/site.xml>



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



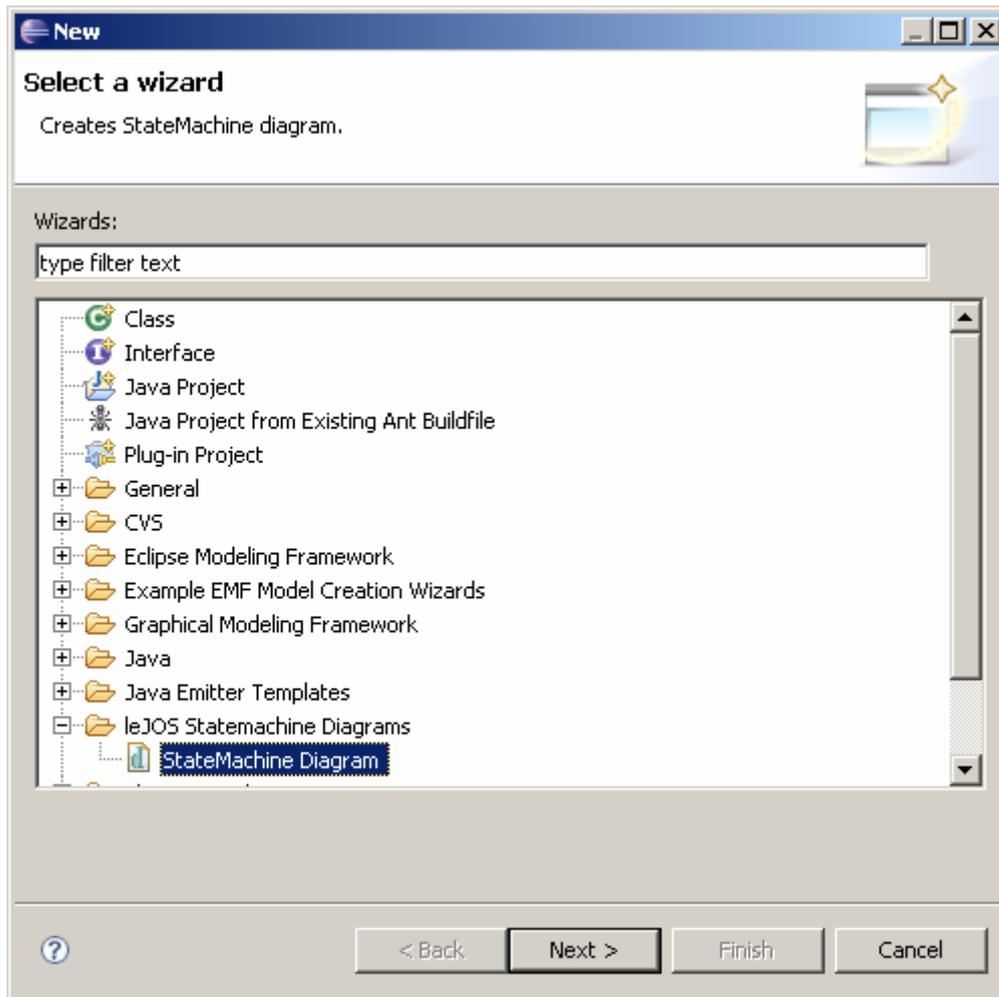
Click in "Next" button to install the toolkit. When you finish, restart the IDE.



15.6.3.- Creating the first project with the toolkit

15.6.3.1.- Create a new project

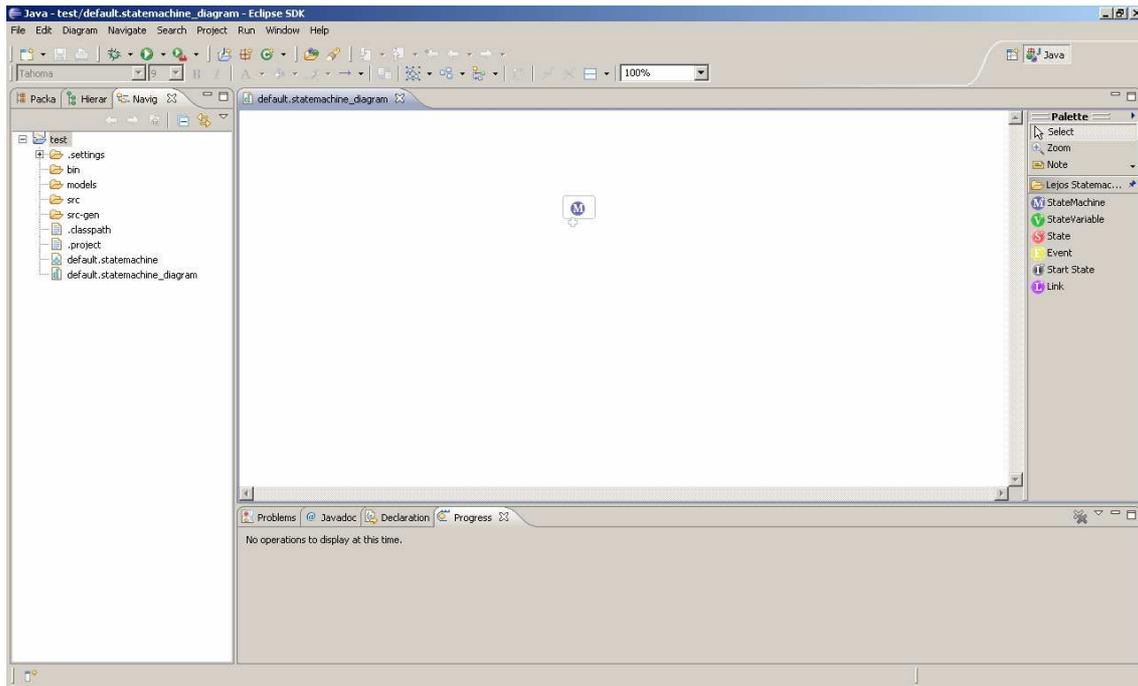
Init Eclipse and create a new project, select the option LeJOS statemachine project.



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.

15.6.4.- 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>

15.7.- Summary

In this chapter you learnt some tools developed by leJOS Developer kit and leJOS community to help you in your leJOS projects.

16.- Robotics projects

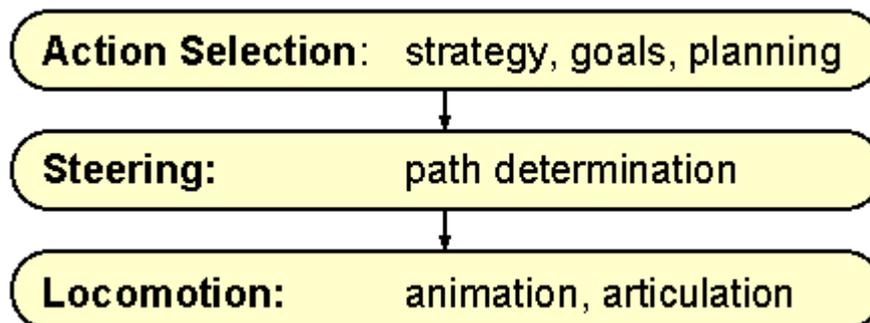
16.1.- Introduction

In this chapter you will learn some concepts about robotics.

16.2.- Steering behaviors

16.2.1.- 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. 'Alignment' 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.



16.2.2.- Behaviors

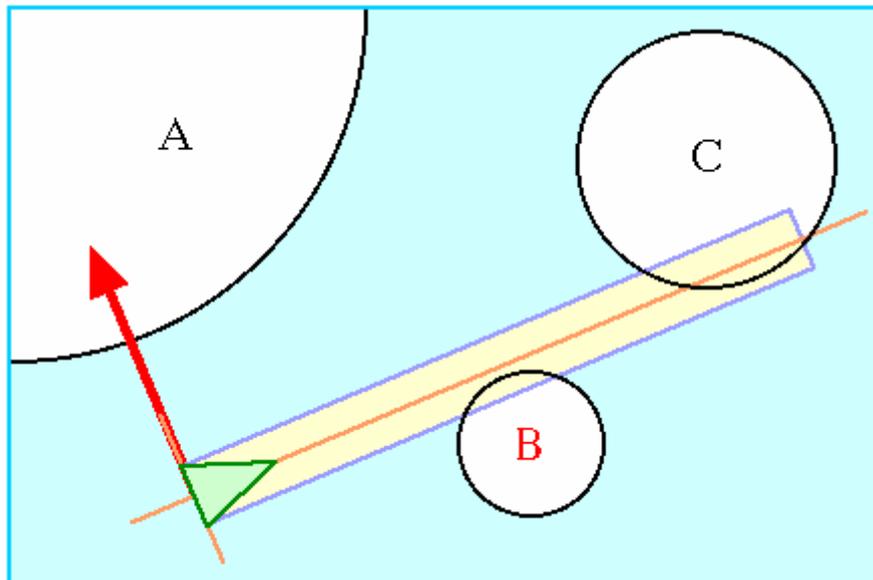
The behaviors researched by Craig Reinolds and Robin Green are:

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>

16.2.3.- Obstacle Avoidance

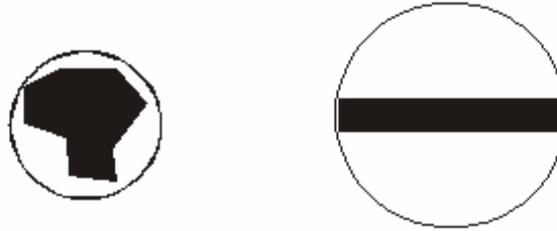


Application

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 giving 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.

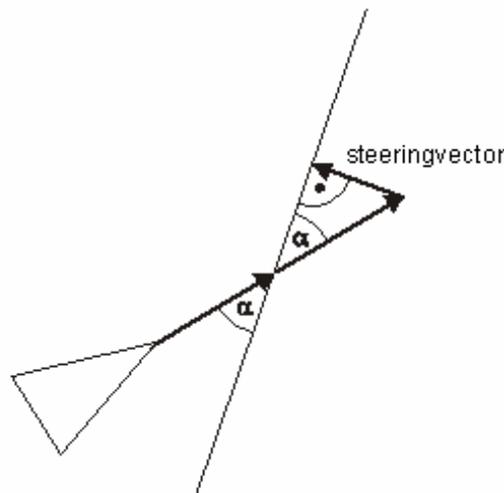


Since the disadvantages of using circles as placeholders clearly outweigh the advantages, the choice was made to use polygons instead. The advantage being that elements of real life, like chairs or tables for instance, can be better approximated. The results achieved in two-dimensional space can be applied 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.



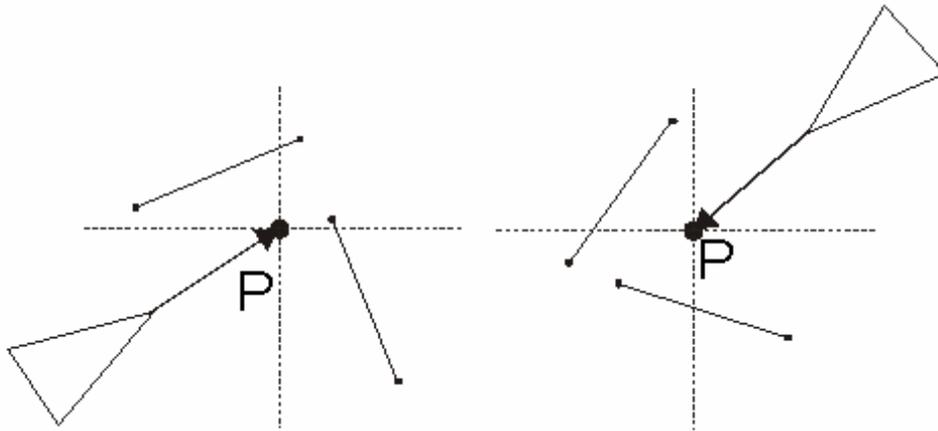
Optimization using "exclusion tests"

If there are obstacles within close vicinity of the vehicle it 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.

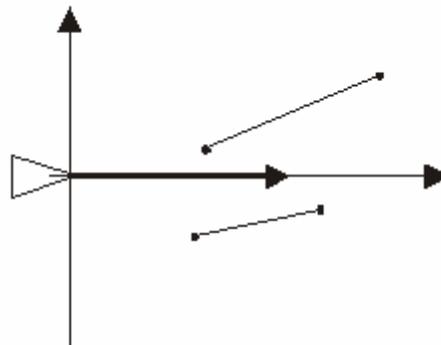
Test 1:

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).



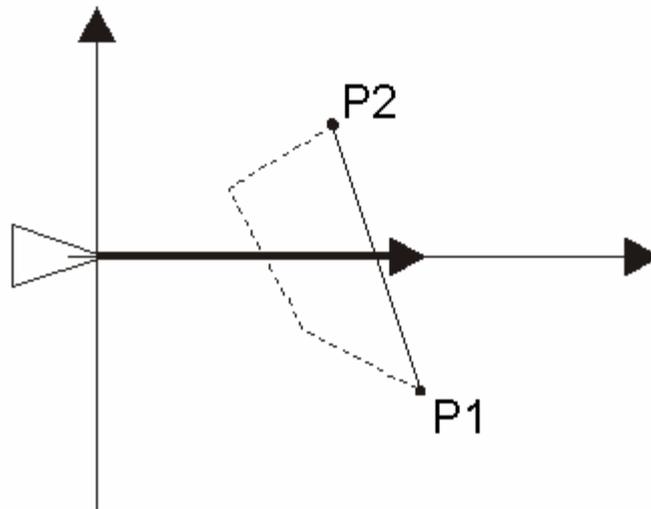
Test 2:

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.



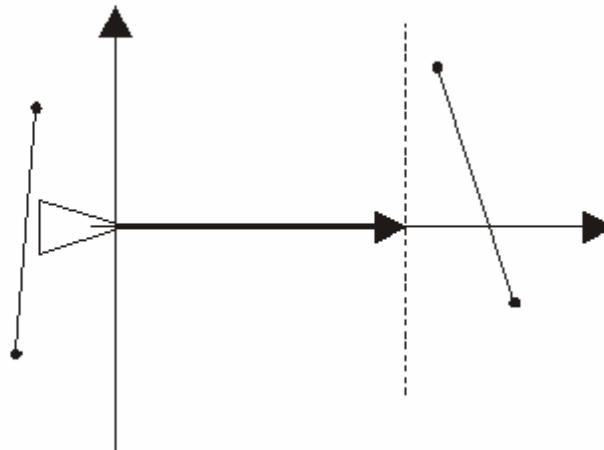
Test 3:

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 x_1 and y_1 define the coordinates of the first edge point. The slope a is

$$a = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

calculated using . If this is inserted into the first formula, solved for

$$x = \frac{(-y_1) \cdot (x_2 - x_1)}{(y_2 - y_1)} + x_1$$

x and y is set to zero, the resulting formula will be

The resulting value for x is the distance of the intersection with the obstacle from 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.

16.3.- Parallel Architectures

16.3.1.- 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.

16.3.2.- 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.

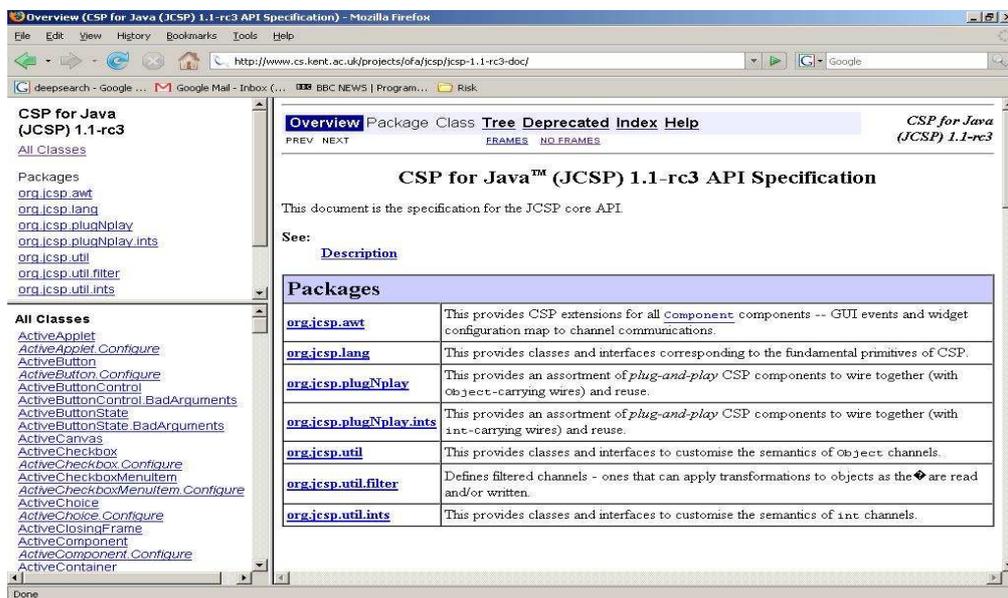
The JCSP library is quite large and thus not suitable for a small operating

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>

16.3.3.- 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.

16.3.4.- 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 CSPProcess interface. This interface looks like this

```
public interface CSPProcess {

    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.

16.3.5.- 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;
    }
}

```

16.3.6.- Line follower robot with JCSP re & leJOS

16.3.6.1.- 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());`

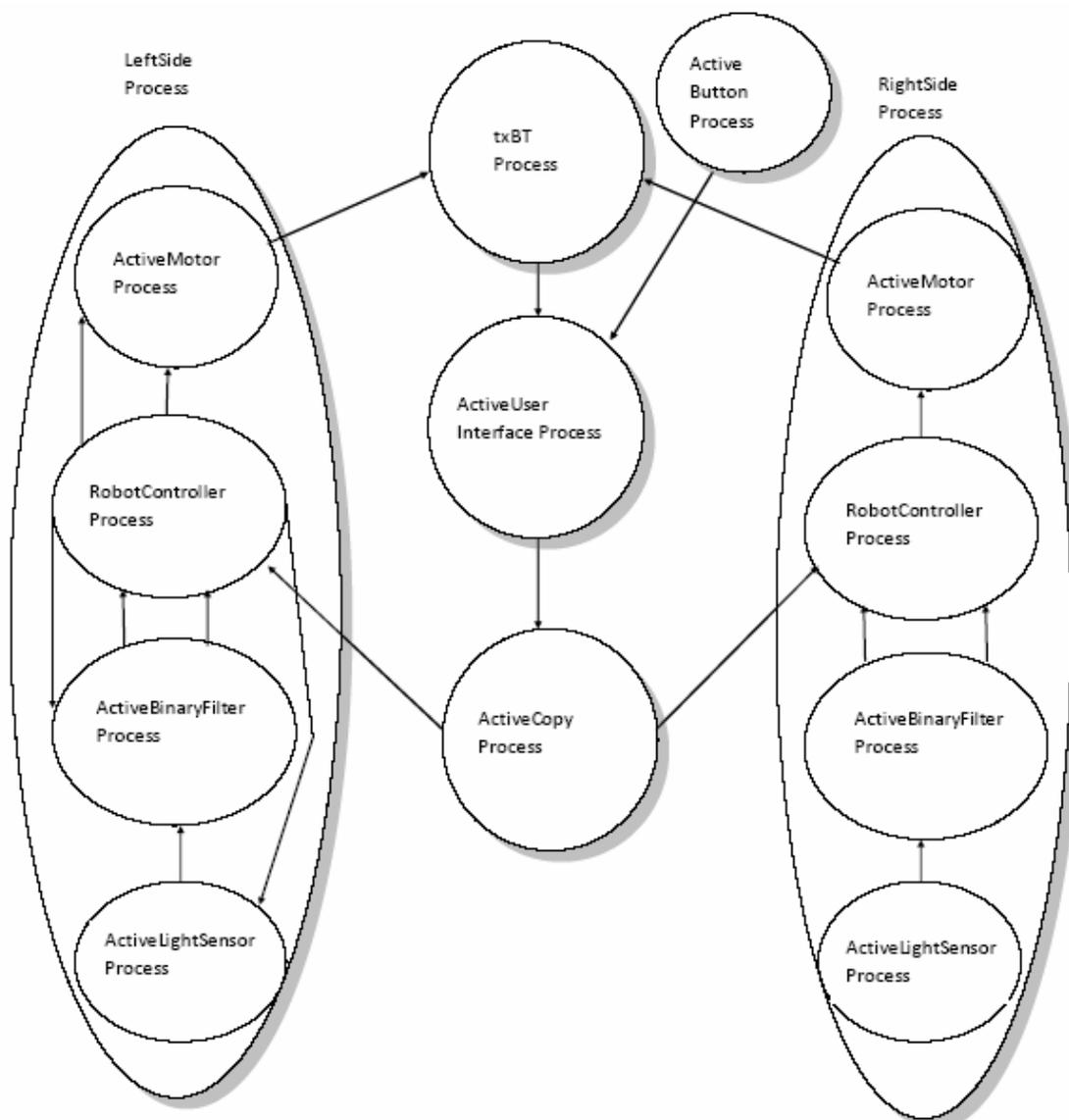
```
ActiveUserInterface ui = new ActiveUserInterface (keyPress.in(), display.out(),
buttonPress.out());
```

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();

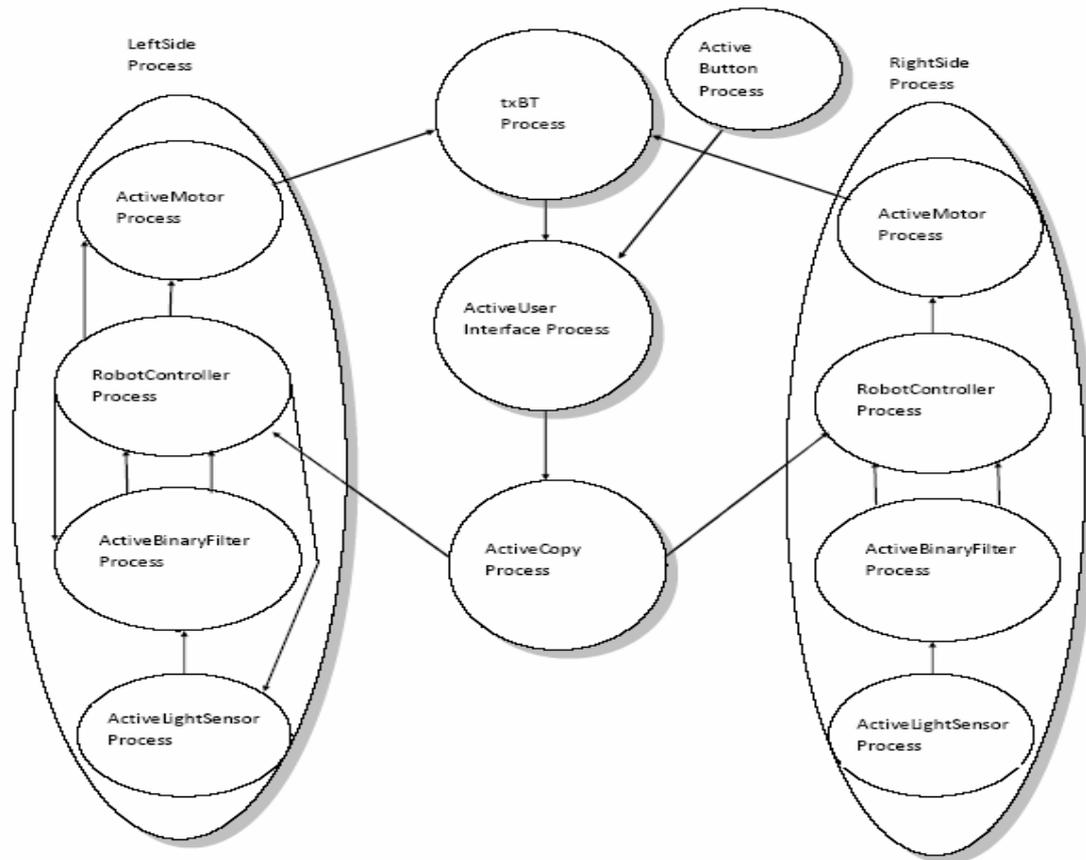
par.addProcess(ui);
par.addProcess(screen);
par.run();
```

16.3.6.2.- 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



16.3.6.3.- 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.pluginplay.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 CSPProcess {
    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) {
rotation
            rotate = buttonPress.read(); // read speed of wheel

            //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, ChannelInputInt
buttonPress, int id) {
        //public RobotSide( SensorPort s, Motor m, ChannelInputInt
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();  
    }  
}
```

17.- FAQ

17.1.- How to reinstall Lego Firmware

17.1.1.- 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.

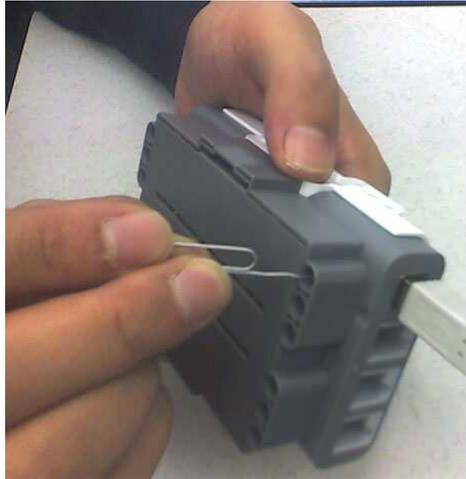
17.1.2.- Download latest Lego firmware

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

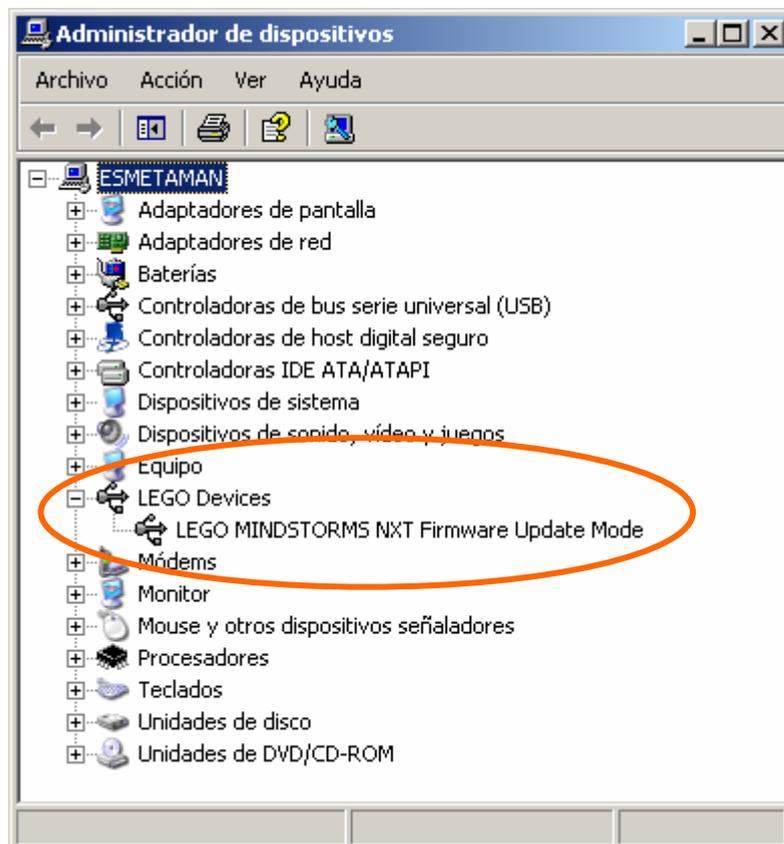
Description	Version	Post Date	PC
<p>Mac OS 10.5 (Leopard) Firmware Fix</p> <p>Description This software addresses an issue on Mac OS 10.5 (Leopard) that prevents updating the Firmware on the NXT.</p> <p><i>Note: Do not attempt to update the Firmware on the NXT from a computer running Mac OS 10.5 without installing this software first.</i></p> <p>If your NXT is clicking when you insert batteries, push the hardware reset button for five seconds before you insert the USB cable. This will ensure that the brick initializes correctly for the firmware download. The hardware reset button is located within the LEGO TECHNIC hole below the USB connector on the NXT brick. If your NXT brick is not clicking when you insert batteries just go through the normal firmware download process as described within the manual when the patch is installed.</p> <p>Instructions Click on the file to save it to your hard drive. Doubleclick the downloaded zip archive and install the legodriver.pkg. Follow the instructions on the screen.</p> <p>System Requirements Mac OS X 10.5 (Leopard) LEGO MINDSTORMS NXT Software version 1.0/1.1 or MINDSTORMS NXT Education Software 1.0/1.1</p>	1.0	07/12/2007	<p>PC: Mac firmware fix not available for PC</p> <p>MAC 479 kb</p>

17.1.3.- 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:

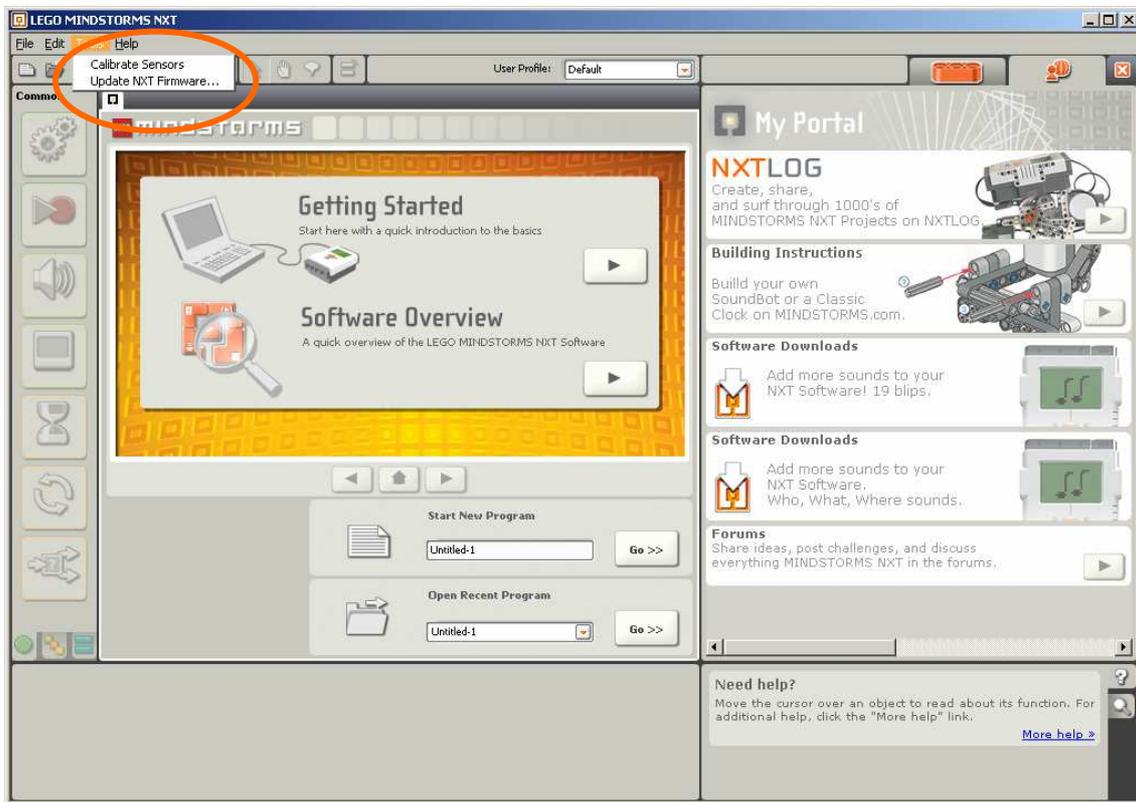


17.1.4.- Reinstall Lego firmware

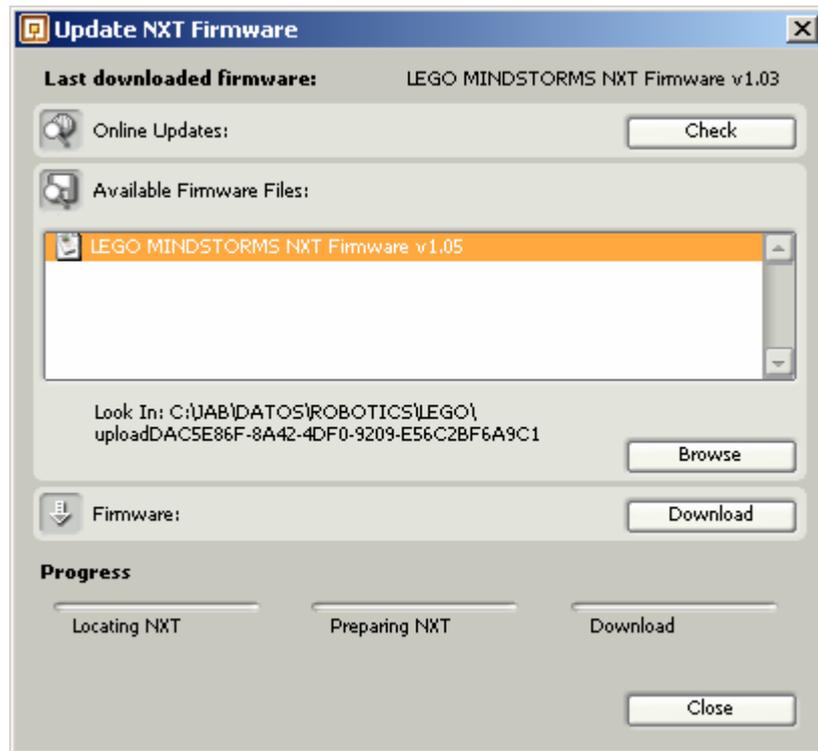
Use Lego Software that you received with your NXT Kit to download Lego Firmware.



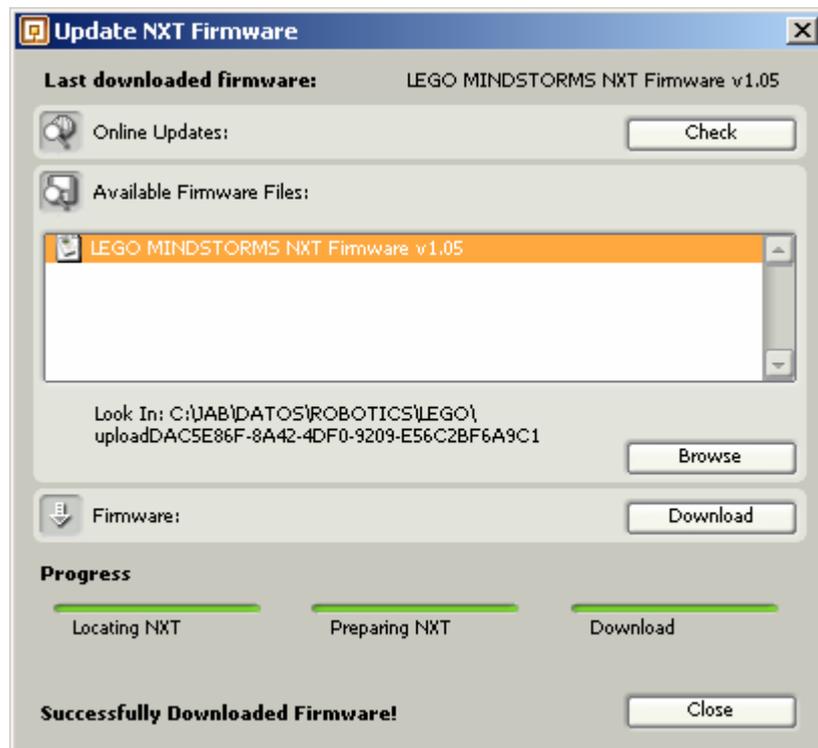
Execute the software and search the option Update NXT Firmware in Tools tab.



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.



17.2.- Using Tortoise SVN to collaborate in leJOS project

17.2.1.- Introduction

17.2.1.1.- 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

17.2.1.2.- 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/>

17.2.1.3.- 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/>

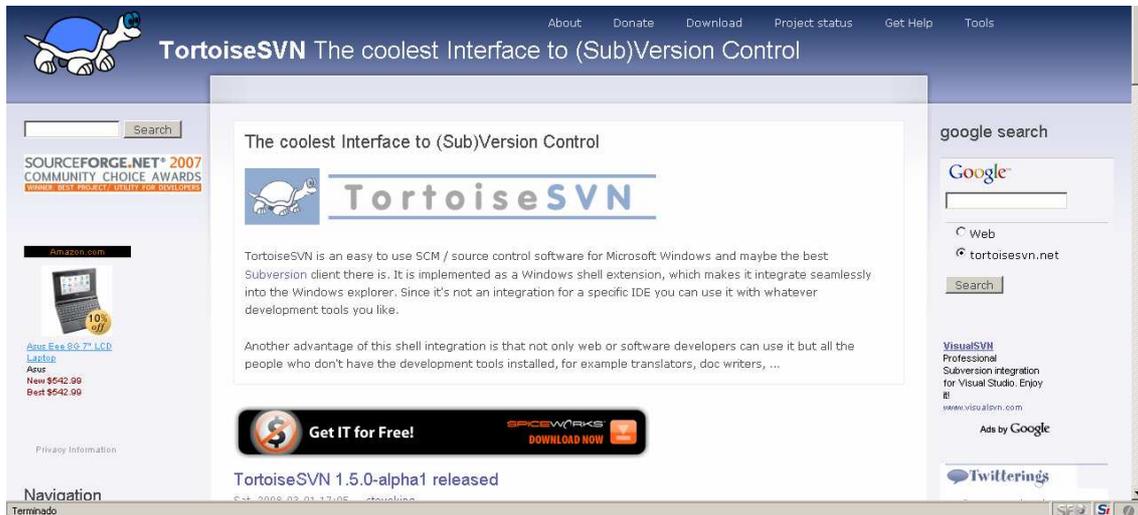
<http://www.polarion.org/index.php?page=overview&project=subversive>

17.2.2.- Installing Tortoise SVN

17.2.2.1.- 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/>



17.2.2.2.- Install Tortoise

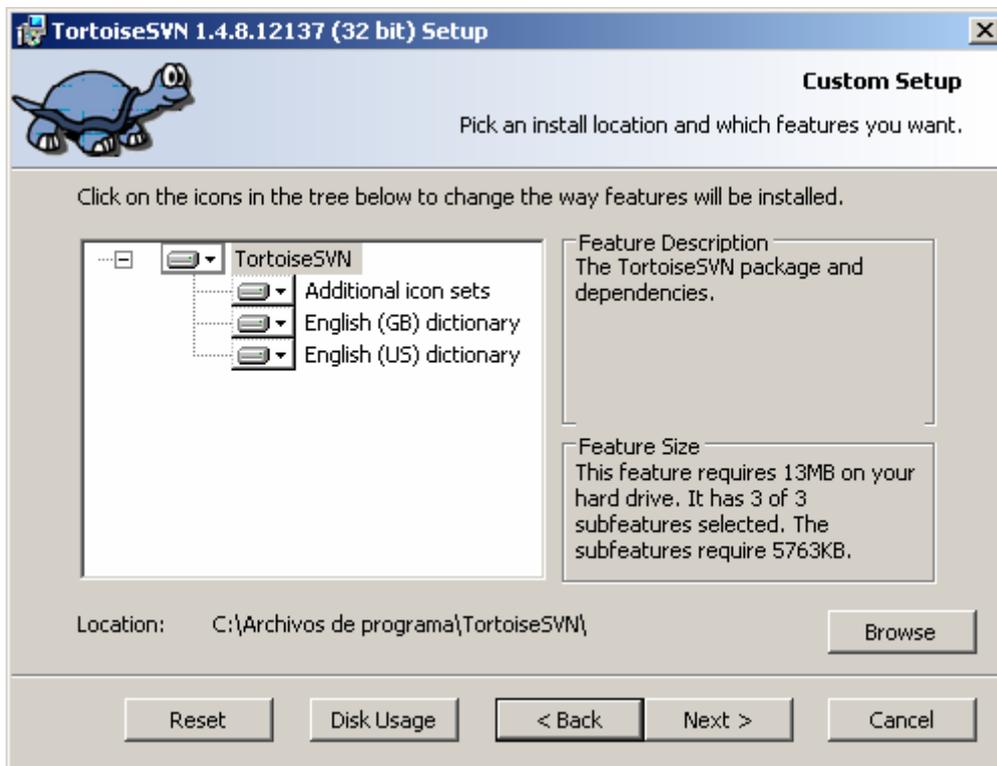
When you have downloaded the client, execute it and follow all steps in the installation.



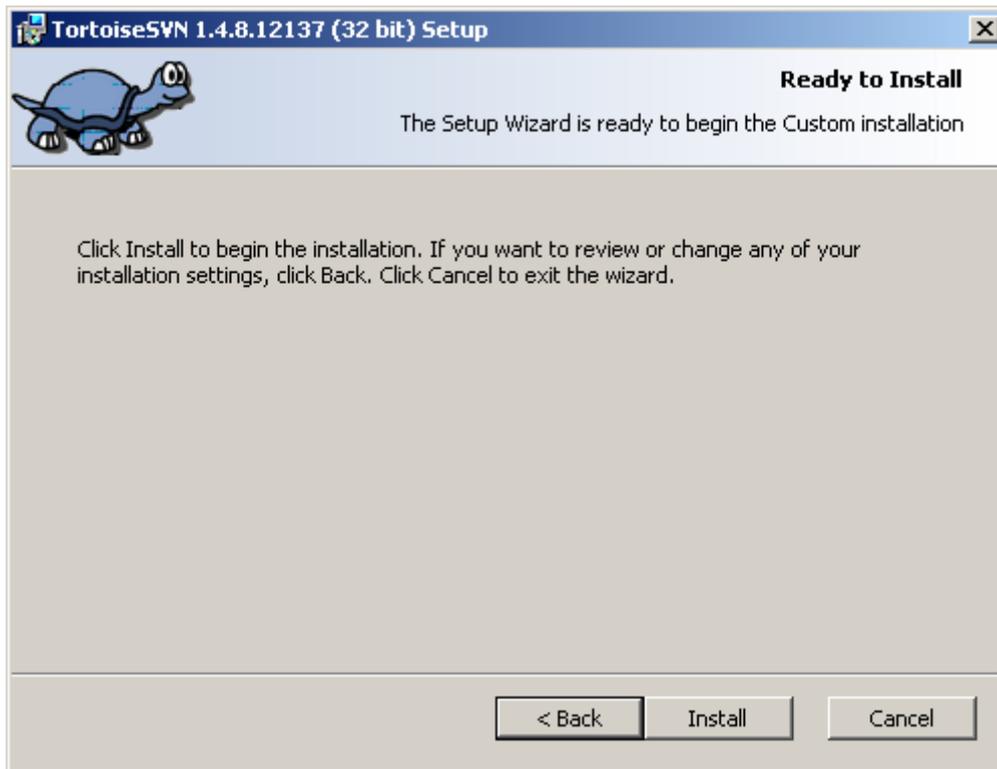
Accept the terms:



Select all components:



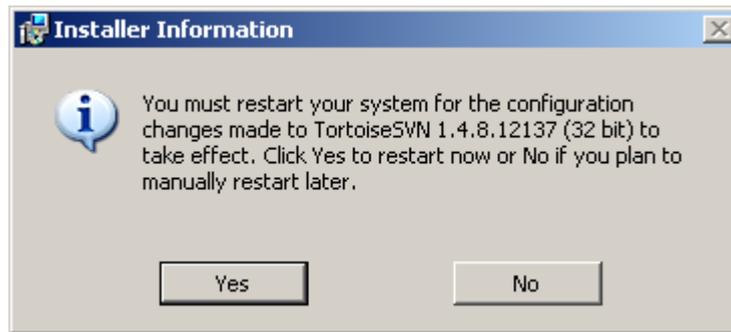
Install the components:



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.



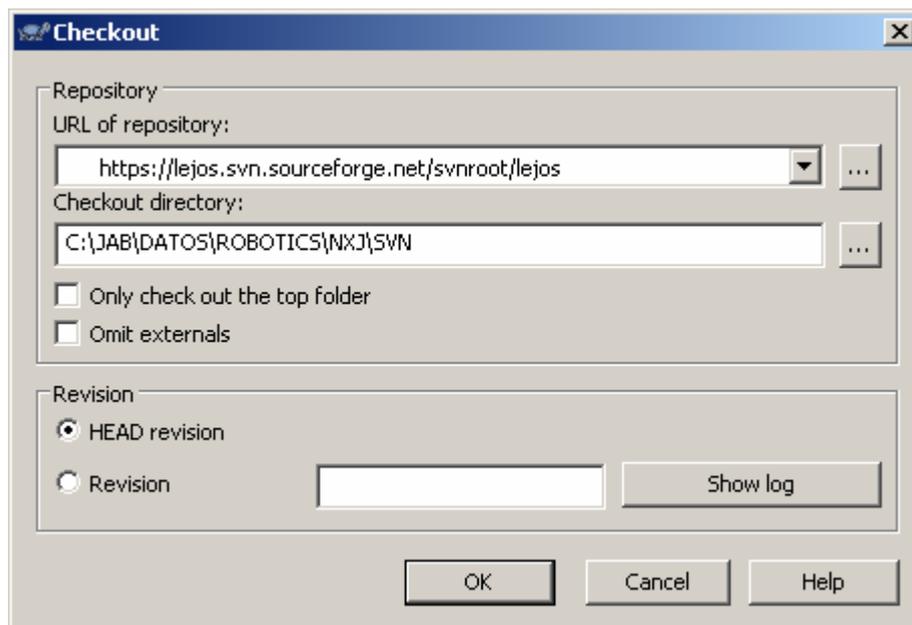
17.2.3.- Downloading LeJOS Repository

17.2.3.1.- 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:



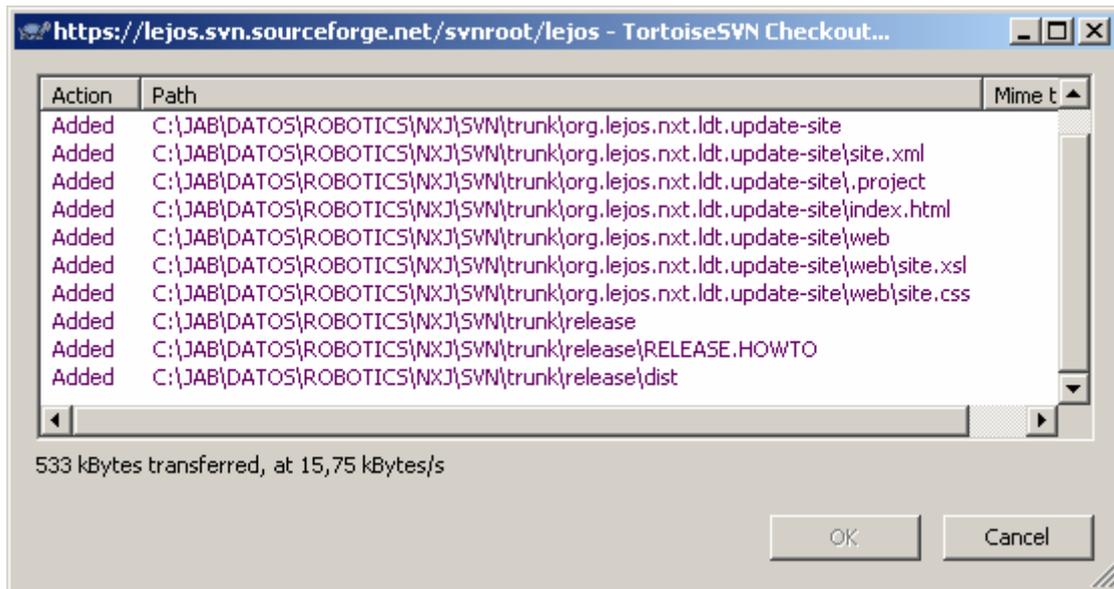
If you want to download all source code, click in the option SVN Checkout:



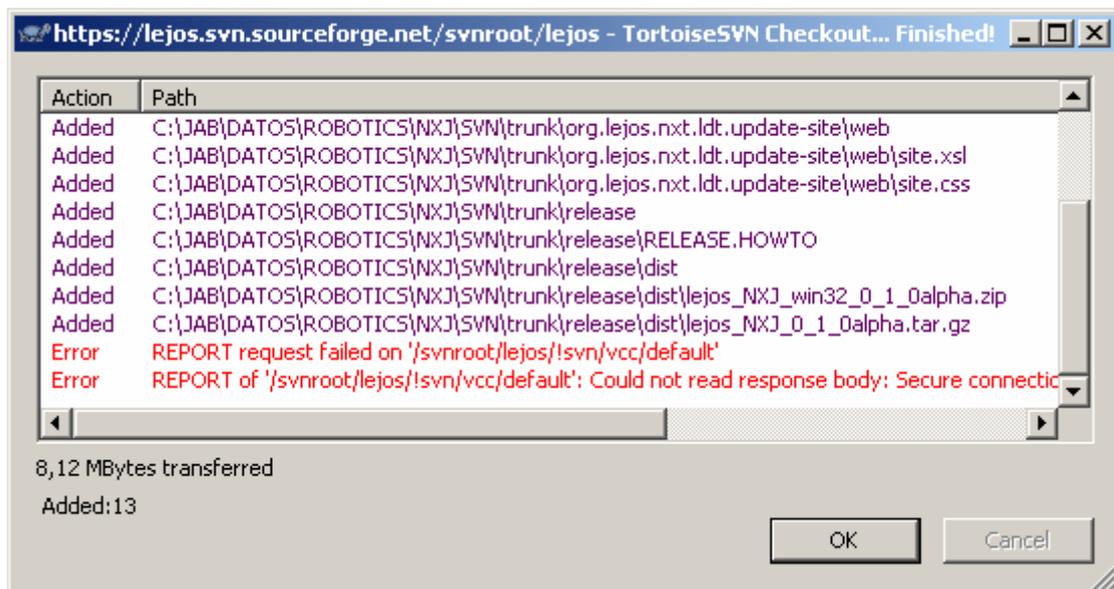
The repository URL is:

<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.



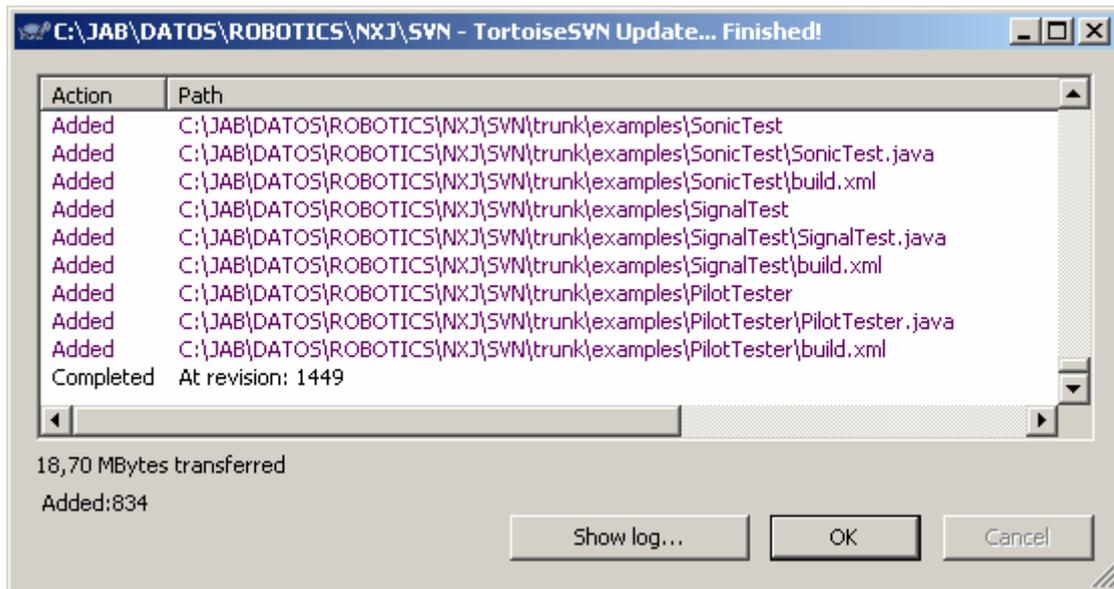
If you lost your internet connection, Tortoise has an option to continue with your download, SVN Update.



Click in SVN Update:



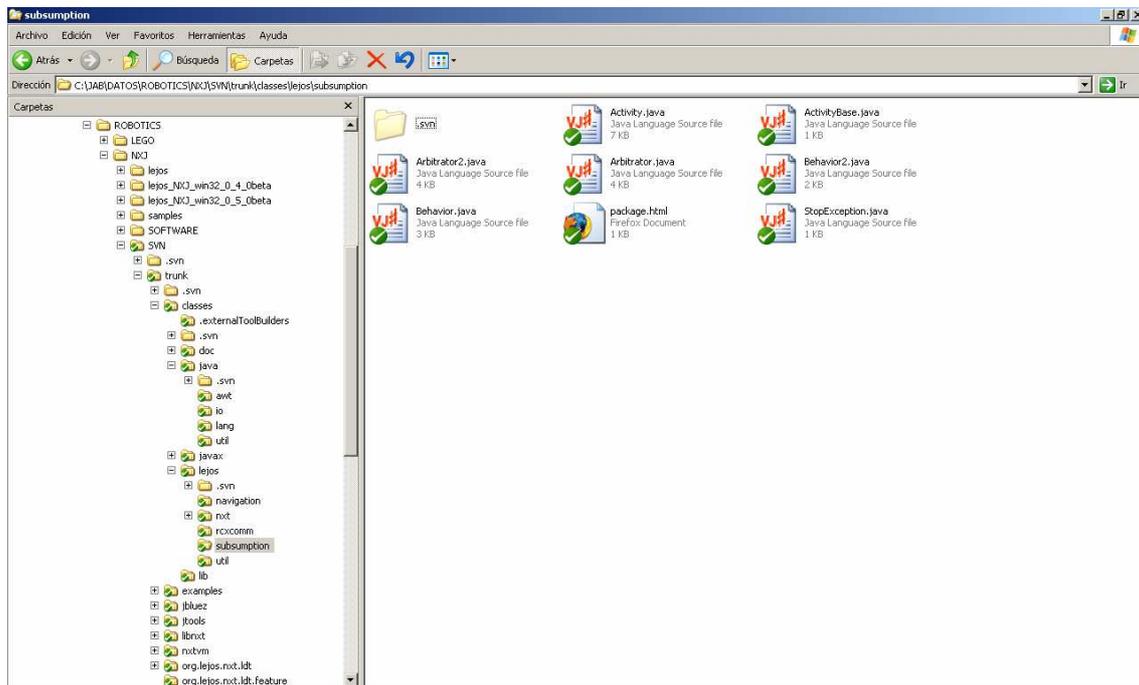
When you finish with your download, you will see this one:



17.2.4.- Using Tortoise in LeJOS

17.2.4.1.- Browsing the repository

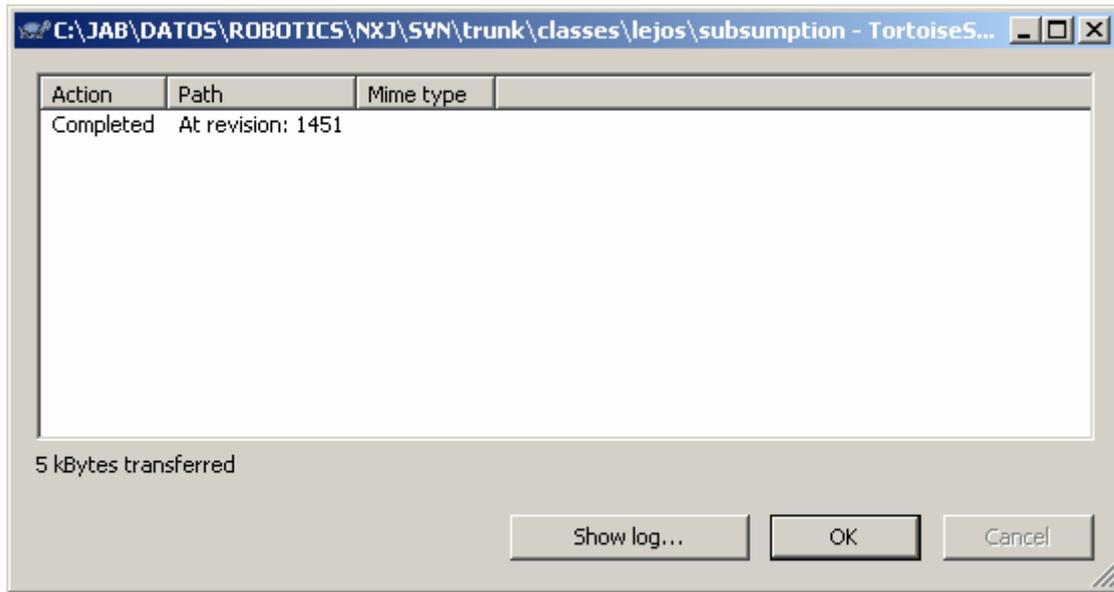
When you have downloaded LeJOS project computer then you will see your Windows Explorer with a new icons:



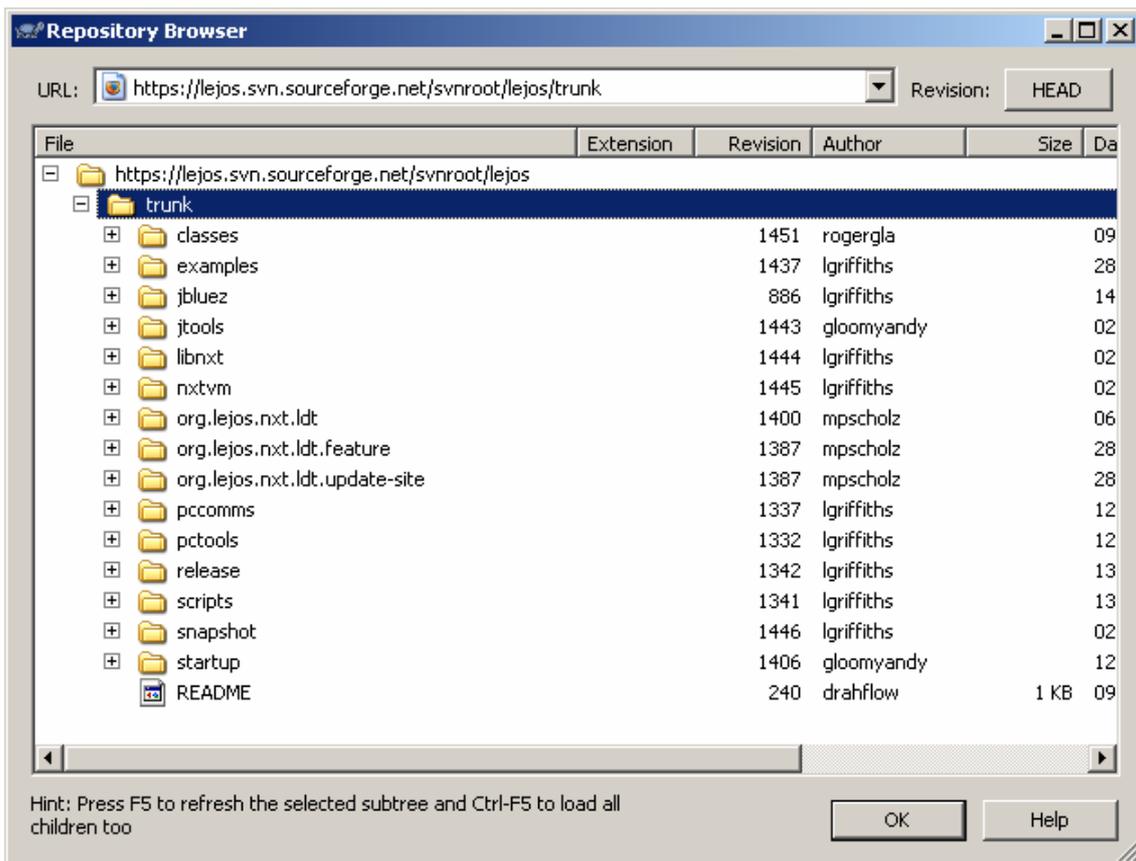
If in any moment you need to update a part of you project, simply select the folder, in this case the folder named "Subsumption" and click in the option "SVN Update"



Then you will update the folder that you have chosen.

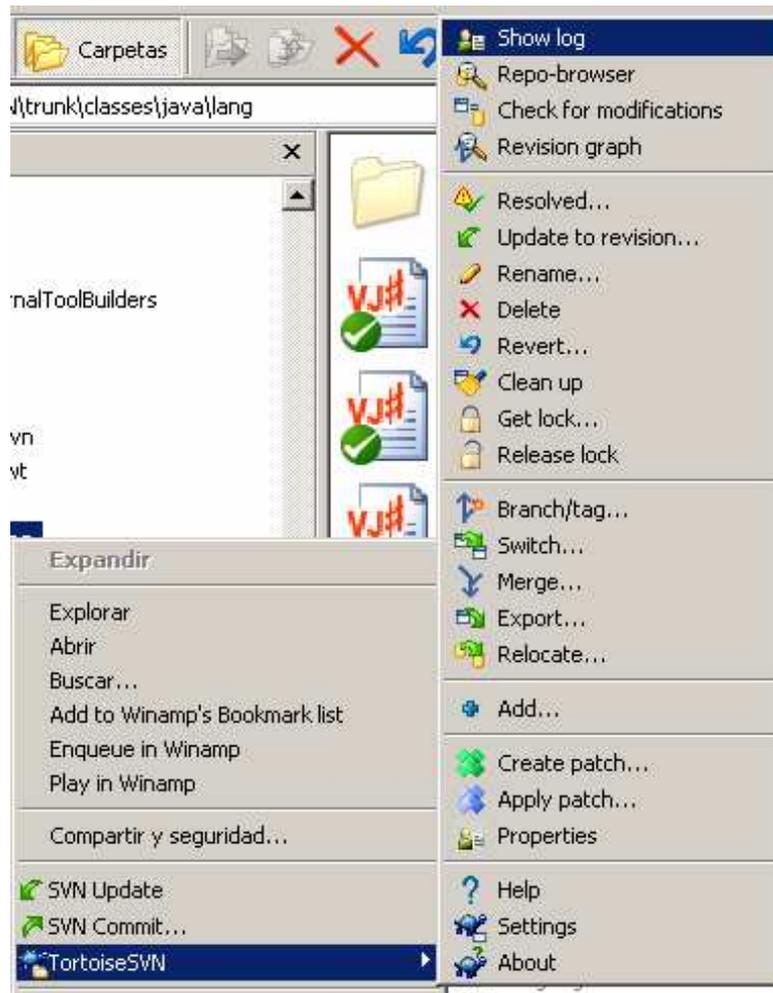


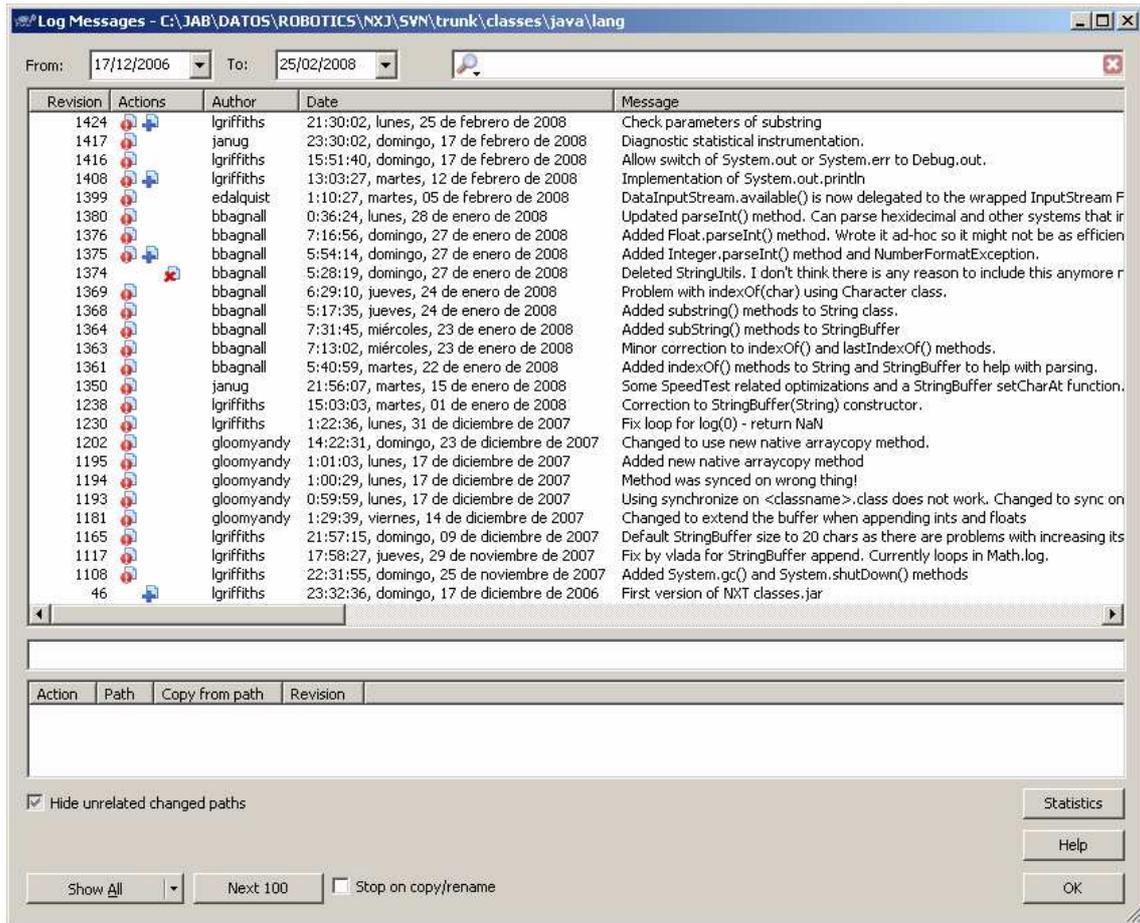
If you need to see the repository in the server, use the option "Repo-browser"



17.2.4.2.- Track the log

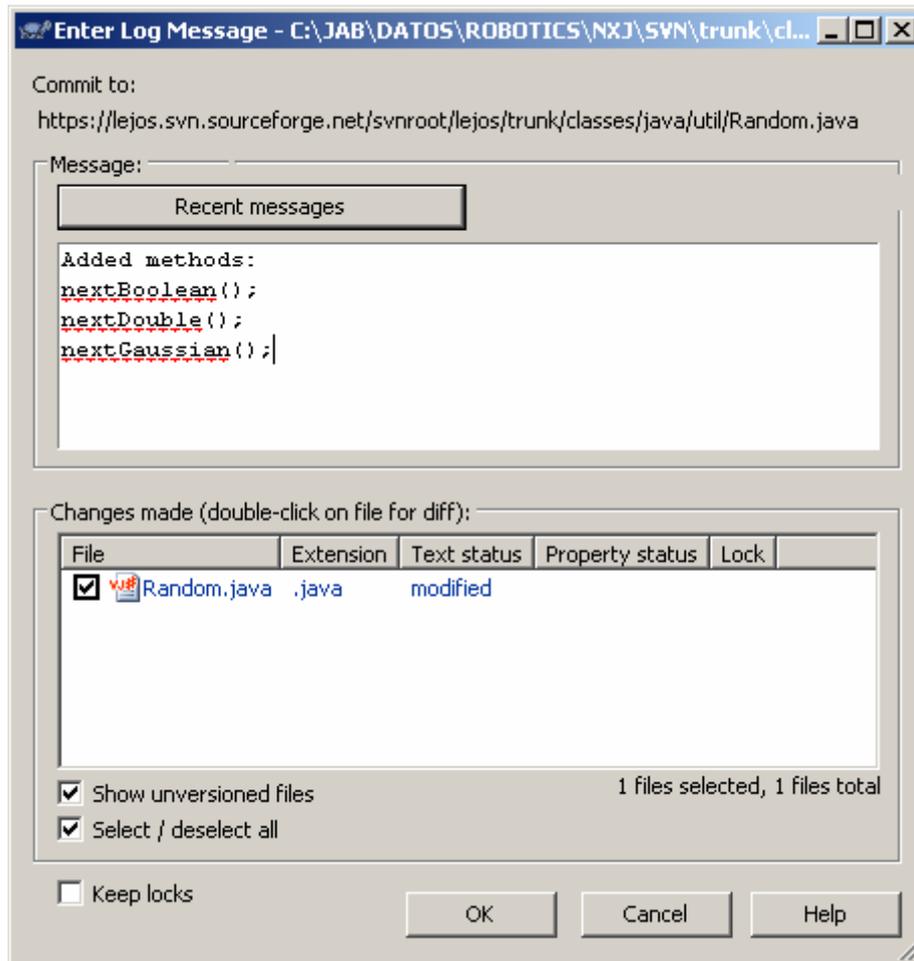
Using subversion technology, it is possible to see the Log. To learn the concept, use a classic package in LeJOS , java.lang. Select the folder lang and click in the option "See log"



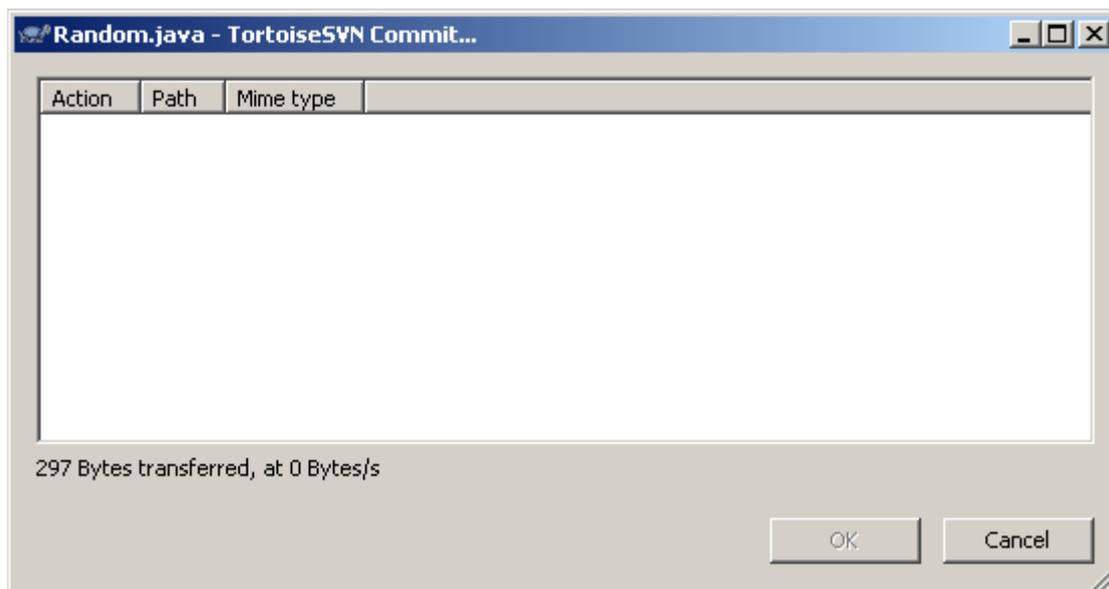


17.2.4.3.- Collaborate with new code

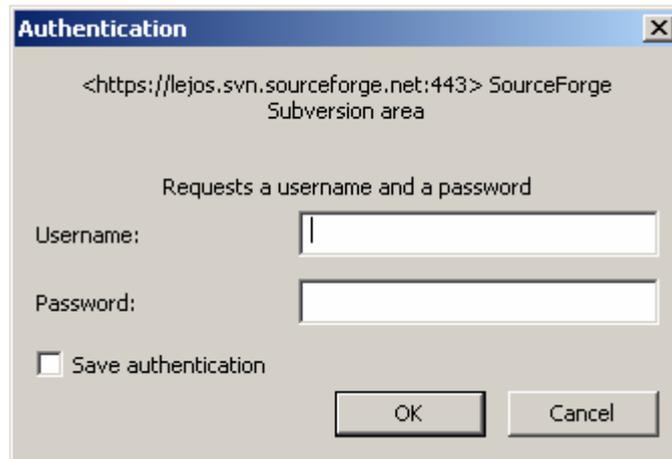
If you want to collaborate with the project then you can update your files in your local computer and click in the option "SVN Commit" then you will see the following window:



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



17.2.5.- How to be a new LeJOS Developer

17.2.5.1.- 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.

17.3.- How to use beta software from leJOS SVN?

If you are an active user then you know that leJOS project releases software every 4-6 months more or less. But in this period, the project suffers many changes and If you want to use latest works into your NXT projects, I recommend you that use the following technique.

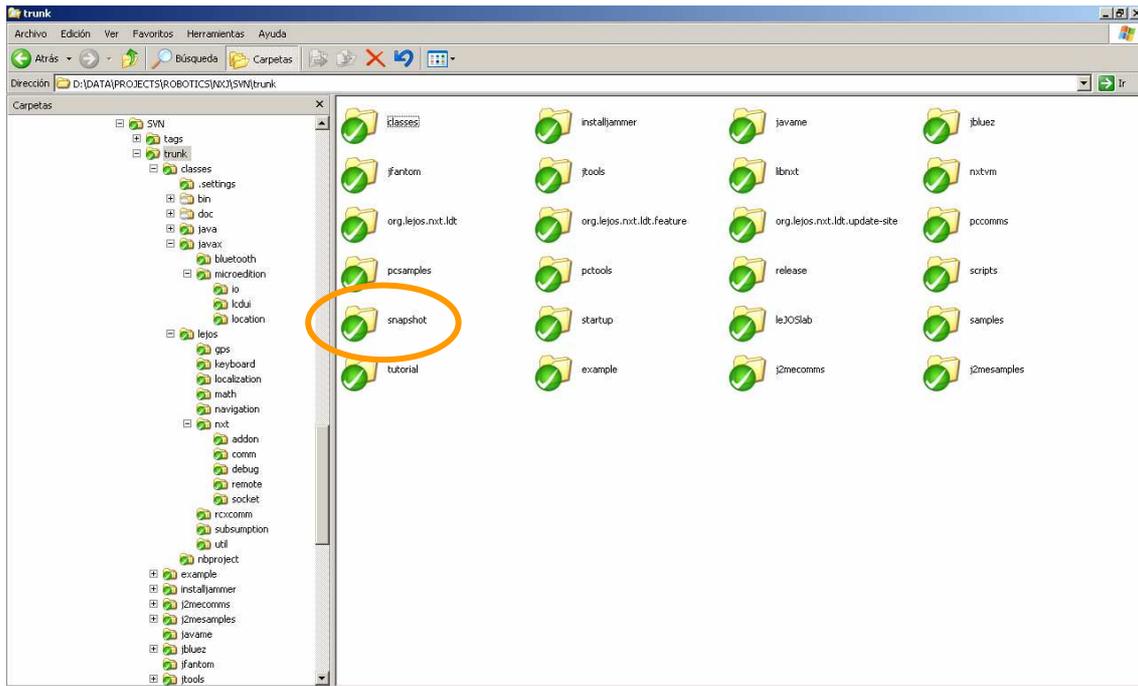
17.3.1.- Make a checkout from leJOS SVN

Use your favorite SVN client to make a checkout from the leJOS SVN.

The repository URL is:

<https://leJOS.svn.sourceforge.net/svnroot/leJOS>

Once you have a local copy of the project then find the folder snapshot:



The folder snapshot contains:

1. Latest firmware release
2. Latest menu release
3. Latest packages for your projects:
 - a. Classes.jar
 - b. Jtools.jar
 - c. Pcomms.jar
 - d. Pctools.jar

17.3.2.- Copy snapshot to your leJOS installation

To use latest work from leJOS project copy the content from the folder snapshot and copy them into your leJOS installation.



In windows, it is not necessary to reboot the system. I recommend that you update your NXT brick when the project update the firmware and classes.

17.4.- How to package NXJ programs with Jar

The Java™ 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	<code>jar cf jar-file input-file(s)</code>
To view the contents of a JAR file	<code>jar tf jar-file</code>
To extract the contents of a JAR file	<code>jar xf jar-file</code>
To extract specific files from a JAR file	<code>jar xf jar-file archived-file(s)</code>
To run an application packaged as a JAR file (requires the Main-class manifest header)	<code>java -jar app.jar</code>

18.- Links

18.1.- 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>

18.2.- 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/>

18.3.- 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%3Abhbv0xwyrIk>

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>

Robotics courses with NXT and leJOS

<http://www.daimi.au.dk/~u020841/Lego/>

<http://legokillerrobot.blogspot.com/2008/09/t-4-sept.html>

18.4.- 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

18.5.- 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/

18.6.- Software links

Eclipse IDE

<http://www.eclipse.org/downloads/>

Notepad++

<http://notepad-plus.sourceforge.net/uk/site.htm>

18.7.- Multithreading Links

<http://sol.cs.fhm.edu/rs/rcx-threads/index.html>

<http://www.cs.fit.edu/~ryan/java/language/thread.html>

<http://profesores.elo.utfsm.cl/~agv/elo329/Java/threads/JavaThreads.html>

<http://www.cs.wustl.edu/~schmidt/patterns-ace.html>

<http://gee.cs.oswego.edu/dl/cpj/>

<http://www.dcs.warwick.ac.uk/~sgm/cs237/java/>

<http://www.cs.umd.edu/class/fall2002/cmsc433-0201/lectures/cpjslides.pdf>