UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

# SIMULADOR BASADO EN EASY JAVA SIMULATIONS EN UN ROBOT LEGO

Proyecto Final de Carrera

Ingeniería Informática

**Autor**: Manuel González Marco

**Director**: Marina Vallés Miquel

Valencia, 4 de Julio de 2013

# Resumen

El proyecto realizado consiste en el desarrollo de un simulador para emular el funcionamiento de la forma más real posible de un robot Lego Mindstorms, para cuyo fin ha sido utilizado como software "Easy Java Simulations (EJS)". El simulador nos permite estudiar distintas estrategias de control del robot Lego en distintos escenarios sin tener que implementarlos sobre el robot real. Así, se dispondría de una fase inicial de validación previa en donde las condiciones de funcionamiento son similares a las del sistema real.

**Palabras clave:** simulador, robot, Lego, Mindstorms, Easy Java Simulations.

# Tabla de contenidos

# 1. Introducción

En este proyecto se ha realizado la simulación de movimientos de un LEGO NXT. Los principales problemas que teníamos a priori era conseguir unos movimientos lo más reales posibles y qué hacer cuando el robot se encuentre un obstáculo en la trayectoria.

Para poder simular los movimientos del LEGO NXT con la mayor precisión y de la forma más real posible, se partía de un modelo dinámico formado por 5 ecuaciones. Dichas ecuaciones se pueden ver a continuación, las cuáles serán explicadas en el apartado 3.2 con mayor detalle.

$$X_k = X_{k-1} + v_{k-1}T_s\cos\left(\theta_{k-1} + \omega_{k-1} \cdot 0{,}5T_{s)}\right.$$

$$Y_k = Y_{k-1} + v_{k-1}T_s\sin\left(\theta_{k-1} + \omega_{k-1} \cdot 0{,}5T_s\right)$$

$$\theta_k = \theta_{k-1} + T_s\omega_{k-1}$$

$$v_k = v_{k-1} + T_s a_{k-1}$$

$$\omega_k = w_{k-1} + T_s\alpha_{k-1}$$

En cuanto a la segunda problemática, la simulación se ha dividido en 3 versiones. La primera versión se ha realizado una simulación sin controlador, en la segunda incluimos un controlador a la simulación para evitar las colisiones y por último, en la tercera, se puede realizar una simulación basada en los movimientos realizados por un robot LEGO NXT comunicado con el PC mediante bluetooth.

# 2. Teoría

## 2.1. Simuladores

Un simulador es una máquina que reproduce el comportamiento de un sistema en ciertas condiciones. Los simuladores son utilizados tanto en el ámbito de ocio y diversión como en el profesional.

En el primero, los simuladores nos permiten conducir coches deportivos, volar aviones... mientras que en el segundo, nos permiten formar a personas que tienen una gran responsabilidad en su cargo, ya que sus eventuales errores podrían ocasionar riesgo en la vida de terceros o, en el caso de la industria, se utilizan con el objetivo de mejorar e incrementar la eficiencia de la misma, ya que nos permite hacer simulaciones de diferentes procesos antes de que ocurran en la realidad, por lo que no es necesario hacer ningún prototipo real con el malgasto de material y dinero que ello conlleva. Los simuladores suelen combinar partes mecánicas o electrónicas y partes virtuales que le ayudan a simular la realidad.

### 2.1.1. Tipos de simulador

Existen muchos tipos de simuladores, a continuación podemos ver una descripción de algunos de ellos.

- Simulador de conducción: permite a alumnos de autoescuelas simular situaciones reales de la conducción de una manera más segura, sobretodo en las primeras clases, para evitar correr riesgo de accidente.
- Simulador de carreras: es el simulador más conocido. Se puede conducir un automóvil, motocicleta, etc.
- Simulador de vuelo o de aviones.
- Simulador de trenes.
- Simulador de vida o de dinámica familiar: permite emular una persona y su vida. Ejemplo: Los sims
- Simulador de negocio: permite simular un entorno empresarial.
- Simulador político: permite actuar como político.
- Simulador de redes.
- Simulador clínico médico: permite practicar sobre pacientes virtuales casos clínicos. Esto ayuda a los médicos a saber afrontar estas situaciones en la vida real. También se utiliza para evaluar la actuación de un colectivo respecto a un caso clínico virtual respecto al protocolo establecido.
- Simulador musical.
- Simulador termo solar.

# 2.2. Métodos de integración numérica

Los métodos de integración numérica forman una gran gama de algoritmos que se utilizan para el cálculo del valor numérico de una integral definida y, por lo tanto, en ocasiones el término se utiliza para describir algoritmos numéricos que resuelven ecuaciones diferenciales. El término cuadratura numérica, también llamado cuadratura de forma abreviada, es más o menos sinónimo de integración numérica, especialmente si se aplica a integrales de una dimensión aunque para el caso de dos o más dimensiones, es decir, integral múltiple, también se utilizan.

Para poder entender mejor qué son los métodos de integración numérica, deberemos entender los sistemas dinámicos continuos y los paradigmas de modelado.

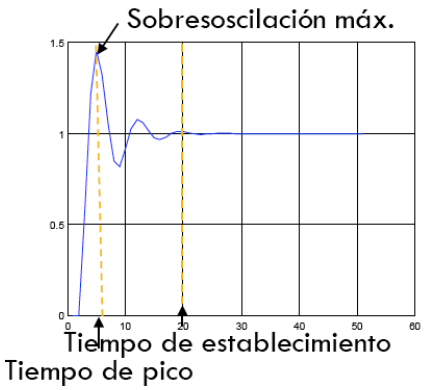
**SISTEMAS DINÁMICOS CONTINUOS**

Los sistemas dinámicos continuos, son aquellos cuyo comportamiento varía con el tiempo y de forma continua respecto a éste.

Existen diversos tipos como sistemas mecánicos, los cuáles están caracterizados por masa, inercia, elasticidad y disipación de energía, y varían debido a fuerzas, pares, desplazamientos específicos… que actúan sobre ellos; los sistemas eléctricos, que son circuitos formados por componentes resistivos, capacitivos o inductivos, los cuáles son excitados por voltaje o corriente; y otros sistemas pueden ser los hidráulicos, térmicos o mixtos.

Los sistemas mixtos se distribuyen en 3 tipos, hidromecánicos, termo-mecánicos y electro-mecánicos.

Las principales características de estos sistemas son la respuesta del sistema ante entradas externas, perturbaciones y condiciones iniciales. La salida con respuesta en el transitorio o en el permanente.




Gráfica de un sistema continuo con sobreoscilación

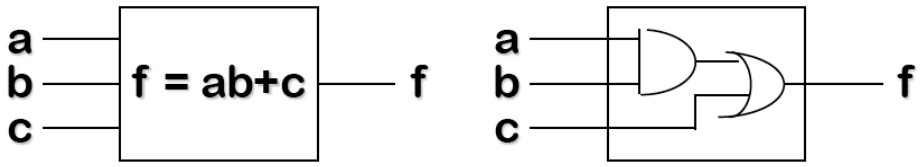
## MODELADO

El modelado tiene una gran variedad de paradigmas, los cuáles se pueden clasificar en base a:

1) Los conjuntos de valores de las variables descriptivas del sistema:
    a. continuos (conjunto de valores reales).
    b. discretos (conjunto discreto).
2) El tiempo:
    a. continuo: el tiempo evoluciona de forma continua.
    b. de estados o eventos discretos: el tiempo evoluciona a saltos.
3) La representación:
    a. comportamental (caja negra): solo observación del comportamiento.
    b. Estructural (caja blanca): se modela la estructura del sistema.

Tanto el modelado comportamental como el estructural nos permiten construir un modelo complejo a base de modelos simples. Estos modelos suelen sufrir simplificaciones debido a la complejidad del sistema a modelar.

La complejidad de los sistemas a modelar puede ser cualitativa o cuantitativa. La primera, depende de la semántica del paradigma, mientras que la segunda depende del número de componentes del modelo, por lo que está ligado al sistema de modelado y no al paradigma usado. Para reducir la complejidad cuantitativa se pueden utilizar modelos jerárquicos.




Ejemplo de representación de un modelo de circuito

Algunos ejemplos de modelado son los sistemas dinámicos de variables continuas(CVDS) o de eventos discretos. Los primeros se describen con ecuaciones diferenciales, paradigma de variables continuas en tiempo continuo y los segundos utilizan un paradigma DEVS de variables discretas a tiempo continuo.

Los modelados se pueden representar de muchas formas, como pueda ser la ecuación diferencial, la función de transferencia o la ecuación del espacio de estados, entre otras.

$$\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = u(t) y(0^-) = y_0;\ \dot{y}(0^-) = \dot{y}_0$$

Ejemplo Ecuación diferencial

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{s^2 + a_1 s + a_0}$$

Ejemplo de función de transferencia

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}0 & -a_0\\ 1 & -a_1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}u(t)$$

$$y(t) = \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

Ejemplo de ecuación del espacio de estados

## MÉTODOS DE INTEGRACIÓN NUMÉRICA

La integración numérica es una herramienta necesaria para obtener valores aproximados de integrales definidas las cuáles no se pueden resolver analíticamente. Este método es utilizado en la ciencia y en la ingeniería.

Los métodos se basan en calcular las soluciones anteriores para instantes de tiempo concretos $t_i$. Para realizar los cálculos utilizamos el llamado paso de integración. Estos pasos son los instantes separados una cantidad fija de tiempo, $\Delta t$. La calidad de la solución dependerá del tamaño del paso.

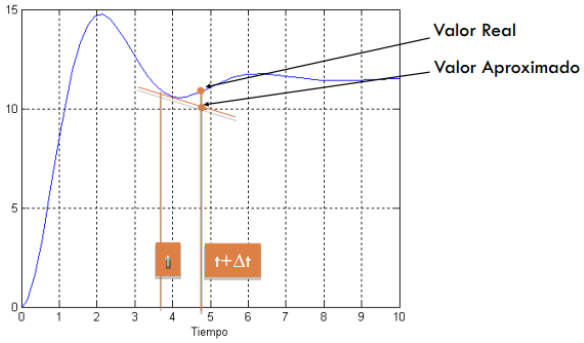
Inicialmente existen los métodos implícitos y los explícitos.

## MÉTODOS EXPLÍCITOS

Los métodos explícitos permiten calcular el vector de variables de estado en cada instante en función de las variables en instantes anteriores. Como ejemplo de método más sencillo encontramos el de Euler. Aunque en la práctica se suelen utilizar métodos avanzados como Runge-kutta o el predictor-corrector debido a que son más exactos y tienen mayor estabilidad numérica.

Estos métodos son los más utilizados debido a su fácil programación, sin embargo, debemos comentar que tienen una gran limitación a la hora de representar simultáneamente fenómenos rapidos y lentos.Para poder preservar la estabilidad de la integración numérica nos vemos obligados a utilizar pasos de integración pqueños debido a la existencia de un

sistema de variables de tiempo pequeñas. Mientras que la presencia de constantes de tiempo grandes obliga a simular periodos de tiempos largos para observar la respuesta del sistema. La presencia simultánea de constantes de tiempo pequeñas y grandes conduce a sistemas matemáticamente "rígidos", que consumen grandes recursos de computación.

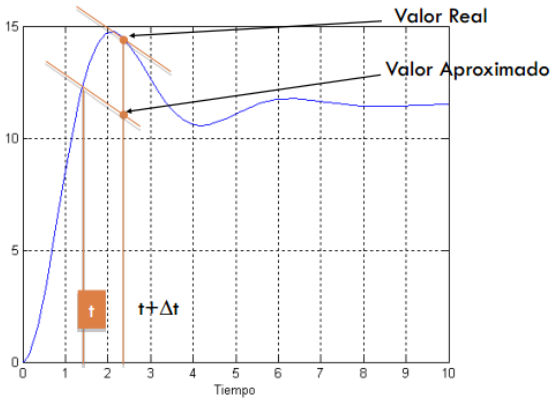



$$v(t + \Delta t) \approx v(t) + \Delta t \cdot \dot{v}(t)$$

Función de Integración

## MÉTODOS IMPLÍCITOS

Los métodos implícitos tienen la ventaja de la mayor exactitud desde el punto de vista numérico, lo cual conlleva que requiere de una carga computacional mayor debido a la necesidad de resolver simultáneamente un conjunto de ecuaciones no lineales al menos una vez en cada paso de integración.




$$v(t + \Delta t) \approx v(t) + \Delta t \cdot \dot{v}(t + \Delta t)$$

Función de Integración

La forma más rápida de resolver un bucle, es hacerlo de forma manual, aunque no siempre es posible debido a que algunos lenguajes de simulación proporcionan resolvederos de bucle implícito.

$$y = IMPL(y_{inicial}, errmax, y_{siguiente})$$
$$x = f(y)$$
$$y_{siguiente} = g(x)$$

Ejemplo código resolución bucle con resolvedero

Algunos lenguajes de simulación no poseen resolvederos de bucle implícito y, por lo tanto, utilizan otros tipos de mecanismos para conseguir el mismo resultado, como puedan ser las secciones procedurales. Estas secciones son un conjunto de instrucciones que el ordenador de instrucciones trata como un todo.

PROCED $y = y_{inicial}$
$x = f(y_{inicial})$
$y = g(x)$

ENDPRO

PROCED $dummy = y$
$y_{inicial} = y$

ENDPRO

Ejemplo Código resolución bucle sin resolvedero

Otros grupos que también podemos encontrar son los métodos monopaso y los multipaso.

## 2.2.1. Métodos Monopaso

Los métodos más comunes son los desarrollos de Taylor, métodos de Euler y los métodos de Runge-Kutta. A continuación, realizaremos una explicación más profunda de los distintos métodos.

### 2.2.1.1. Desarrollo de Taylor

Dada la ecuación diferencial $^{x}/_{dt} = f(t, x(t))$ y la condición inicial x (t0) = x0, se puede aplicar el desarrollo de Taylor a la solución de x (t) de esta ecuación en el punto $t_0$ con el incremento h.

$$x(t_0 + h) = x(t_1) = x(t_0) + hf\big(t_0, x(t_0)\big) + \frac{h^2}{2!}f'(t_0, x\big(t_0, x(t_0)\big) + \cdots +$$
$$+ \frac{h^m}{m!}f^{(m-1}(t_0, x\big(t_0, x(t_0)\big) + \frac{h^{m+1}}{(m+1)!}f^{(m}\big(\xi, x(\xi)\big); \xi \in (t_0, t_1)$$
$$\frac{dx}{dt} = f(t, x(t))$$

- Si prescindimos de todos los términos a partir de la potencia m-pésima tendríamos una aproximación de x ($t_1$) que llamaremos $x_1$.
- El cálculo diremos que es de orden $h^m$ y el error cometido de orden $h^{m+1}$.

El error es de truncamiento local en el subintervalo [$t_0$, t1]. Su acotación puede venir dada por:

$$\big|\varepsilon_{0,1}\big| \leq \frac{h^{m+1}}{(m+1)!}k_1 \qquad\qquad si\ k_1 \geq \max_{[t_0,t_1]}\big|f^{(m}(t, x(t))\big|$$

Cuantos más términos se tomen mejor aproximación podremos calcular.

## 2.2.1.2. Método de Euler

Tanto en matemática como en computación, el método de Euler, llamado así en honor a Leonhard Euler, es un procedimiento para resolver ecuaciones diferenciales ordinarias a partir de un valor inicial.

Dada una ecuación diferencial de primer orden, $\dot{x}(t) = f(x,t)$ , lo que se desea obtener x (t) en instantes sucesivos de tiempo, por tanto:

$$\int_{x_j}^{x_{j+1}} dx(t) = \int_{t_j}^{t_{j+1}} f(x,t)dt$$

Si suponemos que el paso es suficientemente pequeño, se puede considerar que f(x, tj) constante en todo el intervalo:

$$x_{j+1} - x_j = f(x_j, t_j)h$$
$$x_{j+1} = x_j + f(x_j, t_j)h$$

El error de aproximación de este método es proporcional a h2 (series de Taylor).

## 2.2.1.3. Métodos de Runge-Kutta

Este último método que vamos a ver se basa en mantener términos de orden superior de la expansión en series de Taylor. El método de orden cuatro utiliza información de la derivada en 4 puntos dentro de un paso de integración, lo que nos proporciona un método de más exactitud para estimar la solución. A su gran exactitud hemos de añadirle su fácil implementación, lo que provoca que sea muy utilizado.

$$X_{J+1} = X_J + \frac{H}{6}(a + 2b + 2c + d)$$

$$A = f(x_j, t_j) \qquad C = f\left(x_j + \frac{h}{2}B, t_j + \frac{h}{2}\right)$$

$$B = f\left(x_j + \frac{h}{2}A, t_j + \frac{h}{2}\right) \qquad D = f(x_j + hC, t_j + h)$$

## 2.2.2. Métodos Multipaso

Mientras que los métodos monopaso solo utilizan para calcular el valor aproximado $x_{j+1}$ de la curva integral x (t) en $t_{j+1}$ el valor inmediatamente anterior $x_j$, los multipaso también tienen en cuenta $x_{j-1}$, $x_{j-2}$,…., $x_{j-p}$. Esto provoca un mejor ajuste.

Todos los algoritmos multipaso se basan en un algoritmo base:

$$x_{j+1} = x_{j-m} + h\left[(m+1)f_j + \frac{1-m^2}{2}\nabla f_j + \frac{5-3m^2+2m^3}{6}\cdot\frac{\nabla^2 f_j}{2!} + \frac{9-4m^2+4m^3-m^4}{4}\frac{\nabla^3 f_j}{3!} + \cdots\right]$$

Este algoritmo nos permite calcular el valor aproximado de $x_{j+1}$ en función de uno precedente y de las diferencias regresivas de $f_i$ hasta la orden p inclusive, para lo cual, es necesario conocer $x_j$, $x_{j-1}$, …, $x_{j-p}$. Lo podemos aplicar a partir de $x_p$. Para obtener los primeros p se debe de utilizar un método monopaso.

La diferencia entre los distintos métodos multipaso se encuentra en el valor de m empleado y el orden de la diferencia regresiva más alta, o dicho de otra forma, el intervalo de integración y el número de puntos para aproximar.

Lo métodos más utilizados son:

- Adams-Bashforth.
- Predictores-correctores de Adams-Moulton.
- Predictores-correctores de Milne.

### 2.2.2.1. Método Adams-Bashforth

La técnica de Adams-Bashforth de cuarto orden es un método explicito de 4 pasos. Considera m=0 y p=3.

$$x_{j+1} = x_j + h\left[f_j + \frac{1}{2}\nabla f_j + \frac{5}{12}\nabla^2 f_j + \frac{3}{8}\nabla^3 f_j\right]$$

Después de desarrollar las diferencias regresivas llegamos a la siguiente ecuación:

$$x_{j+1} = x_j + \frac{h}{24}\left[55f_j - 59f_{j-1} + 37f_{j-2} - 9f_{j-3}\right]$$

## 2.2.2.2. Predictor-corrector de Adams-Moulton

Este método emplea 2 fórmulas:

$$x_{j+1} = x_j + \frac{h}{24}\left[55f_j - 59f_{j-1} + 37f_{j-2} - 9f_{j-3}\right]$$
$$x_{j+1} = x_j + \frac{h}{24}\left[9f_{j+1} + 19f_j - 5f_{j-1} + f_{j-2}\right]$$

siendo $f_{j+1} = f(t_{j+1}, x_{j+1})$. Necesario conocer $x_{j+1}$.

El procedimiento está compuesto por 5 pasos:

1) Calcular el valor aproximado de $x_{j+1}$ utilizando la primera fórmula.
2) Calcular $f_{j+1}$ a partir de este valor.
3) Aplicar la segunda fórmula para tener una mejor aproximación de $x_{j+1}$.
4) Si la diferencia entre el primer $x_{j+1}$ y el segundo es pequeña, devuelve el $2^0$.
5) Si no se repite el segundo cálculo de $x_{j+1}$ usando el último.

## 2.2.2.3. Predictor-corrector de Milne

Este predictor-corrector es tanto de cuarto orden como de sexto orden. Cada orden posee 2 formulas.

- Cuarto Orden:

$$\begin{cases} x_{j+1} = x_{j-3} + \frac{4h}{3}(2f_j - f_{j-1} + 2f_{j-2}) \\ x_{j+1} = x_{j-1} + \frac{h}{3}(f_{j+1} + 4f_j + f_{j-1}) \end{cases}$$

- Sexto Orden:

$$\begin{cases} x_{j+1} = x_{j-5} + \frac{3h}{10}(11f_j - 14f_{j-1} + 26f_{j-2} - 14f_{j-3} + 11f_{j-4} \\ x_{j+1} = x_{j-3} + \frac{2h}{45}(7f_{j+1} + 32f_j + 12f_{j-1} + 32f_{j-2} + 7f_{j-3} \end{cases}$$

## 2.2.3. Selección del paso de simulación

La selección acertada de un paso de simulación es algo muy importante, ya que si es muy grande puede llevar a errores grandes o que el proceso numérico sea inestable; y, por el contrario, si es muy pequeño provoca excesivo número de iteraciones y errores de computación grandes debidos al redondeo.

Esto nos lleva a tener que buscar un compromiso a la hora de dividir en puntos la curva que queremos simular. Se debe de tener en cuenta 2 factores dinámicos como son la dinámica natural del sistema y la forma de onda de la señal de entrada.

**BASADO EN LA DINAMICA NATURAL DEL SISTEMA**

Si nos basamos en la dinámica natural del sistema, las raíces de la ecuación característica pueden ser reales o complejas. En caso de que sean reales, la respuesta del sistema será no oscilatoria mientras que si alguna raíz es compleja, la respuesta será inicialmente oscilatoria.

**Si raíces reales:**

$$s = -a$$
$$\lambda = |-a|$$
$$h = \frac{1}{10\lambda_{max}}$$

**Si raíces complejas:**

$$s = -a \pm bj$$
$$\lambda = \left|\sqrt{a^2 + b^2}\right|$$
$$h = \frac{1}{10\lambda_{max}}$$

También se debe tener en cuenta que si el orden del sistema es elevado, las raíces son difíciles de determinar, por lo que en este caso será recomendado determinar el paso de simulación a partir de los parámetros de la ecuación característica.

Si ecuación característica es:

$$\alpha_n s^n + \alpha_{n-1} s^{n-1} + \cdots + \alpha_1 s + \alpha_0 = 0$$

La media logarítmica de los valores propios es:

$$\bar{\lambda}_L = \sqrt[n]{\frac{\alpha_0}{\alpha_n}} \qquad h = \frac{1}{10\bar{\lambda}_L}$$

## BASADO EN LA SEÑAL DE ENTRADA

Se escoge su frecuencia $\omega_{in}$ para obtener el paso, si la entrada contiene frecuencias mayores que la dinámica del sistema. Por el contrario, si la señal de entrada es muy compleja, se escoge el menor periodo reconocible $\tau_{in}$.

$$\Delta t = \frac{1}{10\omega_{in}} \qquad O \qquad \Delta t = \frac{\tau_{in}}{10}$$

Existen diversos métodos para la selección de paso de variable, como por ejemplo:

1. Métodos de Tamaño de Paso Variable.
2. Ecuaciones Stiff.

## MÉTODOS DE TAMAÑO DE PASO VARIABLE

El tamaño de paso fijo puede que sea solo necesario en un determinado periodo de tiempo, debido a que conforme avanza el tiempo cambia el contenido frecuencia. Para poder tener un paso de integración dinámico se debe realizar algunas estimaciones de la exactitud de los resultados actuales calculados. El método más sencillo consiste en calcular la respuesta en el siguiente punto de 2 formas, como podemos observar a continuación:

- Se evalúa el punto en t+h de la forma habitual = $x_a$.
- Se evalúa otra solución considerando 2 pasos de h/2 = $x_c$.
- Una vez calculado, comprobamos con el error de truncamiento.

$$E_T \cong \frac{x_c - x_a}{2^P - 1}$$

## ECUACIONES STIFF

Aquellas ecuaciones del sistema cuyos valores propios difieren en varios órdenes de magnitud son llamadas ecuaciones stiff. Imponen diferentes requerimientos de paso de integración y se requieren métodos especiales.

Para que una ecuación sea de stiff debe cumplir que la solución buscada varía lentamente, pero existen soluciones cercanas que varían rápidamente, de tal forma que el paso de simulación debe ser pequeño para obtener resultados satisfactorios.

Los métodos para resolver ecuaciones de stiff pueden tomar pasos de tamaño más grande ya que realizan más cómputo en cada paso. Son implícitos y algunos se basan en la fórmula de Rosenbrock.

## 2.3. Robots

Un robot es una entidad mecánica artificial o virtual. Por lo general, en la práctica, es un sistema electromecánico que ofrece la sensación de tener un propósito propio debido a su apariencia o sus movimientos. La palabra robot puede referirse tanto a sistemas virtuales de software como a mecanismos físicos, aunque para los primeros se suele utilizar el término de bots.

No existe un acuerdo sobre qué maquinas pueden considerarse robots, pero sí existe un acuerdo general entre los profesionales y el público sobre las acciones que debe realizar un robot:

- Moverse.
- Hacer funcionar un brazo mecánico.
- Sentir y manipular su entorno.
- Mostrar un comportamiento inteligente, especialmente en el caso de que imite el comportamiento humano o de algún animal.

Las máquinas totalmente autónomas aparecen en el siglo XX, aunque la historia sobre ayudantes, acompañantes artificiales y los intentos por crearlos tienen una larga historia. El primer robot programable y dirigido de forma digital, el Unimate, fue instalado en 1961 para levantar piezas calientes de metal de una máquina de tinte y colocarlas.

Normalmente, la gente reacciona de forma positiva ante los robots con los que interactúa. Actualmente, se está convirtiendo en algo común en los hogares disponer de robots para la limpieza y mantenimiento del hogar. Aunque, sí que es cierto, que existe una cierta ansiedad sobre el impacto económico de la automatización y la amenaza del armamento robótico, la cual se ve reflejada en el retrato a menudo perverso y malvado de robots presentes en obras culturales. Realizando una comparación con los robots de ficción, los reales son aún muy limitados.

### 2.3.1. Robótica

Es la ciencia y la tecnología de los robots. Se ocupa del diseño, manufactura y aplicaciones de los robots. La robótica está formada por diversas disciplinas como son:

- La mecánica.
- La electrónica.
- La informática
- La inteligencia artificial (IA).
- La ingeniería de control.
  Otras áreas importantes: el álgebra, los autómatas programables y las máquinas de estados.

# 2.3.2. Historia de la robótica

La historia de la robótica va ligada a la construcción de *"artefactos"*, que trataban de cumplir el deseo humano de crear seres a su semejanza y que lo descargasen del trabajo. Leonardo Torres Quevedo, ingeniero español que construyó, en otras muchas cosas, el primer mando a distancia para su automóvil mediante telegrafía, el ajedrecista automático y el primer transbordador aéreo, acuño el término *"automática"* en relación con la teoría de la automatización de tareas tradicionalmente asociadas a los humanos.

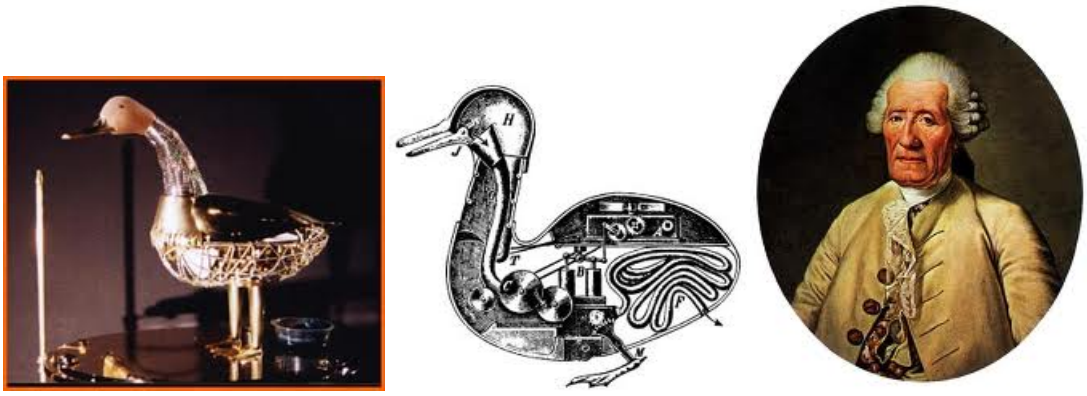
Karel Čapek, escritor checo, acuño en 1921 el término *"Robot"* en su obra dramática *"Rossum's Universal Robots / R.U.R."*, a partir de la palabra checa robota, la cual significa trabajo forzado o servidumbre. Isaac Asimov definió el término robótica como ciencia que estudia a los robots. También creo las 3 leyes de la robótica en el año 1942:

1. Un robot no puede actuar contra un ser humano o, mediante la inacción, que un ser humano sufra daños.
2. Un robot debe de obedecer las órdenes dadas por los seres humanos, salvo que estén en conflictos con la primera ley.
3. Un robot debe proteger su propia existencia, a no ser que esté en conflicto con las 2 primeras leyes.

En el siglo I a.C. y antes ya se realizaron descripciones de más de 100 maquinas y autómatas, incluyendo un artefacto con fuego, un órgano de viento, una maquina operada mediante una moneda, una máquina de vapor, en Pneumática y autómata de Herón de Alexandria. El nombre del robot es autómata. Estas descripciones fueron realizadas por Ctesibius de Alexandria, Filón de Bizancio, Herón de Alexandria y otros. Más en concreto, el año 350 a.C. el matemático griego Arquitas de Tarento construye un pájaro mecánico a vapor.

El Matematico griego Arquitas de Tarento

En 1206, Al-Jazari, construyó el llamado “Barco con cuatro músicos robotizados”. Se trata del primer humanoide programable. En 1495, Leonardo Da Vinci realizo el diseño de un robot humanoide llamado Caballero mecánico. Alrededor de 250 años después, Jacques de Vaucanson diseño un robot llamado Digesting Duck, que era un pato mecánico que era capaz de comer, agitar sus alas y excretar.



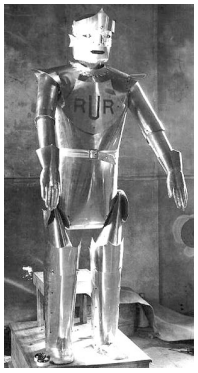
Imágenes del robot Digesting Duck junto al retrato de Jacques de Vaucanson, su creador

Durante el siglo XIX, Hisasshige Tanaka creó los llamados juguetes Karakuri. Estos son juguetes mecánicos japoneses que sirven té, disparan flechas y pintan. También en este siglo, en el año 1801, un francés llamado Josep Jacquard construye un telar automático controlado por tarjetas perforadas.

Juguete Karakuri


Telar automático

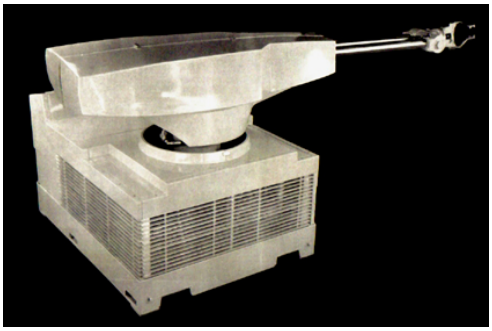
El siglo XX, es un siglo muy productivo en lo que respecta a robots. En 1921 aparece el primer autómata de ficción llamado “robot”, aparece el R.U.R. (Rossum’s Universal Robots) creado por el austro-húngaro Karel Čapek.




Imágen del R.U.R.

En 1938, los americanos  Willard Pollard y Harold Roselund fabricaron la primera máquina para pintar con spray,  la empresa que se encargo de fabricarla fue “DeVilbiss”, la cual sigue abierta. Unos años más tarde, entre los años 1939 y 1940, la compañía Westinghouse Electric Corporation exhibe un robot humanoide en World’s Fairs, cuyo nombre era Electro. Unos años después en 1948, William Grey Walter realiza la exhibición de un robot con comportamiento biológico simple llamado Elsie y Elmer.

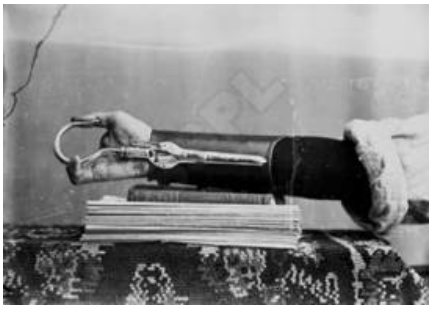
En 1954, George Devol diseña el primer robot programable comercial, fabricado por la compañía Unimation fundada por George Devol y Joseph Engelberger. Su nombre era “Unimate”. Pero no será hasta tres años después, en 1961, cuando se comience a comercializar.



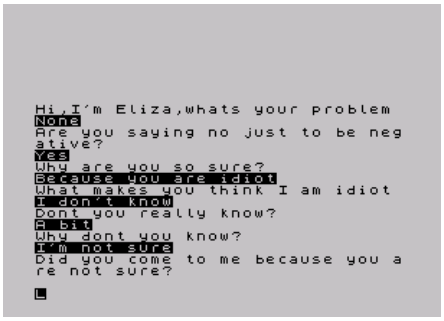
Imágen de una unidad Unimate

En 1959 se funda el Artificial Inteligencie Laboratory en el MIT. Este ha sido la piedra angular en robótica universitaria durante el siglo XX, solo comparable al Robotices Institute de la “Carnegie Melon University”. Este mismo año sale al mercado el primer robot comercial llamado “Versatran”.

En 1962 aparece el primer brazo industrial robótico. Al año siguiente, en 1963, el primer robot “palletizing” creado por Fuji Yusoki Kogyo, el cual se llamaba Palletizer. Dos años después, en 1965, se funda el Robotics Institute en la “Carnegie Melon University”. Unos años más tarde, en 1966, se crea “Eliza”, el primer programa de inteligencia artificial.

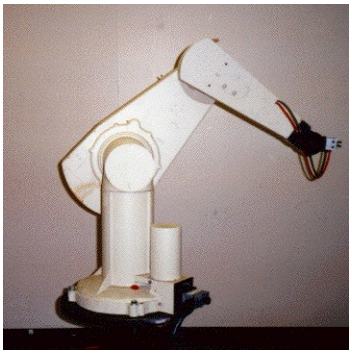


Primer brazo industrial




Eliza, el primer programa de I.A.

En 1973, KUKA Robot Group, crea Famulus, el primer robot con 6 ejes electromecánicos. También en el año 1973, aparece el primer robot controlado por un mini-ordenador, el robot fue conocido como “T3”. Dos años después, aparece el brazo manipulador programable universal, un producto de Unimation. Su nombre era “Puma” y fue creado por Victor Scheinman.



En estas imágenes podemos ver a la izquierda el robot Famulus y a la derecha el Puma

En 1976, el robot de la NASA “Vinking II” aterriza en Marte. Este disponía de un brazo articulado. A partir de 1978, comienzan a surgir numerosas empresas dedicadas a la fabricación de robots para la industria. Durante la década de los 80, solo en EE.UU. surgieron más de 10 empresas de gran capital social, fundadas desde sus respectivas corporaciones matrices del sector industrial.

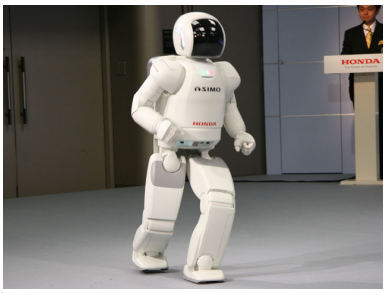
En 1986, la empresa japonesa Honda inicia un proyecto para construir un robot humanoide. Tanto su evolución como los numerosos problemas, los cuales dieron pie a que numerosos científicos consideraran que Honda no lograría su objetivo, se mantuvieron en secreto. En 1989, el Massachusetts Institute of Technology (EE.UU.) desarrolla el primer robot andante llamado “Genghis”.

Imagen del Genghis

Después de 11 años de investigación, en 1997, Honda presentó a P3 un enorme robot humanoide. Cuando la marca hizo público los resultados, muchos investigadores que trabajaban en proyectos parecidos se quedaron atónitos, ya que sus proyectos estaban a años luz de este.

En 1999, Sony lanza el perro mascota robótico “Aibo” y un año más tarde en 2000, presentó a “Robodex 2000”, un pequeño robot humanoide. También en el 2000, Honda presenta a “Asimo” el robot humanoide.

Robot Aibo interactuando con un perro




Humanoide Asimo

Ya en el siglo XXI, en 2001, en Estados unidos es autorizado para su uso en hospitales Estadounidenses un cirujano robótico llamado Cyberknife.

Primer robot cirujano, Cyberknife

En 2003, Sony comienza a comercializar un robot humanoide llamado "Qrio", el cuál es completamente autónomo y capaz de correr. Un año después, en 2004, tiene lugar la primera edición del "Darpa Grand Challenge". En 2005, investigadores de la Universidad de Cornell (EE.UU.) construyen un robot que se replica a sí mismo.

## 2.3.3. Clasificación según arquitectura

La arquitectura es definida por el tipo de configuración general del robot, puede ser metamórfica. El concepto de metamorfismo se ha introducido para incrementar la flexibilidad funcional de un robot a través del cambio de su configuración por el propio robot. El metamorfismo está compuesto por varios niveles, desde los más elementales, como pueda ser el cambio de una herramienta, hasta los más complejos, como el cambio o alteración de algunos de sus elementos o subsistemas estructurales. Debido a la gran cantidad de dispositivos y mecanismos que se pueden considerar como robots, es muy complejo establecer una clasificación coherente de dichos mecanismos que resista un análisis crítico y riguroso.

La clasificación de robots según sus arquitectura está compuesta por: Poliarticulados, móviles, androides, zoomórficos e híbridos.

- **1. Poliarticulados**

  Este grupo está formado por robots de muy diversa forma y configuración pero tienen una característica en común. Esta característica es la de ser sedentarios y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas y con un número limitado de grados de libertad. Este grupo está formado por los manipuladores, los robots industriales y los robots cartesianos. Son empleados cuando es preciso abarcar una zona de trabajo bastante amplia o alargada, reducir el espacio ocupado en el suelo o actuar sobre objetos con un plano de simetría vertical.

- **2. Móviles**

  En este grupo se encuentran robots con gran capacidad de desplazamiento, basada en carros o plataformas y dotada de un sistema locomotor de tipo rodante. Reciben información de su entorno a través de sus sensores para guiarse o bien siguen su camino por telemando. Este tipo de robots se utilizan para transportar piezas de un punto a otro de una cadena de montaje. Se guían mediante pistas materializadas a través de la radiación electromagnéticas de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos. Además, están dotados de un nivel relativo de inteligencia.

- **3. Androides**

  Son robots que reproducen de una manera parcial o total la forma y el comportamiento cinemático del ser humano. Hoy en día son dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, principalmente, al estudio y experimentación. La mayor dificultad de estos robots es la locomoción bípeda, de hecho, es en este aspecto en el que se centran la mayoría de los trabajos en este tipo de robot. La dificultad que encontramos en este sentido es el problema de controlar dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del robot.

- **4. Zoomórficos**

  En este tipo de robots podríamos incluir a los androides en un sentido no restrictivo. Constituyen una clase caracterizada por imitar los sistemas de locomoción de diversos seres vivos. Debido a la gran variedad de sistemas de locomoción que podemos encontrarnos en los seres vicos, aunque podríamos agruparlos en más grupos, se dividen en 2 grupos: caminadores y no caminadores. El grupo de los no caminadores está muy poco evolucionado. Solo se han realizado algunos experimentos en Japón basados en segmentos cilíndricos biselados acoplados entre si y dotados de un movimiento relativo de rotación. Mientras que los zoomórficos caminantes multipedos son muy numerosos y están siendo experimentados en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, piloteando o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

- **5. Híbridos**

  En este tipo de robot son incluidos aquellos que son difíciles de clasificar debido a que su estructura es una combinación de alguna de las anteriormente explicadas, ya sea por conjunción o por yuxtaposición. Un ejemplo de hibrido es un robot segmentado articulado y con ruedas, es al mismo uno de los atributos de los robots móviles y de los robots zoomórficos. Del mismo modo se pueden considerar híbridos algunos robots formados por la yuxtaposición de un cuerpo formado por un carro móvil y de un brazo semejante al de los robots industriales. En parecida situación se encuentran algunos robots antropomorfos y que no pueden clasificarse ni como móviles ni como androides, tal es el caso de los robots personales.

## 2.3.4. Clasificación según generación

La clasificación por generación es la más común y va desde la primera hasta la cuarta.

- **1ª Generación**

  Manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control, ya sea manual, de secuencia fija o de secuencia variable.

- **2ª Generación**

  Robots de aprendizaje. Son robots que repiten una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a trabes de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.

- **3ª Generación**

  Robots con control sensorizado. El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

- **4ª Generación**

  Robots inteligentes. Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

- **Próximas generaciones**

  La siguiente generación, es decir, la quinta, será una tecnología que incorporara 100% inteligencia artificial (IA) y utilizara métodos como los modelos de conducta y una arquitectura de subsumisión, además de otras tecnologías actualmente en desarrollo como la nanotecnología.

  Esta etapa depende totalmente de la nueva generación de jóvenes interesados en robótica, una nueva era de robots nos espera.

# 2.3.5. Inteligencia artificial (IA)

Se denomina a la inteligencia artificial a la rama de las ciencias de la computación dedicada al desarrollo de agentes racionales no vivos.

Para poder entender la anterior definición de una manera correcta se debe entender a un agente como cualquier cosa capaz de percibir su entorno (recibir entradas), procesar tales percepciones y actuar en su entorno (proporcionar salidas), y entiéndase a la racionalidad como la característica que posee una elección de ser correcta, más concretamente, de tender a maximizar un resultado esperado. Para definir la naturaleza del objetivo de esta disciplina se utiliza el concepto de racionalidad debido a que es más general y por ello más adecuado que inteligencia.

Por lo tanto, y de manera más específica, la inteligencia artificial es la disciplina que se encarga de construir procesos que al ser ejecutados sobre una arquitectura física producen acciones o resultados que maximizan una medida de rendimiento determinada, basándose en la secuencia de entradas percibidas y en el conocimiento almacenado en tal arquitectura.

Podemos encontrar distintos tipos de conocimiento y medios de representación del conocimiento, el cual puede ser cargado en el agente por su diseñador o puede ser aprendido por el mismo agente utilizando técnicas de aprendizaje.

También se distinguen varios tipos de procesos validos para obtener resultados racionales, que determinan el tipo de agente inteligente. De más complejos a más simples, los cinco principales tipos de procesos son:

- Razonamiento mediante una lógica formal (análogo al pensamiento abstracto humano).
- Redes neuronales artificiales (análogo al funcionamiento físico del cerebro de animales y humanos).
- Algoritmos genéticos (análogo al proceso de evolución de las cadenas de ADN).
- Búsqueda del estado requerido en el conjunto de los estados producidos por las acciones posibles.
- Ejecución de una respuesta predeterminada por cada entrada (análogas a actos reflejos en seres vivos).

También existen distintos tipos de percepciones y acciones, las cuales pueden ser obtenidas y producidas, respectivamente por sensores físicos y sensores mecánicos en máquinas, pulsos eléctricos u ópticos en computadoras, tanto como por entradas y salidas de bits de un software y su entorno software.

Varios ejemplos se encuentran en el área de control de sistemas, planificación automática, la habilidad de responder a diagnósticos y a consultas de los consumidores, reconocimiento de escritura, reconocimiento del habla y reconocimiento de patrones.

Los sistemas de IA actualmente son parte de la rutina en campos como economía, medicina, ingeniería y la milicia, y se ha usado en gran variedad de aplicaciones de software, juegos de estrategia como ajedrez de computador y otros videojuegos.

### 2.3.6. Inteligencia automática

La ingeniería automática, o también llamada automatización industrial es el uso de sistemas de control, como son el control numérico (NC), controladores lógicos programables (PLC) y otros sistemas de control industriales, relacionados con otras aplicaciones de la tecnología de la información, como son tecnologías de ayuda por computador [CAD, CAM, CAx], para el control industrial de maquinaria y procesos, reduciendo la necesidad de intervención humana.

En el ámbito de la industrialización, la automatización está un paso por delante de la mecanización. Mientras que la mecanización provee operadores humanos con maquinaria para ayudar a exigencias musculares de trabajo, la automatización reduce considerablemente la necesidad para exigencias humanas sensoriales y mentales. Los procesos y los sistemas también pueden ser automatizados.

- PLC: Programmable Logic Controller.
- PAC: Programmable Automation Controller.

# 2.4. Lego Mindstorms NXT

Lego Mindstorms es algo más que un juguete de la famosa marca danesa, puede ser usado por aficionados como perfecto centro de investigación e incluso por ingenieros para desarrollar proyectos más elaborados o difíciles. Es ideal tanto para niños a partir de los 10 – 11 años, incluso 7 si tienen ayuda.

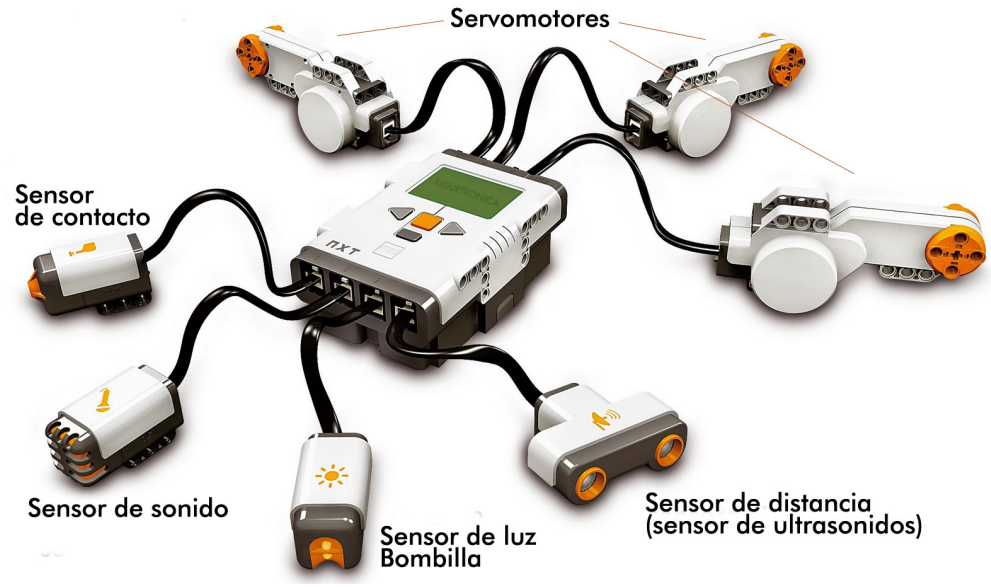
Las posibilidades son muy amplias, por lo que es considerado una gran herramienta de trabajo, y además, su desembolso económico no es muy elevado si lo que se desea es construir un pequeño robot o artefacto programado para hacer algo medianamente curioso. Fomenta la imaginación, planificación, desarrollo de la orientación espacial y, especialmente, la lógica.

Ejemplo de construcción de Lego Mindstorm

La clave de este robot está en el concepto de NXT de ciber-inteligencia, con conexión USB y con un control computarizado de los actos procesados por el cerebro del ladrillo del Lego NXT.

El Lego está compuesto por un microprocesador de 32 bits, una pantalla LCD, puertos de sensores, puertos de locomoción, además de un panel de control y un altavoz. Este gran sistema de movimientos y sensores nos permite crear casi todo lo que se nos ocurra, se puede decir que el límite del mindstorms está en la imaginación y conocimientos del usuario. Se puede llegar a controlar el robot mediante periféricos que tengan Bluetooth como un teléfono móvil, PC, etc.

## 2.4.1. Componentes




Imágen general del ladrillo NXT con varios componentes

**LADRILLO NXT**

Es el cerebro de cualquier proyecto que se realice con el lego. En el interior se encuentra un microcontrolador programable, dentro del cual se encuentra el programa necesario para mover los motores cuando se deban mover, el robot realice los cálculos necesarios o cualquier fin que se nos ocurra.

**SERVOMOTORES**

Los servomotores tienen doble función. Podemos hacer que se muevan a una determinada velocidad o con determinados ángulos y, además, también pueden actuar como sensores de rotación. Por ejemplo, se podría conectar a un volante, y además de girarlo con el motor, si alguien moviera el volante, también permitiría saber cuánto ha girado.

**SENSOR DE SONIDO NXT**

El sensor de sonido posee un pequeño micrófono, el cual permite poder programar un robot que al escuchar una palmada se quede quieto, o al escuchar 2 palmadas apague la luz.

**SENSOR DE CONTACTO**

Este sensor tiene un pequeño pulsado, con el cual se puede detectar si el robot ha chocado con algún obstáculo.

**SENSOR DE DISTANCIA**

El sensor de distancia puede medir distancias de 7 a 1,80 metros, lo que le da al Lego la capacidad de visión. El funcionamiento es el mismo que el empleado por los murciélagos, utiliza ultrasonidos.

**SENSOR DE COLOR**

Este sensor distingue 3 niveles de colores (RGB). Se utiliza para distinguir el color de las pelotas incluidas en el Lego Mindstorms NXT.

**SENSOR BRUJULA NXT**

El sensor magnético puede medir el campo magnético de la tierra y calcular una dirección magnética con la que guiar el sistema de dirección del robot.

**SENSOR BUSCADOR IR NXT**

Este sensor se dedica a detectar señales de infrarrojos. Internamente está formado por una array de sensores.

**SENSOR DE ACELERACION NXT**

Se utiliza para medir la aceleración del robot. Contiene un acelerómetro de 3 ejes que mide la aceleración de los 3 ejes x, y, z. El rango de la aceleración varía de -2g a +2g con una resolución aproximada de 200 unidades por g. También se puede utilizar el sensor para medir la inclinación en los 3 ejes.

**SENSOR ENLACE IR NXT**

Este sensor permite enlazar el ladrillo inteligente Lego Mindstorms NXT con otros modelos Lego que hagan uso de comunicaciones por infrarrojos.

**SENSOR DE TEMPERATURA NXT**

Se utiliza para medir la temperatura, ya sea del ambiente o del líquido, por ejemplo.

**VERSIONES DEL LEGO MINDSTORMS**

Existen 2 versiones del Lego Mindstorms, la NXT 2.0 y la Educa las cuales tienen ligeras diferencias como podemos observar a continuación:

| Versión Comercial | Versión Educa |
|---|---|
| 1 Ladrillo NXT | 1 Ladrillo NXT |
| 1 Sensor de ultrasonidos | 1 Sensor de ultrasonidos |
| 2 Sensores de contacto | 2 Sensores de contacto |
| 3 Servomotores | 3 Servomotores |
| | 1 Sensor de sonido |
| | 1 Sensor de luz |
| 1 Sensor de color | |
| 619 Piezas para construir robots | 437 Piezas para construir un robot |
| Instrucciones de montaje | |
| Pista de pruebas | |
| 1 Cable USB | 1 Cable USB |
| 7 Cables de conexion NXT | 7 Cables de conexion NXT |
| | 3 Lámparas |
| | 3 Cables convertidores RCX - NXT |
| | Caja de plástico |
| | 1 Batería recargable (2100mAh) + Cargador 220v |

## 2.5. Comunicación: Bluetooth

El termino Bluetooth es una especificación industrial para redes inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4GHz (corto alcance).

Los objetivos principales que esta tecnología pretende obtener son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar los cables y conectores entre estos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.

Permite realizar comunicaciones a distancia de hasta 10 metros, incluso a través de obstáculos, como puedan ser paredes…

### 2.5.1. Inicios

Fue desarrollado como un reemplazo del cable en 1994 por Jaap Haartsen y Mattisson Sven. Trabajaban para la compañía Ericsson en Lund, Suecia. La utilidad se basa en la tecnología de saltos de frecuencia de amplio espectro.

Las prestaciones fueron publicadas por una asociación comercial formada por líderes en telecomunicaciones, informática e industria de red llamada SIG (Bluetooth Special Interest Group). Fueron anunciadas formalmente el 20 de Mayo de 1998. Este grupo fue creado por Ericsson, IBM, Intel, Toshiba y Nokia; alcanzando en la actualidad 14000 empresas miembros de la asociación.

### 2.5.2. Nombre y Logo

El nombre procede del rey danés y noruego Harald Blätand, cuya traducción al inglés es Harald Bluetooth. Fue conocido por su capacidad de ayudar a la gente a comunicarse, consiguiendo unificar las tribus noruegas, suecas y danesas, y convertirlas al cristianismo.

Jim Kardash propuso este nombre. Creó un sistema que permitía a los teléfonos móviles comunicarse con los ordenadores y unificar la comunicación de los sistemas digitales.

El logo son las runas de las iniciales del anteriormente mencionado rey danés, la  (Hagall) y la  (Berkana).

Logotipo bluetooth

## 2.5.3. Usos y Aplicaciones

El protocolo bluetooth de comunicaciones está diseñado especialmente para dispositivos de bajo consumo, los cuales requieren corto alcance de emisión y basados en transceptores de bajo costo.

Imagen del ratón de Apple, Mighty mouse

La comunicación es realizada a través de radio frecuencia, lo que nos permite que los dispositivos no tengan que estar alineados ni estar ambos en la misma habitación, siempre que no se salgan del rango. Se clasifican en 3 clases distintas, llamadas “clase1”, “clase2” o “clase3” de pendiendo de la potencia de transmisión del dispositivo. Son compatibles los dispositivos de una clase con los de otra.

| Clase | Potencia máxima permitida (mW) | Potencia máxima permitida (dBm) | Alcance (aproximado) |
|---|---|---|---|
| **Clase 1** | 100 mW | 20 dBm | ~100 metros |
| **Clase 2** | 2.5 mW | 4 dBm | ~10 metros |
| **Clase 3** | 1 mW | 0 dBm | ~1 metro |

Tabla características según clase bluetoth

Casi en todos los casos, la cobertura efectiva de un “clase2” se extiende cuando la conexión es con un “clase1” debido a la mayor sensibilidad y potencia de transmisión de este último, dicho de otra manera, la mayor potencia de transmisión del dispositivo de “clase1” permite que la señal llegue con suficiente energía hasta el “clase2”. Por otra parte, la mayor sensibilidad del dispositivo de “clase1” permite la recepción de la señal del otro pese a ser más débil.

Otra posible clasificación para los dispositivos bluetooth es según su ancho de banda:

| Versión | Ancho de banda |
|---|---|
| **Versión 1.2** | 1 Mbit/s |
| **Versión 2.0 + EDR** | 3 Mbit/s |
| **Versión 3.0 + HS** | 24 Mbit/s |
| **Versión 4.0** | 24 Mbit/s |

Tabla clasificicación según ancho de banda

### 2.5.4. Lista de Aplicaciones

Las posibilidades del bluetooth son muchas, por ello, lo importante de este sistema. A continuación, se enumeran algunas de estas:

- Eliminación de la necesidad de conexiones por cable entre protocolos y accesorios electrónicos.
- Transferencia de fichas de contacto, citas y recordatorios entre dispositivos vía OBEX.
- Sincronización y transferencia de archivos entre dispositivos.
- Control remoto.
- Enlace inalámbrico entre sistemas de audio y los altavoces.
- Se utilizan en los mandos de videoconsolas, como el Dual Shock Sisáis de la PlayStation 3(Sony) o en el mando de la Xbox 360 (Microsoft). También se utiliza en el Wii Remote pero en este caso combinado con infrarrojo.
- Enviar propaganda ligera desde una tienda a la gente que pase por delante del establecimiento.


Mando Dualshock 3(PS3) ejemplo de dispositivo con bluetooth

## 2.5.5. Perfil Bluetooth

Para poder utilizar Bluetooth, un dispositivo debe implementar alguno de los perfiles predeterminados. Estos definen el uso del canal bluetooth, así como canalizar al dispositivo que se quiere vincular.

## 2.5.6. Versiones

Todas las versiones de los estándares de Bluetooth están diseñadas para la compatibilidad hacia abajo, es decir, el último estándar debe cubrir todas las versiones anteriores.

- **Bluetooth v1.0 y v1.0b:** Estas versiones tuvieron muchos problemas, y los fabricantes tenían dificultades para hacer sus productos interoperables. En hardware incluyen de forma obligatoria la dirección del dispositivo Bluetooth (BD_ADDR) en la transmisión, esto fue un gran revés para algunos servicios previstos para su uso en entornos Bluetooth.

- **Bluetooth v1.1 (2002):** Esta versión fue ratificada como estándar IEEE 802.15.1-2002. En esta versión se corrigieron muchos fallos de la v1.0b, se añadió soporte para canales no cifrados y un indicador de señal recibida (RSSI).

- **Bluetooth v1.2 (2005):** Esta versión es compatible con USB 1.1 y tiene una conexión más rápida y Discovery (detección de otros dispositivos bluetooth). El salto de frecuencia pasa a ser adaptable de espectro ampliado (AFH), lo que mejora la resistencia a las interferencias de radio frecuencia, evitando el uso de las frecuencias de lleno en la secuencia de saltos. La velocidad de transmisión es mayor, de hasta 72 kit/s que en v1.1. Añade las conexiones sincrónicas extendidas (ESCO), que mejoran la calidad de la voz de los enlaces de audio al permitir la retransmisión de paquetes corruptos, y, opcionalmente, puede

aumentar la latencia de audio para proporcionar un mejor soporte para la transferencia de datos simultánea.

También incorpora el Host Controller Interface (HCI) y el apoyo a tres hilos UART. Por último, introdujo el control de flujo y los modos de retransmisión de L2CAP. Esta versión fue ratificada como estándar IEEE 802.15.1-2005.

- **Bluetooth v2.0 + EDR (2004):** Esta versión de la especificación fue lanzada en 2004. La principal mejora de esta versión es la introducción de una velocidad de datos mejorada (EDR "Enhanced Data Rate") para acelerar la transferencia de datos. La tasa nominal de EDR es de 3 Mbit/s y la tasa de transferencia de datos práctica es de 2,1 Mbit/s. El EDR utiliza una combinación de Phase Keyng modulation (PSK) y GFSK con dos variantes, π/4-DQPSK y 8DPSK. También puede proporcionar un menor consumo de energía a través de trabajo reducido.

  EDR es una característica opcional, esto hace que los productos pueden reclamar el cumplimiento de "Bluetooth v2.0" sin el apoyo de la mayor tasa de datos. También, hay otras pequeñas mejoras.

- **Bluetooth v2.1 + EDR (2007):** Esta versión, Bluetooth Core Version especificación 2.1 + EDR, es totalmente compatible con v1.2, y fue adoptada por el Bluetooth SIG el 26 de Julio de 2007. El 12 de Junio de 2007, Nokia y SIG anunciaron que Wibree formaría parte de la especificación de Bluetooth.

  Secure Simple Pairing (SSP) es la función titular, la cual proporciona una mejora de la experiencia de emparejamiento de dispositivos Bluetooth. Hay otras mejoras como "la respuesta amplia investigación" (EIR), la cual proporciona más información durante el procedimiento de investigación para permitir un mejor filtrado de los dispositivos antes de la conexión, y oler subrating, lo que reduce el consumo de energía en modo de bajo consumo.

- **Bluetooth v3.0 + HS (2009):** Fue aprobado por le SIG el 21 de Abril de 2009. Soporta velocidades de transferencia de datos teórica de hasta 24 Mbits/s entre sí, aunque no a través del enlace Bluetooth propiamente dicho. La conexión nativa se utiliza para la negociación y el establecimiento, mientras que mediante un enlace 802.11 se realiza el tráfico de datos de alta velocidad. La principal novedad que incorpora es AMP (Alternate MAC / PHY), la adición de 802.11 como transporte de alta velocidad. UWB estaba prevista inicialmente pero fue desechada para esta especificación.

  La transmisión a alta velocidad no es obligatoria, y por lo tanto, los dispositivos marcados con "+HS" incorporan el enlace 802.11 de alta velocidad de transferencia de datos. Si un dispositivo Bluetooth 3.0 no lleva el sufijo "+HS" no apoyara a alta velocidad, y sólo admite una característica introducida en Bluetooth 3.0 +HS (o en CSAT).

También incorpora la alternativa MAC / PHY, la cual permite el uso de alternativas MAC y PHY para el transporte de datos de perfil Bluetooth. Para la detección de dispositivos, la conexión y la configuración de perfil se utiliza la radio Bluetooth, mientras que cuando se envían gran cantidad de datos, se utiliza PHY MAC 802.11(generalmente asociados con WI-fi) para transportar los datos. El modo bajo energía se utiliza cuando el sistema está inactivo, y la radio 802.11 cuando se necesita enviar grandes cantidades de datos.

Otra incorporación de esta versión es Unicast de datos sin conexión, que se trata de los datos de los permisos de servicio para ser enviado sin establecer un canal L2CAP. Está diseñada para su uso en aplicaciones que requieren baja latencia entre la acción del usuario y la reconexión / transmisión de datos. El control de energía fue mejorado y se realizó una actualización de la función de control de potencia para eliminar el control de lazo abierto de energía y para aclarar las ambigüedades en el control de energía debido a los esquemas de modulación nuevos añadidos para EDR. Con el control de potencia mejorada se eliminan las ambigüedades, también añade un control de potencia de bucle cerrado. Se añadió una solicitud para "ir directamente a la máxima potencia". En la versión 3.0 de Bluetooth la alta velocidad (AMP) se basa en 802.11, pero el mecanismo de AMP fue diseñado para ser utilizado con otras frecuencias. Originalmente se pensó en UWB pero en Marzo de 2009 la WiMedia Alliance anunció su disolución. El 16 de marzo de 2009, la WiMedia Alliance anunció que iba a llegar a acuerdos de transferencia de las especificaciones de la tecnología WiMedia Ultra-Wideband (UWB). La WiMedia ha transferido todas las especificaciones actuales y futuras. Una vez finalizado la transferencia de tecnología, marketing y cuestiones administrativas, la WiMedia Alliance dejó de operar.

En Octubre de ese mismo año, el Grupo de desarrollo especial de UWB para Bluetooth como parte de la alternativa a MAC / PHY estaba en suspensión. Un grupo pequeño, pero significativo, de antiguos miembros de WiMedia no tenía y no iba a firmar los acuerdos necesarios para la transferencia de propiedad intelectual. El SIG de Bluetooth se encuentra en el proceso de evaluar otras opciones para su plan de acción a largo plazo.

- **Bluetooth v4.0 (2010):** En esta versión el SIG ha completado la especificación del Núcleo de Bluetooth, la cual incluye el Bluetooth clásico, de alta velocidad los protocolos Bluetooth de bajo consumo. El Bluetooth clásico está formado por protocolos Bluetooth legado mientras que el de alta velocidad se basa en Wi-Fi. Fue adoptada el 30 de Junio de 2010. El Bluetooth baja energía (BLE) es un subconjunto de Bluetooth v4.0 con una nueva pila de protocolo para la rápida acumulación de enlaces sencillos. LA versión BLE está dirigida a aplicaciones de potencia muy bajas corriendo una célula de la moneda. Los diseños de chips permiten 2 tipos de implementación, de modo dual y de modo único; y versiones mejoradas de las anteriores.

En modo único, solo esta implementado el protocolo de pila de baja energía. CSR, Nordic Semiconductor y Texas Instruments han dado ha conocer las soluciones del modo único del modo bajo consumo del Bluetooth.

En modo dual, la funcionalidad de bajo consumo de Bluetooth está integrada en un controlador clásico de Bluetooth. En Marzo de 2011, las empresas de semiconductores anunciaron la disponibilidad de chips que cumplían la norma: Atheros, CSR, Broadcom y Texas Instruments. Las acciones realizadas en la arquitectura del Bluetooth clásico son compatibles con todos los dispositivos existentes y sus funcionalidades, causando un coste insignificante respecto al Bluetooth clásico.

El 17 de Diciembre de 2009, El SIG adoptó la tecnología Bluetooth como la principal característica de la versión 4.0. WiBree y Bluetooth ULP (Ultra Low energía) fueron abandonados y durante un tiempo se utilizó el nombre BLE. A finales de 2011, se presentaron los nuevos logotipos como la cara pública de BLE, “Smart Bluetooth Ready” para los anfitriones y “Smart Bluetooth” para los sensores.

El coste reducido de los chips de modo único, que permitía una alta integración y dispositivos compactos, destaca una capa fina de conexión que proporciona un consumo de energía muy pequeño para una operación en modo ocioso, la detección de dispositivos simples y una fiable transferencia de datos punto a multipunto con un avanzado ahorro de energía y seguras conexiones encriptadas con el menor coste posible.

### 2.5.7. Información técnica

La especificación de Bluetooth define un canal de comunicación a un máximo de 720 bit/s, 1 Mbit/s de capacidad bruta, con rango óptimo de 10m, opcionalmente 100m con repetidores.

La frecuencia de radio en la que opera es de 2.4 a 2.8 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos por segundo. Los saltos de frecuencia se dan entre un total de 79 frecuencia con intervalos de 1MHz, lo que permite dar seguridad y robustez.

La potencia de salida para transmitir a una distancia máxima de 10 metros es de 0 dBm, es decir, 1mW. Por su parte, la versión de largo alcance transmite entre 20 y 30 dBm, es decir, entre 100 mW y 1W.

Como objetivo se tenía el conseguir bajo consumo y bajo costo, para ello se ideó como solución implementarlo en un solo chip utilizando circuitos CMOS. De esta forma, se consiguió una solución de 9x9 mm y con un consumo de energía aproximadamente de un 97% menos que un móvil común.

El protocolo de banda base combina conmutación de circuitos y paquetes. Los slots pueden ser reservados por paquetes síncronos, para asegurar que los paquetes no lleguen fuera de orden. Para conseguirlo se utiliza un salto diferente de señal para cada paquete. La conmutación de paquetes puede ser síncrona o asíncrona. Cada canal soporta tres canales de datos síncronos, es decir, voz, o un canal de datos síncrono y otros asíncrono.

Los canales de voz soportan una tasa de transferencia de 64 kbit/s en cada sentido, lo que es suficiente para la transmisión de voz. Por su parte, un canal asíncrono puede transmitir como mucho a 721 kbit/s en una dirección y 56 kbit/s en la contraria. Sin embargo, si el enlace es simétrico, una conexión síncrona puede soportar 432,6 kbit/s en ambas direcciones.

### 2.5.8. Arquitectura Hardware

El dispositivo Bluetooth está compuesto por dos partes:

1. Un dispositivo de radio, encargado de modular y transmitir la señal.
2. Un controlador digital, compuesto por una CPU, un procesador de señales digitales (DSP – Digital Signal Processor) llamado Link Controller, o controlador de Enlace, y las interfaces con el dispositivo anfitrión.

Del procesamiento de la banda base y del manejo de los protocolos ARQ y FEC de la capa dídica se encarga el LC o Link Controller. También se encarga de las funciones de transferencia tanto síncrona como asíncrona, la decodificación de audio y el cifrado de datos.

La CPU se encarga de las instrucciones relacionadas con Bluetooth en el dispositivo anfitrión, para así simplificar su operación. Para conseguirlo, en la CPU corre un software denominado Link Manager, el cual se encarga de la comunicación con otros dispositivos por medio del protocolo LMP.

Micro adaptador bluetooth USB

Entre las tareas realizadas por el LC y el Link Manager, encontramos:

- Envío y recepción de datos.
- Paginación y Peticiones.
- Establecimiento de conexiones.
- Autenticación.
- Negociación y establecimiento de tipos de enlace.
- Establecimiento del tipo de cuerpo de cada paquete.
- Establecer el dispositivo en modo sniff o hold. Sniff realizara lo que se denomina un rastreo de paquetes, es decir, leerá todos los datos que se envíen en esa frecuencia por cualquier otro dispositivo Bluetooth. Mientras que el modo hold se encarga de que el dispositivo se mantenga en una frecuencia aunque no emita ni reciba nada, manteniéndola siempre disponible aunque otros dispositivos la utilicen.

### 2.5.9. Bluetooth VS Wi-Fi

Las tecnologías inalámbricas Bluetooth y Wi-Fi son tecnologías complementarias, es decir, entre las dos tecnologías cubren las necesidades que podamos tener en un entorno doméstico: desde la creación de redes y las labores de impresión a la transferencia de datos entre distintos dispositivos. Ambas tecnologías operan en las bandas de frecuencias no reguladas, banda ISM.

La tecnología Bluetooth se creó para sustituir los cables entre dispositivos cercanos, como puedan ser, los móviles, ordenadores o portátiles dentro de un radio de 10 metros.

Mientras, la tecnología Wi-Fi está más enfocada a ser utilizada como acceso a Ethernet inalámbrico de alta velocidad. Su alcance tiene un mayor radio, por ejemplo, un router típico tiene un radio de aprox. 45m en interiores y 90m en aire libre.

# 3. Práctico

## 3.1. Aplicaciones utilizadas

A continuación, conoceremos los dos programas utilizados para realizar el proyecto. Por un lado, tenemos el Easy Java Simulations (EJS), el cual hemos utilizado para la programación de la parte gráfica del simulador, es decir, la interfaz gráfica, además, de la parte de la simulación que no necesita del Lego NXT. Por otro lado, hemos utilizado el eclipse para realizar la programación del robot, es decir, el manejo del robot dependiendo de la opción de simulación que elija el usuario a través de la interfaz.

### 3.1.1. Easy Java Simulations (EJS)

El principal objetivo del EJS es la realización de pequeñas simulaciones discretas por ordenador, es decir, un programa que intenta reproducir, ya sea con fines pedagógicos o científicos, un fenómeno natural a través de la visualización de los diferentes estados que éste puede presentar. Estos estados se describen por un conjunto de variables, las cuáles varían su valor en el tiempo debido a la iteración de un cierto algoritmo.

En definitiva, el uso más frecuente del EJS es la creación de simulaciones científicas.

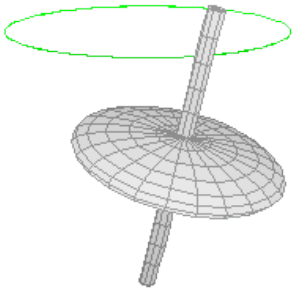
La gran diferencia del EJS como programa de simulación frente a otros programas del mismo tipo, es que no está pensado para facilitar la vida a los programadores profesionales, sino que ha sido creado por profesores de ciencias con la intención de que fuera utilizado por profesores estudiantes de ciencias. En otras palabras, está enfocado a personas que les interesa el fenómeno que se simula y no tanto la programación realizada para conseguir dicha simulación.

Es una herramienta de modelado, cuyo autor se dedica expresamente a esto. Permite a los usuarios trabajar a un alto nivel conceptual, mediante la utilización de un conjunto de herramientas simplificadas y concentrando la mayoría de su tiempo en los aspectos científicos de la simulación, mientras que el ordenador realiza de una forma automática todas las tareas necesarias pero fácilmente automatizables.

EJS nos proporciona la creación de aplicaciones Java que son independientes y multiplataforma, o applets que se pueden visualizar usando cualquier navegador Web, por lo que podemos distribuirlo por internet; las cuáles pueden leer datos a través de la red y ser controlados usando scripts incluidos en las páginas HTML.

Debido a que en el proceso de creación de unas simulaciones existe un valor educativo, puede utilizarse como herramienta pedagógica. Una de los posibles usos es la de ayudar a los estudiantes a que hagan explícitos sus conocimientos y conceptos. Su utilización en grupos

puede servir para mejorar las capacidades de los alumnos para discutir y comunicarse sobre cuestiones científicas.

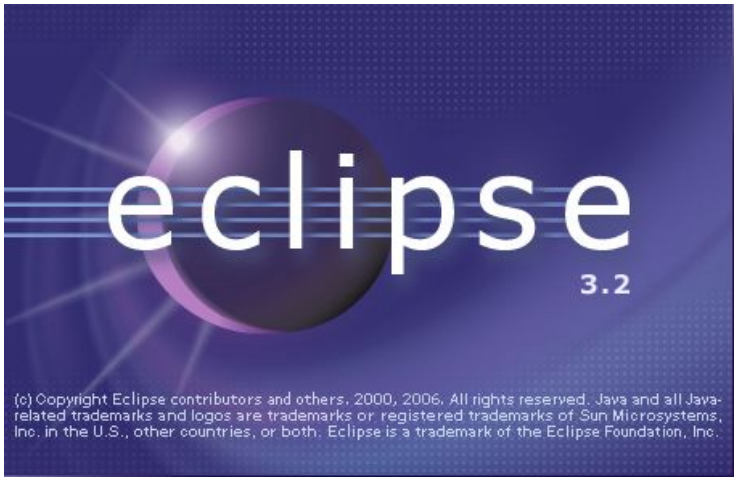


Ejemplo de simulación de EJS

Logotipo des Science Education Prize

## 3.1.2. Eclipse

Eclipse es un programa compuesto por diversas herramientas de programación de código libre, también llamadas de código abierto. Es multiplataforma por lo que podemos importar nuestros proyectos sin ningún tipo de problemas entre diversos sistemas operativos. Está pensado para desarrollar lo que conocemos como “Aplicaciones de Cliente Enriquecido”.




Logotipo de Eclipse

### 3.1.2.1. Los principios

Antes de conocer un poco más en profundidad el eclipse, vamos a hablar de sus comienzos. El Proyecto Eclipse fue creado por IBM en noviembre del año 2001 con el apoyo del consorcio de proveedores de software. Unos años después, en 2004, aparece la llamada Fundación Eclipse, la cuál es una organización sin ánimo de lucro independiente que actuará como administrador de la comunidad Eclipse y da apoyo corporativo a los proyectos de Eclipse y ayuda a cultivar tanto una comunidad de código libre como un ecosistema de productos y servicios complementarios.

Hoy en día Eclipse se considera como una comunidad de personas y empresas que desean colaborar en el comercio del software libre. Los proyectos se centran en la construcción de una plataforma de desarrollo libre formada por marcos extensibles, herramientas y tiempos de ejecución para la construcción, despliegue y gestión a lo largo de su ciclo de vida.

### 3.1.2.2. Arquitectura

Como ya hemos comentado antes Eclipse se basa en la plataforma de cliente enriquecido (Rich Client Platform). Esta plataforma está compuesta por:

- Plataforma Principal – Inicio de Eclipse, ejecución de plugins.
- OSGi – Una plataforma para bunding estándar.
- El Standard Widget Toolkit (SWT) – Un widget toolkit portable.
- JFace – Manejo de archivos, de texto y editores de texto.
- El workbench – Vistas, editores, perspectivas y asistentes.

A diferencias de la mayoría de aplicaciones Java, los widgets están implementados en Standard Widget Toolkit, en vez del estándar Abstract Window Toolkit (AWT) o Swing. La interfaz también es ligeramente diferente ya que dispone de una capa GUI intermedia llamada JFace, la cual nos facilita la construcción de aplicaciones basadas en SWT.

EL IDE o entorno de desarrollo integrado de eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido. Esta es otra diferencia ya que otros entornos monolíticos incluyen todas las funcionalidades, las necesites o no. Este mecanismo de módulos es una plataforma ligera para componentes de software, además de que permite a Eclipse extenderse utilizando otros lenguajes de programación como pueda ser C/C++.

También permite a Eclipse trabajar con procesadores de lenguajes como LaTex, aplicaciones en red como Telnet y Sistema de gestión de base de datos. Se puede escribir cualquier extensión deseada gracias a la arquitectura plugin. Se provee soporte para Java y CVS en el SDK de eclipse.

Si hablamos de las aplicaciones cliente, Eclipse para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc.… provee al programador con frameworks muy ricos.

El SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java, lo cual permite técnicas avanzadas de refactorización y análisis de código. Estas herramientas también están disponibles para otros lenguajes mediante diversos plugins. El IDE también utiliza un espacio de trabajo, en este caso un grupo de metadato en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

#### 3.1.2.3. Características

Eclipse dispone de un Editor de texto con resaltado de sintaxis. La compilación la realiza en tiempo real y posee pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes para la creación de proyectos, clases, etc.… y refactorización.

También, a través de “plugins” libres podemos añadir control de versiones con Subversión e integración de Hibernate.

# 3.2. Ecuaciones utilizadas

Las ecuaciones utilizadas en el EJS para realizar la simulación de una manera realista, han sido obtenidas de la tesina de master *"NAVEGACIÓN DE UN ROBOT MÓVIL DE CONFIGURACIÓN DIFERENCIAL BASADA EN FUNSIÓN SENSORIAL"* realizado por Leonardo Marín P. en el DISA UPV. La parte de la tesina utilizada ha sido el modelo dinámico de 2 partículas.

Como anteriormente pudimos ver, las ecuaciones utilizadas han sido las siguientes:

$$X_k = X_{k-1} + v_{k-1}T_s\cos\left(\theta_{k-1} + \omega_{k-1}\cdot 0{,}5T_{s)}\right.$$

$$Y_k = Y_{k-1} + v_{k-1}T_s\sin\left(\theta_{k-1} + \omega_{k-1}\cdot 0{,}5T_s\right)$$

$$\theta_k = \theta_{k-1} + T_s\omega_{k-1}$$

$$v_k = v_{k-1} + T_s a_{k-1}$$

$$\omega_k = w_{k-1} + T_s\alpha_{k-1}$$

Estas 5 ecuaciones son el modelo dinámico utilizado en esta simulación. Como podemos observar con las 2 primeras ecuaciones hallamos el valor de X e Y actual, las cuáles dependen directamente de los valores de las 3 ecuaciones siguientes, en el instante anterior, donde se calcula la orientación del robot, velocidad lineal y velocidad angular. Como podemos observar también necesitamos la aceleración y la aceleración angular, ambas en el instante anterior.

Para calcular estas variables disponemos de las siguientes ecuaciones.

$$a_{k-1} = 0{,}5a_{R,k-1} + 0{,}5a_{L,k-1}$$
$$\alpha_{k-1} = \lambda\left(0{,}5a_{R,k-1} - 0{,}5\mathrm{a}_{\mathrm{L,k-1}}\right)$$

La velocidad la calculamos como la suma de la mitad de la aceleración en el instante anterior, de esta manera realizamos una media. Mientras que la aceleración angular se cacula como la diferencia entre la aceleración de la rueda derecha y la rueda izquierda, ambas en el instante anterior multiplicadas por lambda.

A continuación, podemos observar las ecuaciones de la aceleración para cada rueda y el cálculo para lambda.

$$a_{R,k} = 1{,}033V_{c,k} - 1{,}033V_{c,k-1} + 0{,}6671a_{R,k-1}$$
$$a_{L,k} = 1{,}033V_{c,k} - 1{,}033V_{c,k-1} + 0{,}6671a_{L,k-1}$$

$$\lambda = \frac{1}{R_{n^2}}$$

$R_n = 0{,}5b$ *donde b es la mitad de la distancia entre ruedas*

## 3.3. Formato trama utilizada

La trama alude al enlace interno entre los distintos elementos que componen un todo. El término se puede utilizar en varios campos, en cada uno de ellos con un significado específico.

Por ejemplo, cuando hablamos de trama de una tela o tejido nos referimos al modo como se disponen sus hilos, formando distintos entramados, de distintos grosores, distintas sensaciones táctiles y visuales. Pero cuando nos referimos a una trama en informática, se conoce como el paquete de datos, que son pequeños paquetes o grupos de información, que se transmiten por internet. En nuestro caso, se transmiten por Bluetooth y se trata de un String, formado por los valores de la tensión de ambas ruedas, *"Tensión_Izq" + ":" + "Tensión_Der" + "\n"*.

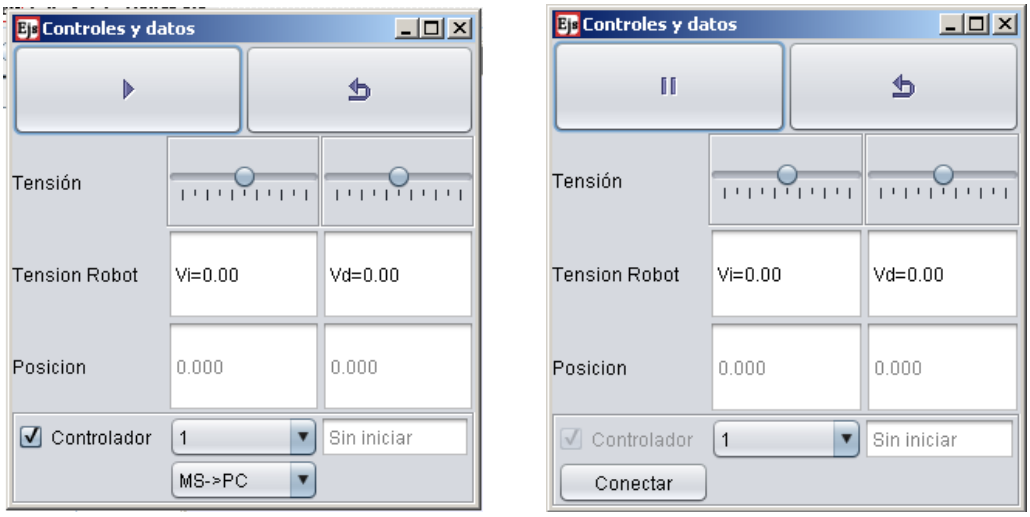
Utilizamos ":" para facilitar la división de las variables correspondientes una vez la información la tenemos donde la queremos, ya que con la función Split (":") lo conseguimos fácilmente, aunque esta función no está incluida en las librerías del NXT por lo que se tiene que hacer a mano.

## 3.4. Aplicación desarrollada en EJS

A continuación, vamos a conocer la aplicación desarrollada tanto por fuera como por dentro, es decir, tanto la interfaz gráfica como el código Java.

### 3.4.1. Interfaz Gráfica

En primer lugar, conoceremos la interfaz gráfica del EJS y cómo utilizarla.




Capturas de la I.G.U. de panel de control. En la primera imagen el simulador está parado mientras que en la segunda esta en marcha

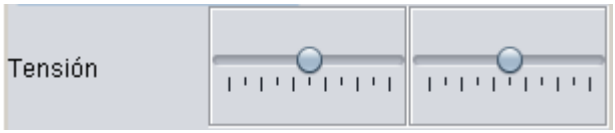
Esta interfaz es la de control de la simulación, es decir, es donde disponemos de toda la información a tiempo real del robot, número de obstáculos y diversas opciones. Está dividida en 4 paneles. La explicación se realizara de arriba abajo y de izquierda a derecha:

- **Panel Encendido**: En este panel podemos observar que hay 2 botones. El primer botón es el de play/pausa, con el que podemos poner en marcha la simulación o detenerla según queramos. Y el segundo, es el restart, el cual, como su propio nombre indica, restablece los valores predeterminados de la simulación.

Captura del panel encendido

- **Panel Tensiones Usuario**: Este panel está formado por una etiqueta y 2 deslizadores. La etiqueta nos indica que son los 2 deslizadores, en este caso se trata de los deslizadores que el usuario puede manipular para variar el valor de las tensiones de los 2 motores del robot.

  Los deslizadores se pueden manipular siempre, excepto, si el simulador se encuentra conectado con el Lego con el tipo de conexión "MS->PC".




Captura del panel Tensiones Usuario

- **Panel Tensiones Reales**: Como en el panel anterior, y el siguiente, encontramos una etiqueta, además de 2 campos numéricos. Los campos numéricos muestran las tensiones reales del robot en cada instante.


Captura del panel Tensiones Reales

- **Panel Posición**: En este caso, podemos observar a parte de la etiqueta, 2 campos numéricos. En este caso los campos numéricos nos indican las posiciones X e Y del robot.

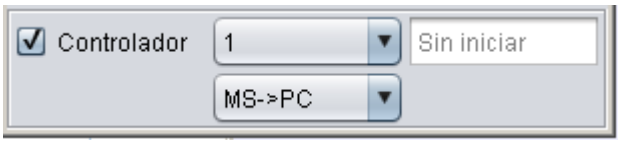

Captura del panel Posición

- **Panel Opciones**: En este último panel podemos encontrar todas las opciones del simulador. El seleccionar o no el campo de controlador, modifica el comportamiento de la simulación respecto a las colisiones. Esta opción solo se tiene en cuanto si el robot no está conectado al simulador. Después encontramos una lista, la cual nos indica el número de obstáculos del simulador, por defecto esta a 1 y puede variar de 0 a 3. Los cambios de número de obstáculos solo se pueden realizar en PAUSE, además, si el robot está conectado por bluetooth no se podrán poner obstáculos.

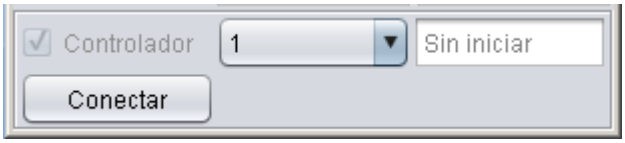
Las siguientes opciones van relacionadas a la conexión bluetooth. El primero es un campo de texto en el que podemos observar el estado de la conexión. Tiene 3 opciones:

1) Sin Iniciar (Por defecto), si no se ha intentado una conexión anteriormente.
2) Error de conexión, si al conectar se ha producido algún error.
3) MS->PC/PC->MS, si se produce correctamente la conexión nos muestra el tipo de conexión realizada.

Nos quedan 2 opciones, las cuáles no pueden estar visibles a la vez. Si el simulador se encuentra en PAUSE podremos modificar el valor de la lista del tipo de conexión que queremos realizar. Si el simulador esta en PLAY, la lista de tipo de conexión desaparecerá y a su lado aparece el botón conectar, que realiza lo que su nombre indica, es decir, intenta la conexión con el Lego NXT mediante Bluetooth.

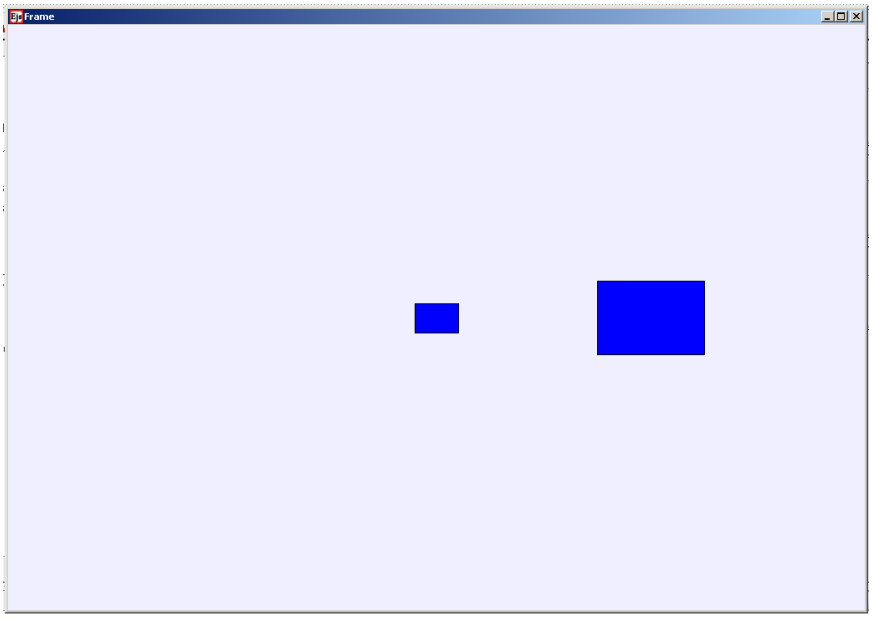



Captura panel Opciones con simulación parada



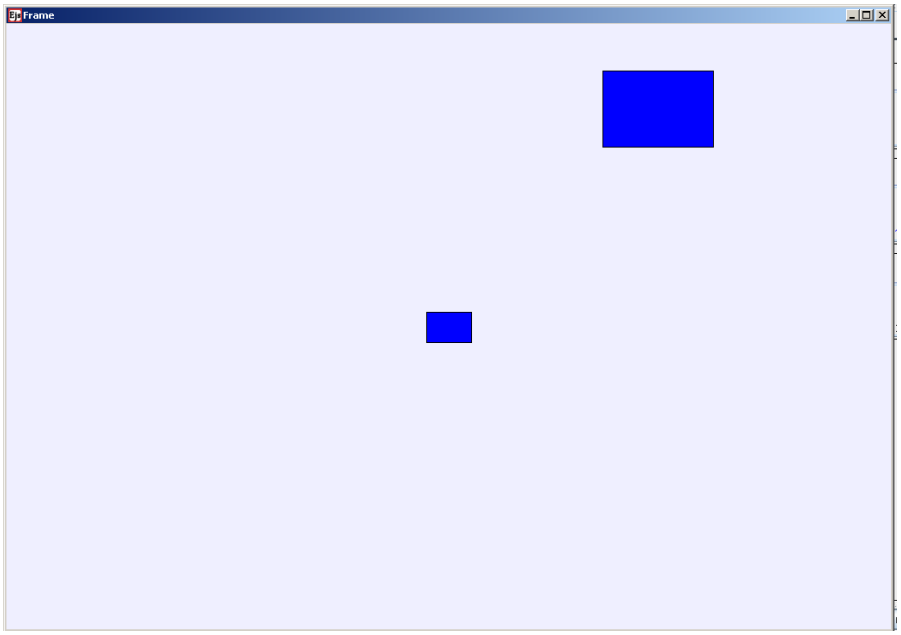

Captura panel Opciones con simulación en marcha

A continuación, podemos observar como es la parte de la interfaz gráfica donde se muestra la simulación según lo elegido en el control de la simulación.




Captura de la interfaz gráfica de la simulación

Como podemos ver, hay 2 rectángulos de distinto tamaño. El cuadrado pequeño simula al robot y el grande a un obstáculo en esta versión 2D. Los obstáculos pueden ser movidos por el usuario a la posición que desee, siempre y cuando, la simulación este en PAUSE. En la siguiente imagen podemos ver esta situación.




Captura de la interfaz gráfica de la simulación con obstáculo cambiado de lugar

Para poder mover la imagen, debemos de mover el puntero al medio del obstáculo que deseamos mover, donde podremos observar como la flechita del puntero cambia por una mano. En ese momento podemos moverlo a la posición deseada.

## 3.4.2. Funciones Importantes

En este apartado vamos a conocer las funciones más importantes para el correcto funcionamiento de la simulación.

- **Colisiones:** En esta función detectamos si el robot esta dentro del rango de colisionar.
- **Evasión Colisiones:** Una vez detectadas las colisiones, realizamos la evasión correspondiente según tengamos, o no, el controlador activo.
- **Controlador:** Realiza la evasión con controlador
- **Conexión Bluetooth:** Realiza la Conexión/Desconexión.
- **Recibir Datos Bluetooth:** Extrae y recalcula los valores que debemos asignar como tensión para la simulación del Lego NXT.
- **Recepción Bluetooth:** Recibe la información del robot.

# 3.4.2.1. Colisiones

Primero comprobamos que la simulación no está conectada con el Lego NXT y que la simulación esté en PLAY. Después, almacenamos la dirección de las ruedas y reseteamos la variable *fObs* al valor 0, esto se realiza para poder volver a tener el control sobre el robot después de la evasión de la colisión con controlador.

Lo siguiente que realizamos es una comprobación de la cercanía del robot con cada uno de los obstáculos y comprobamos si esta dentro del rango especificado según este activo o no el controlador. Si se encuentra en algún caso dentro del rango, ponemos las tensiones de ambas ruedas a 0 para que se paren y almacenamos en fObs_lado, de 1 a 4, el lado por el que vamos a colisionar y fObs lo ponemos a 1 para indicar que hay una colisión.

```
if(num_conec == false && _isPlaying() == true) {

 //fObs == 0

 //Se almacena en variables si la Tensión es positiva para luego ser tratada en una posible colisión

 if(fObs == 0) {

            if(Tension_act_der > 0)

                        aux_T_der = 1;

  else

                        aux_T_der = -1;

             if(Tension_act_izq > 0)

                         aux_T_izq = 1;

            else

                        aux_T_izq = -1;

 }

 //Reiniciamos la variable fObs

 if(fObs == 4){

   fObs = 0;

}
```

```
 for(int i=0;i < num_obs;i++) {

   //Frenado con el controlador

   if(sel_con==true) {

    //Izquierda Obstáculo

    if((fObs == 0) && (Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <=
  dist_seg || Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg)
  && (((Y_act + BaseY/2 <= Pos_obs_y[i] + Tam_obs_y[i]/2 + dist_seg) && (Y_act +
  BaseY/2 >= Pos_obs_y[i] - Tam_obs_y[i]/2 - dist_seg)) || (Y_act - BaseY/2 >=
  Pos_obs_y[i] - Tam_obs_y[i]/2 - dist_seg) && (Y_act - BaseY/2 <= Pos_obs_y[i] +
  Tam_obs_y[i]/2 + dist_seg))) {

      Tension_act_der = 0;

      Tension_act_izq = 0;

      fObs = 1;

fObs_lado = 1;

     }

     //Derecha Obstaculo

     else if((fObs == 0) && (Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2))
  <= dist_seg || Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <=
  dist_seg) && (((Y_act + BaseY/2 <= Pos_obs_y[i] + Tam_obs_y[i]/2 + dist_seg) && (Y_act
  + BaseY/2 >= Pos_obs_y[i] - Tam_obs_y[i]/2 - dist_seg)) || (Y_act - BaseY/2 >=
  Pos_obs_y[i] - Tam_obs_y[i]/2 - dist_seg) && (Y_act - BaseY/2 <= Pos_obs_y[i] +
  Tam_obs_y[i]/2 + dist_seg))) {

      Tension_act_der = 0;

      Tension_act_izq = 0;

      fObs = 1;

      fObs_lado = 2;

     }

     //Arriba Obstáculo

     else if((fObs == 0) && (Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2))
  <= dist_seg || Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <=
  dist_seg) && (((X_act + BaseX/2 <= Pos_obs_x[i] + Tam_obs_x[i]/2 + dist_seg) && (X_act
  + BaseX/2 >= Pos_obs_x[i] - Tam_obs_x[i]/2 - dist_seg)) || (X_act - BaseX/2 >=
```

```
Pos_obs_x[i] - Tam_obs_x[i]/2 - dist_seg) && (X_act - BaseX/2 <= Pos_obs_x[i] +
Tam_obs_x[i]/2 + dist_seg))) {

      Tension_act_der = 0;

      Tension_act_izq = 0;

      fObs = 1;

      fObs_lado = 3;

    }

    //Abajo Obstaculo

    else if((fObs == 0) && (Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] - Tam_obs_y[i]/2))
<= dist_seg || Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] - Tam_obs_y[i]/2)) <= dist_seg)
&& (((X_act + BaseX/2 <= Pos_obs_x[i] + Tam_obs_x[i]/2 + dist_seg) && (X_act +
BaseX/2 >= Pos_obs_x[i] - Tam_obs_x[i]/2 - dist_seg)) || (X_act - BaseX/2 >=
Pos_obs_x[i] - Tam_obs_x[i]/2 - dist_seg) && (X_act - BaseX/2 <= Pos_obs_x[i] +
Tam_obs_x[i]/2 + dist_seg))) {

      Tension_act_der = 0;

      Tension_act_izq = 0;

      fObs = 1;

      fObs_lado = 4;

    }

  }

  //Frenado sin controlador

  else {

    //Izquierda Obstaculo1

    if((fObs == 0) && (Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <=
dist_sc || Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_sc) &&
(((Y_act + BaseY/2 <= Pos_obs_y[i] + Tam_obs_y[i]/2 + dist_sc) && (Y_act + BaseY/2 >=
Pos_obs_y[i] - Tam_obs_y[i]/2 - dist_sc)) || (Y_act - BaseY/2 >= Pos_obs_y[i] -
Tam_obs_y[i]/2 - dist_sc) && (Y_act - BaseY/2 <= Pos_obs_y[i] + Tam_obs_y[i]/2 +
dist_sc))) {

      Tension_act_der = 0;

      Tension_act_izq = 0;

      fObs = 1;
```

```
        fObs_lado = 1;

        X_act = (Pos_obs_x[i] - Tam_obs_x[i]/2);

      }

      //Derecha Obstaculo

      else if((fObs == 0) && (Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2))
<= dist_sc || Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_sc)
&& (((Y_act + BaseY/2 <= Pos_obs_y[i] + Tam_obs_y[i]/2 + dist_sc) && (Y_act + BaseY/2
>= Pos_obs_y[i] - Tam_obs_y[i]/2 - dist_sc)) || (Y_act - BaseY/2 >= Pos_obs_y[i] -
Tam_obs_y[i]/2 - dist_sc) && (Y_act - BaseY/2 <= Pos_obs_y[i] + Tam_obs_y[i]/2 +
dist_sc))) {

        Tension_act_der = 0;

        Tension_act_izq = 0;

        fObs = 1;

        fObs_lado = 2;

      }

      //Arriba Obstaculo

      else if((fObs == 0) && (Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2))
<= dist_sc || Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_sc)
&& (((X_act + BaseX/2 <= Pos_obs_x[i] + Tam_obs_x[i]/2 + dist_sc) && (X_act + BaseX/2
>= Pos_obs_x[i] - Tam_obs_x[i]/2 - dist_sc)) || (X_act - BaseX/2 >= Pos_obs_x[i] -
Tam_obs_x[i]/2 - dist_sc) && (X_act - BaseX/2 <= Pos_obs_x[i] + Tam_obs_x[i]/2 +
dist_sc))) {

        Tension_act_der = 0;

        Tension_act_izq = 0;

        fObs = 1;

        fObs_lado = 3;

      }

      //Abajo Obstaculo

      else if((fObs == 0) && (Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] - Tam_obs_y[i]/2))
<= dist_sc || Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] - Tam_obs_y[i]/2)) <= dist_sc)
&& (((X_act + BaseX/2 <= Pos_obs_x[i] + Tam_obs_x[i]/2 + dist_sc) && (X_act + BaseX/2
>= Pos_obs_x[i] - Tam_obs_x[i]/2 - dist_sc)) || (X_act - BaseX/2 >= Pos_obs_x[i] -
```

```
Tam_obs_x[i]/2 - dist_sc) && (X_act - BaseX/2 <= Pos_obs_x[i] + Tam_obs_x[i]/2 +
dist_sc))) {
      Tension_act_der = 0;
      Tension_act_izq = 0;
      fObs = 1;
      fObs_lado = 4;
    }
  }
 }
}
```

## 3.4.2.2. Evasión Colisiones

Al igual que en *Colisiones*, realizamos en primer lugar la misma comprobación que anteriormente. Ahora, dependiendo de si esta activo el controlador o no, decidimos qué camino tomar con un "if () /else".

Si el controlador esta activo llamamos a la función controlador (). En caso contrario, según por el lado del robot que se colisionó y la tensión que llevaban las ruedas en dicho instante, se pone fObs a 0 y comprobamos que las tensiones actuales no son iguales en signo a los que produjeron la colisión, en cuyo caso pondremos las tensiones a 0 para evitar una nueva colisión. Si la comprobación anterior no ocurre lo que realizaremos es una nueva comprobación, como la de la función *Colisiones*, para evitar que este nuevo movimiento produzca otra colisión.

```
if(num_conec == false && _isPlaying() == true) {
 if(sel_con == true) {
  controlador ();
 }
 else {
  // fObs = 1;
  if(fObs == 1 ) {
```

```
for(int i=0;i < num_obs;i++) {
 if(fObs_lado == 1) {
  //Ruedas + +
  if(aux_T_der == 1 && aux_T_izq == 1) {
   fObs = 0;
   if(Tension_act_der > 0) {
    Tension_act_der = 0;
   }
   if(Tension_act_izq > 0) {
    Tension_act_izq = 0;
   }
   if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i]- Tam_obs_x[i]/2)) <= dist_seg)
   fObs = 1;
  }
  //Ruedas - -
  if(aux_T_der == -1 && aux_T_izq == -1) {
   fObs = 0;
   if(Tension_act_der < 0) {
    Tension_act_der = 0;
    fObs = 1;
   }
   if(Tension_act_izq < 0) {
    Tension_act_izq = 0;
    fObs = 1;
   }
```

```
            if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg)

            fObs = 1;

           }

           //Ruedas + -

           if(aux_T_der == 1 && aux_T_izq == -1) {

            fObs = 0;

            if(Tension_act_der > 0) {

              Tension_act_der = 0;

              fObs = 1;

            }

            if(Tension_act_izq < 0) {

              Tension_act_izq = 0;

              fObs = 1;

            }

            if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg)

            fObs = 1;

           }

           //Ruedas - +

           if(aux_T_der == -1 && aux_T_izq == 1) {

            fObs = 0;

            if(Tension_act_der < 0) {

              Tension_act_der = 0;

              fObs = 1;

            }

            if(Tension_act_izq > 0) {

              Tension_act_izq = 0;
```

```
        fObs = 1;
      }
      if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] - Tam_obs_x[i]/2)) <= dist_seg)
      fObs = 1;
     }
    }
    if(fObs_lado == 2) {
     //Ruedas + +
     if(aux_T_der == 1 && aux_T_izq == 1) {
      fObs = 0;
      if(Tension_act_der > 0) {
       Tension_act_der = 0;
      }
      if(Tension_act_izq > 0) {
       Tension_act_izq = 0;
      }
      if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg)
      fObs = 1;
     }
     //Ruedas - -
     if(aux_T_der == -1 && aux_T_izq == -1) {
      fObs = 0;
      if(Tension_act_der < 0) {
       Tension_act_der = 0;
       fObs = 1;
      }
```

```
            if(Tension_act_izq < 0) {
              Tension_act_izq = 0;
              fObs = 1;
            }
            if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg)
            fObs = 1;
          }
          //Ruedas + -
          if(aux_T_der == 1 && aux_T_izq == -1) {
            fObs = 0;
            if(Tension_act_der > 0) {
              Tension_act_der = 0;
              fObs = 1;
            }
            if(Tension_act_izq < 0) {
              Tension_act_izq = 0;
              fObs = 1;
            }
            if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg)
            fObs = 1;
          }
          //Ruedas - +
          if(aux_T_der == -1 && aux_T_izq == 1) {
            fObs = 0;
            if(Tension_act_der < 0) {
              Tension_act_der = 0;
```

```
          fObs = 1;
        }
        if(Tension_act_izq > 0) {
          Tension_act_izq = 0;
          fObs = 1;
        }
        if(Math.abs((X_act + BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg ||
Math.abs((X_act - BaseX/2) - (Pos_obs_x[i] + Tam_obs_x[i]/2)) <= dist_seg)
        fObs = 1;
       }
      }
      if(fObs_lado == 3) {
       //Ruedas + +
       if(aux_T_der == 1 && aux_T_izq == 1) {
        fObs = 0;
        if(Tension_act_der > 0) {
          Tension_act_der = 0;
        }
        if(Tension_act_izq > 0) {
          Tension_act_izq = 0;
        }
        if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg)
        fObs = 1;
       }
       //Ruedas - -
       if(aux_T_der == -1 && aux_T_izq == -1) {
        fObs = 0;
```

```
            if(Tension_act_der < 0) {
              Tension_act_der = 0;
              fObs = 1;
            }
            if(Tension_act_izq < 0) {
              Tension_act_izq = 0;
              fObs = 1;
            }
            if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg)
              fObs = 1;
           }
           //Ruedas + -
           if(aux_T_der == 1 && aux_T_izq == -1) {
            fObs = 0;
            if(Tension_act_der > 0) {
              Tension_act_der = 0;
              fObs = 1;
            }
            if(Tension_act_izq < 0) {
              Tension_act_izq = 0;
              fObs = 1;
            }
            if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg)
              fObs = 1;
           }
```

```
          //Ruedas - +
          if(aux_T_der == -1 && aux_T_izq == 1) {
            fObs = 0;
            if(Tension_act_der < 0) {
              Tension_act_der = 0;
              fObs = 1;
            }
            if(Tension_act_izq > 0) {
              Tension_act_izq = 0;
              fObs = 1;
            }
            if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] + Tam_obs_y[i]/2)) <= dist_seg)
            fObs = 1;
          }
        }
        if(fObs_lado == 4) {
          //Ruedas + +
          if(aux_T_der == 1 && aux_T_izq == 1) {
            fObs = 0;
            if(Tension_act_der > 0) {
              Tension_act_der = 0;
            }
            if(Tension_act_izq > 0) {
              Tension_act_izq = 0;
            }
            if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[i] - Tam_obs_y[i]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[i] - Tam_obs_y[i]/2)) <= dist_seg)
```

```
  fObs = 1;
 }
 //Ruedas - -
 if(aux_T_der == -1 && aux_T_izq == -1) {
  fObs = 0;
  if(Tension_act_der < 0) {
   Tension_act_der = 0;
   fObs = 1;
  }
  if(Tension_act_izq < 0) {
   Tension_act_izq = 0;
   fObs = 1;
  }
  if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[0] - Tam_obs_y[0]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[0] - Tam_obs_y[0]/2)) <= dist_seg)
  fObs = 1;
 }
 //Ruedas + -
 if(aux_T_der == 1 && aux_T_izq == -1) {
  fObs = 0;
  if(Tension_act_der > 0) {
   Tension_act_der = 0;
   fObs = 1;
  }
  if(Tension_act_izq < 0) {
   Tension_act_izq = 0;
   fObs = 1;
  }
```

```
            if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[0] - Tam_obs_y[0]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[0] - Tam_obs_y[0]/2)) <= dist_seg)
             fObs = 1;
            }
            //Ruedas - +
            if(aux_T_der == -1 && aux_T_izq == 1) {
             fObs = 0;
             if(Tension_act_der < 0) {
               Tension_act_der = 0;
               fObs = 1;
             }
             if(Tension_act_izq > 0) {
               Tension_act_izq = 0;
               fObs = 1;
             }
             if(Math.abs((Y_act + BaseY/2) - (Pos_obs_y[0] - Tam_obs_y[0]/2)) <= dist_seg ||
Math.abs((Y_act - BaseY/2) - (Pos_obs_y[0] - Tam_obs_y[0]/2)) <= dist_seg)
             fObs = 1;
            }
           }
          }
         }
        }
       }
```

# 3.4.2.3. Controlador

En primer lugar comprobamos que el valor de fObs sea mayor o igual a 0. Si esto se cumple, ponemos las tensiones con valor 2.5 con signo contrario al que ocasionó la colisión. Después, sumamos 1 a fObs y ponemos fObs_lado a 0. La suma a fObs se realiza para conseguir que durante un corto período de tiempo el usuario no pueda actualizar los valores de las tensiones y de esta forma aseguramos que el robot esta a una distancia segura del obstáculo.

```
public void controlador () {
 //Aplicamos Tensión necesaria para salir de la situación de colisión
 if(fObs >= 1){
   if(aux_T_der == 1 && aux_T_der == 1){
     Tension_act_der = -2.5;
     Tension_act_izq = -2.5;
     }
   else if(aux_T_der == -1 && aux_T_der == 1){
     Tension_act_der = 2.5;
     Tension_act_izq = -2.5;
     }
   else if(aux_T_der == -1 && aux_T_der == -1){
     Tension_act_der = 2.5;
     Tension_act_izq = 2.5;
     }
   else if(aux_T_der == 1 && aux_T_der == -1){
     Tension_act_der = -2.5;
     Tension_act_izq = 2.5;
     }
  fObs++;
  fObs_lado = 0; }}
```

# 3.4.2.4. Conexión Bluetooth

Dentro de esta función tenemos tanto la conexión como la desconexión. Es llamada desde el botón de conectar. Lo primero que se realiza es la declaración de nxtComm = null. A continuación, tenemos un "if()/else" para diferenciar entre la conexión o desconexión. Si el if cumple las condiciones de que el robot no está conectado con el robot y que la simulación está en PLAY, entramos en la conexión.

Lo primero hacemos es la asignación a nxtComm de un NXTCommFactory dentro de un cuerpo try/catch.

**CONEXIÓN**

Creamos la variable nxtInfo donde creamos la conexión poniéndole tipo NXTCommFactory.BLUETOOTH, nxt como nombre y por último la dirección MAC del robot a enlazar y la variable opened con valor inicial *false.*

A continuación, comprobamos si la conexión se ha realizado correctamente y, en caso contrario, mostramos un error por pantalla. Si la conexión se ha realizado correctamente a num_conec le asignamos valor true, llamamos a la función *resetear*_Variables, la cuál resetea algunas variables como su nombre indica y, mostramos por pantalla "Conectado a nxt".

Por último, creamos todas las variables necesarias para la lectura/escritura del bluetooth, como por ejemplo is1. Una vez tenemos las variables, comprobamos que tipo de conexión se ha realizado para poder enviarle un 0 o un 1 para que sepa si debe escuchar o enviar, es decir, si el simulador escucha al robot (MS->PC) o por el contrario es el robot el que escucha al simulador (PC->MS). En caso de que el tipo no sea una de las 2 establecidas mostramos un error por pantalla.

**DESCONEXIÓN**

Al igual que en conexión realizamos una comprobación. En este caso se debe cumplir que num_conec tiene que ser *true* y que la simulación este en PLAY. Si se cumple la comprobación, lo primero que hacemos es mostrar por pantalla un mensaje avisando de la desconexión. Después se hace la desconexión y, por último, ponemos num_conec a *false* y llamamos a *resetear_Variables.*

```
public void conexion_Bluetooth() {

 //NXTComm

 nxtComm = null;

//CONEXIÓN DEL BLUETOOTH

 if(num_conec == false && _isPlaying() == true) {
```

```
        System.out.println("Conectando");
        try {
          nxtComm = NXTCommFactory.createNXTComm(NXTCommFactory.BLUETOOTH);
        } catch(NXTCommException e) {
        }
        NXTInfo nxtInfo =  new NXTInfo(NXTCommFactory.BLUETOOTH ,"NXT",
"00:16:53:13:7e:a0");
        boolean opened = false;
        try {
          opened = nxtComm.open(nxtInfo);
        } catch(NXTCommException e) {
        }
        if (!opened) {
          System.out.println("Failed to open " + nxtInfo.name);
          Conectado = "Error de conexión";
        }
        else {
          num_conec = true;
          reseteo_Variables ();
          System.out.println("Conectado a " + nxtInfo.name);
        }
        InputStream is1 = nxtComm.getInputStream();
        OutputStream os1 = nxtComm.getOutputStream();
        dos1 = new DataOutputStream(os1);
        dis1 = new DataInputStream(is1);

        try {
          if(Tipo_conec == "MS->PC") {
```

```
        dos1.writeInt(1);
        dos1.flush();
        Conectado = "MS->PC";
       }
       else{
        dos1.writeInt(0);
        dos1.flush();
        Conectado = "PC->MS";
       }
      }
      catch(IOException c) {
       System.out.println("ERROR: Tipo conexion no existente");
      }
     }
     //DESCONEXIÓN DEL BLUETOOTH
     else if(num_conec == true && _isPlaying() == true) {
      System.out.println("Desconectando");
      try {
       nxtComm.close();
      } catch(IOException c) {
      }
      num_conec = false;
      reseteo_Variables ();
     }
    }
```

# 3.4.2.5. Recibir Datos Bluetooth

En primer lugar realizamos una comprobación, en la cual num_conec debe ser *true*, la simulación debe estar en PLAY y el tipo de conexión debe ser MS->PC. Una vez se cumple la comprobación, asignamos a la variable global cadena el valor de retorno de la función recepcionBluetooth, la cual veremos a continuación. Una vez tenemos los datos recibidos en la cadena, le realizamos un Split y el resultado se lo asignamos a la variable global datos. Por último, realizamos un escalado de los valores de datos y se lo asignamos a las tensiones. El escalado se realiza debido a que las tensiones en el Lego varían de 0 a 100, mientras que en el robot va de -5 a 5.

```
if(num_conec == true && _isPlaying() == true && Tipo_conec == "MS->PC"){
 //Una vez tenemos los datos debemos separarlos
 try{
 cadena = recepcionBluetooth();
 } catch(IOException c) {
  System.out.println("Excepcion Recibir datos bluetooth");
 }
 datos = cadena.split(":");
 //Ahora asignamos los datos a sus variables convirtiendolos a tipo Double
 Tension_act_izq = (5 * Double.valueOf(datos[0])) / 100;
 Tension_act_der = (5 * Double.valueOf(datos[1])) / 100;
 }
```

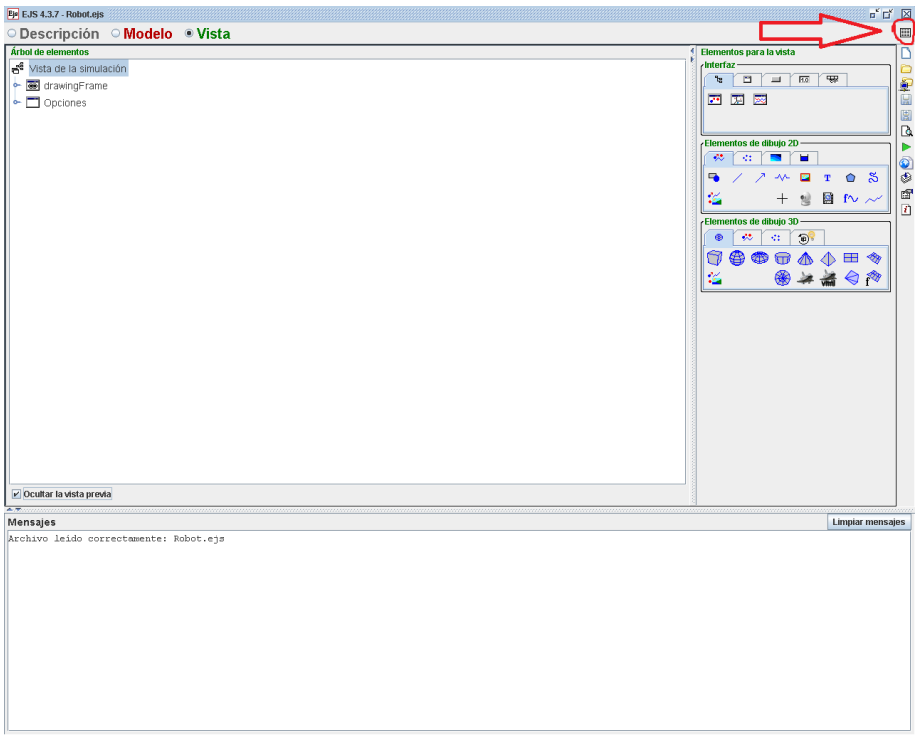
# 3.4.2.6. Recepción Bluetooth

Primero creamos un buffer de String llamado strb, y leemos el primer carácter en la variable c. Ahora entramos en un bucle while donde leemos carácter a carácter y siempre que no sea el carácter especial \n vamos añadiendo el valor de c al buffer. Y, una vez terminado el bucle, devolvemos strb pero haciéndole un casting a String.

```
public String recepcionBluetooth () throws IOException {
    StringBuffer strb = new StringBuffer();
    char c = dis1.readChar();
    while(c != '\n') {
       strb.append(c);
       c = dis1.readChar();
    }
    return strb.toString();
```

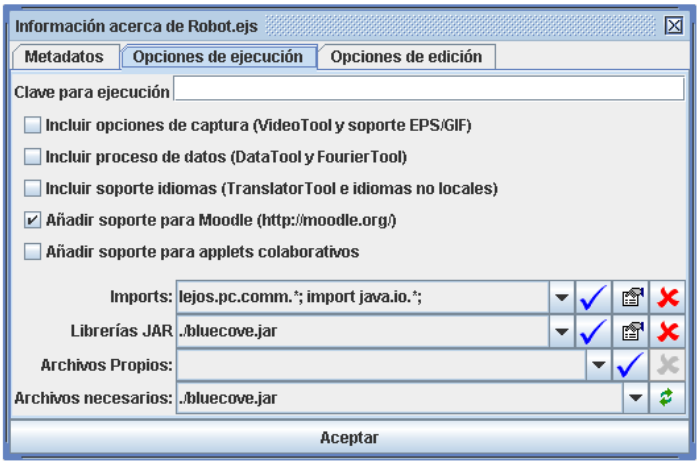
# 3.5. Añadir librerias externas a EJS

A diferencia de cualquier programa para programar como pueda ser visual studio o eclipse, donde las librerías se añaden por línea de código, en EJS se añaden mediante una opción que encontramos *Información del proyecto.*




Captura de la interfaz del EJS. La flecha nos señala opción para añadir librerías externas

En la ventana que se nos mostrará a continuación deberemos seleccionar la pestaña *Opciones de ejecución*, donde nos aparecererá en las opciones de debajo del todo la opción *Librerías JAR y Archivos Necesarios* desde donde seleccionamos nuestra librería externa que queremos añadir.




Captura del panel de información de proyecto

# 4. Conclusiones

## 4.1. Problemas encontrados

El problema más importante encontrado ha sido realizar una correcta elección de los elementos ha utilizar en la simulación debido a ciertos requisitos que debiamos satisfacer y las propiedades que estos elementos nos proporcionaban.

La elección de una librería de bluetoth que funcionara correctamente con el EJS, finalmente se nos aconsejo la utilización de la librería *bluecove.*

Y por útimo, algún problema con el cambio de los valores de la trama de String a Float, debido a algún carácter extra que añadíamos de forma involuntaria.

## 4.2. Posibles usos de la aplicación

Debido a que es un simulador todos sus posibles usos se centran en probar el correcto funcionamiento del robot simulado, ya sea como comprobación antes de salir a la venta o como para solucionar algún problema encontrado. Podemos comprobar si la trayectoria hecha en el simulador es la que esperábamos.

Otro posible uso sería en la enseñanza para tener un primer contacto con un pequeño simulador fácil de utilizar.

# 5. Bibliografía


**METODOS DE INTEGRACION NUMERICA**

1. Wikipedia
2. Programando con café
3. Pacop
4. Tema 2 de asignatura SSD (Optativa de 3er Curso de Ingeniería en Informática)

**ROBOTS**

1. Wikipedia
2. Alegsa
3. Definición
4. Seguritron
5. Slideshare

**HISTORIA DE LOS ROBOTS**

1. Rincón del vago
2. MundoTech
3. RoboticsSpot
4. Robodosis

**LEGO NXT**

1. Wikipedia
2. Ro-botica
3. Gizmos
4. Página Oficial Lego Mindstorms

**BLUETOOTH**

1. Wikipedia
2. Masadelante

**SIMULADORES**

1. Definición
2. Ecured
3. Wikipedia

**TRAMA UTILIZADA**

1. deconceptos

**ECLIPSE**

1. Wikipedia
2. Eclipse

**EJS**

1. Pagina oficial EJS
2. Wikipedia