ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD POLITÉCNICA DE VALENCIA

TRANSPORTE Y DESPLAZAMIENTO DE OBJETOS MEDIANTE COORDINACIÓN DE ROBOTS MÓVILES

PROYECTO FIN DE CARRERA

Realizado por:

Bernat Escolá Ferrer

Dirigido por:

Dr. D. Ángel Valera Fernández

Dto. Ingeniería de Sistemas y Automática

VALENCIA, NOVIEMBRE DE 2011

Índice

1.- INTRODUCCIÓN

1.1 Introducción y justificación

Desde los tiempos en que aparecieron los primeros mecanismos, el ser humano no ha cesado de buscar y evolucionar instrumentos o máquinas para poder mejorar los niveles de vida de la sociedad en la que viven. De hecho, en la época actual en la que vivimos, es complicado encontrar alguna actividad tanto profesional como personal en la que no encontremos alguna máquina a fin de hacernos la vida más sencilla.

De todos los mecanismos que podríamos encontrar en nuestras vidas, la Robótica y la Automatización es una ciencia que justamente está realizando un papel verdaderamente importante en lo que es el ámbito tecnológico. La tecnología mecánica y electrónica ha evolucionado, y dejando a un lado los estereotipos de robots como máquinas que acaban descontrolándose y tomando iniciativa propia, poco a poco se han ido implantando en la sociedad como herramientas que facilitan tareas o resuelven problemas.

Pese a los grandes avances y las grandes investigaciones, todavía faltan muchísimas cosas por descubrir, ya que la complejidad que existe para que los robots interactúen con el entorno al que están sometidos es de un grado altísimo, pero que sirve de autentica motivación para los científicos e ingenieros que están investigando en dicho campo de la tecnología, la integración de visión artificial, reconocimiento de formas o la planificación de trayectorias, son algunos de los retos en la línea de investigación de la robótica.

Hoy en día, existen infinidad de tareas en las que la ejecución de las mismas puede resultar peligrosa para el ser humano, o que la precisión y velocidad que pueda dar una máquina, jamás podría conseguirla el ser humano. Es aquí donde la robótica nos tiende su mano, y nosotros como investigadores, deberemos trabajar duramente para poder alcanzar retos, y al mismo tiempo descubrir nuevas motivaciones y afrontar nuevas dificultades.

1.2 Objetivos

El principal objetivo es la generación de trayectorias para la coordinación y el control de robots móviles, en concreto, se utilizarán robots Lego Mindstorms NXT.

Los objetivos específicos del presente Proyecto son:

- Generar e implementar trayectorias de robots mediante aproximación de puntos basados en curvas de Bezier de 4 y/o 5 puntos.
- Implementación de las curvas para uno, dos o tres robots móviles simultáneamente.
- Abordar el concepto de comunicación Bluetooth entre robots móviles para la realización de trayectorias coordinadas.
- Programar algoritmos de control cinemático para robots móviles. Dichos algoritmos están basados en el control de trayectorias mediante algoritmos de Control de Posición por Punto Descentralizado.

2.- DESARROLLO TEÓRICO

2.1 Introducción a la robótica

El término "robot" se debe a Karel Capek, quien lo utilizó en 1917 por primera vez, para denominar a unas máquinas construidas por el hombre y dotadas de inteligencia. Deriva de “robota” que en checo significa trabajo o prestación personal o de "robotnik" que define al esclavo de trabajo.

Joseph Engelberger, un pionero en la industria robótica, expresó claramente esta idea con su frase: "No puedo definir un robot, pero reconozco uno cuando lo veo".

La definición adoptada por la Robotics Industries Association (RIA), anteriormente el Robotics Institute of América, aceptada internacionalmente para Robot es:

- Manipulador multifuncional y reprogramable, diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos programados y variables que permiten llevar a cabo diversas tareas.

Otra definición con la que nos podemos encontrar es:

- Un robot es un dispositivo generalmente mecánico, que desempeña tareas automáticamente, ya sea de acuerdo a supervisión humana directa, a través de un programa predefinido o siguiendo un conjunto de reglas generales, utilizando técnicas de inteligencia artificial. Generalmente estas tareas reemplazan, asemejan o extienden el trabajo humano, como ensamble en líneas de manufactura, manipulación de objetos pesados o peligrosos, trabajo en el espacio, etc.

También hay definiciones de lo que es un robot basándose en las propiedades o características que deben poseer:

- Un robot, es un agente artificial mecánico o virtual. Es una máquina usada para realizar un trabajo automáticamente y que es controlada por una computadora.

En resumen se puede decir que:

- Su característica fundamental es poder manejar/manipular objetos. Un robot se diseña con este fin, teniendo en cuenta que ha de ser muy versátil a la hora de utilizar herramientas y manejarlas.
- La segunda peculiaridad que lo diferencia de otras máquinas automáticas es su capacidad para realizar trabajos completamente diferentes adaptándose al medio e incluso pudiendo tomar decisiones, o sea, que es multifuncional y reprogramable.

En general, un robot, para ser considerado como tal, debería presentar algunas de estas propiedades:

- No es natural, sino que ha sido creado artificialmente.
- Puede sentir su entorno.
- Puede manipular cosas de su entorno.
- Tiene cierta inteligencia o habilidad para tomar decisiones basadas en el ambiente o en una secuencia pre programada automática.
- Es reprogramable.
- Puede moverse en uno o más ejes de rotación o traslación.
- Puede realizar movimientos coordinados.

Una vez comprendido el concepto de robot podemos avanzar hacia la definición de la ciencia que estudia este tipo de dispositivos, la Robótica, que ha evolucionado rápidamente en estos últimos años.

Según lo visto una posible definición de Robótica podría ser:

El diseño, fabricación y utilización de máquinas automáticas programables con el fin de realizar tareas repetitivas como el ensamble de automóviles, aparatos, etc. y otras actividades.

Básicamente, la robótica se ocupa de todo lo concerniente a los robots, lo cual incluye el control de motores, mecanismos automáticos o neumáticos, sensores, sistemas de cómputos, etc.

De esta definición podemos concluir que la Robótica es una tecnología multidisciplinar porque hace uso de los recursos que le proporcionan otras ciencias, ya que en el proceso de diseño y construcción de un robot intervienen muchos campos pertenecientes a otras ramas de la ciencia, como por ejemplo la Mecánica, la Electrónica, la Automática, la Informática, la Matemática, etc.

2.2 Historia de la robótica

El hombre durante toda su historia ha convivido con el deseo de construir máquinas capaces de realizar tareas que facilitasen su trabajo. Desde hace cientos de años antes de Cristo, ya se intentaban crear dispositivos, denominados artefactos o máquinas, que tuvieran un movimiento sin fin y que no estuvieran controlados ni supervisados por personas.

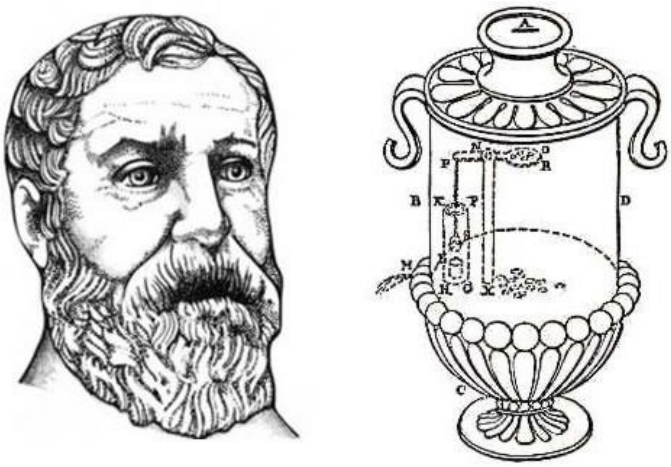
Los primeros autómatas que aparecen en la historia son ingenios mecánicos, más o menos complicados, que desarrollaban una tarea de forma continua. Sin embargo, las primeras máquinas construidas no tenían una utilidad práctica, sino que su principal objetivo era entretener a sus dueños. Generalmente funcionaban por medio de movimientos ascendentes de aire o agua caliente. El vertido progresivo de un líquido o la caída de un peso provocaba rupturas de equilibrio en diversos recipientes provistos de válvulas; otros mecanismos se basaban en palancas o contrapesos. Mediante sistemas de este tipo se construían pájaros artificiales que podían “cantar" o "volar", o puertas que se abrían solas.

El origen de los autómatas se remonta al Antiguo Egipto, donde las estatuas de algunos de sus dioses despedían fuego por los ojos, poseían brazos mecánicos manejados por los sacerdotes del templo o emitían sonidos cuando los rayos del sol los iluminaba. Estos ingenios pretendían causar temor y respeto entre la gente del pueblo.

Hacia el año 1300 a. C., *Amenhotep,* hijo de Hapu, hace construir una estatua de Memon, rey de Etiopía, que emite sonidos cuando la iluminan los rayos del sol al amanecer. Los egipcios desarrollaron modelos matemáticos muy avanzados y construyeron automatismos muy sofisticados, como el reloj de agua. Se tiene constancia de la existencia del ábaco ya entre el año 1000 y 500 a d C, aunque existen dudas sobre si fue en Babilonia o en China dónde fue inventado. Este ingenio matemático permitió el desarrollo de la computación y la inteligencia artificial que fueron desarrollándose paralelamente al interés por los automatismos y el diseño de máquinas imitadoras del ser humano.

En la mitología romana, Vulcano construye ingenios mecánicos que utiliza como sirvientes; mientras que en la hebrea el Gólem cobra vida con una combinación de palabras en el contexto de la magia cabalística.

Sin embargo, los primeros datos descriptivos acerca de la construcción de un autómata aparecen en el siglo I. El matemático, físico e inventor griego Herón de Alejandría describe múltiples ingenios mecánicos en su libro *Los Autómatas*, por ejemplo aves que vuelan, gorjean y beben.



Herón de Alejandría junto uno de sus inventos, un dispensador de agua sagrada operado por monedas.

Figura 1: Herón con uno de sus inventos, dispensador de agua sagrada operado por monedas

Las construcciones de la escuela de Alejandría se extendieron por todo el Imperio Romano y posteriormente por el mundo árabe, siendo estos genuinos aparatos los antecesores de los autómatas actuales.

La cultura árabe heredó y difundió los conocimientos griegos. Al-jazari, uno de los más grandes ingenieros de la historia, fue el creador de muchos inventos de control automático como el cigüeñal o uno de los primeros relojes mecánicos movidos por pesos y agua. Estuvo también muy interesado en la figura del autómata y escribió “El libro del conocimiento de los ingeniosos mecanismos”, obra considerada de las más importantes sobre la historia de la tecnología. Cabe destacar su complejo reloj elefante, animado por seres humanos y animales mecánicos que se movían y marcaban las horas o un autómata con forma humana que servía distintos tipos de bebidas.

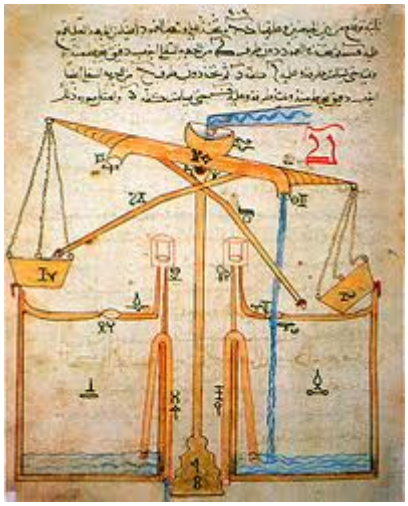


Figura 2: Libro del conocimiento de ingeniosos mecanismos de Al-jazari

Del siglo XIII son otros autómatas de los que no han llegado referencias suficientemente bien documentadas, como el *Hombre de Hierro* de Alberto Magno o la *Cabeza Parlante* de Roger Bacon. Otro ejemplo relevante de la época y que aún se conserva en la actualidad es el *Gallo de Estrasburgo*, situado en la catedral de esta misma ciudad, que mueve el pico y las alas al dar las horas. En España el Papamoscas de la catedral de Burgos, construido en el siglo XVI, consiste en un hombre mecánico que se mueve con los cambios horarios y funciona aún hoy día.

Figura 3: el papamoscas de la catedral de Burgos

Durante los siglos XV y XVI algunas de las figuras más relevantes del Renacimiento también realizaron sus propias aportaciones a la historia de los

autómatas. Leonardo da Vinci (1452–1519), algunos piensan que influenciado por Al-Jazari, diseñó los planos para un robot humanoide en torno a 1495. Los cuadernos del artista, recuperados en los años 50, contienen detallados dibujos de un caballero mecánico ahora conocido como el robot de Leonardo, que podía sentarse, alzar los brazos y mover la cabeza y la mandíbula. El diseño se basaba probablemente en la investigación anatómica registrada en su Hombre de Vitruvio. Se desconoce si intentó construir el mecanismo.

El que si llegó a construir fue el famoso León Mecánico para el rey Luis XII de Francia, que abría su pecho con la garra y mostraba el escudo de armas del rey.

Más tarde, durante la primera mitad del siglo XVI, el ingeniero Giovanni Torriani (Juanelo Turriano), quien fuera Matemático Mayor de la Corte de Felipe II, construyó un autómata llamado “El Hombre de Palo” o “Patapalo”. Tal era la complejidad del mecanismo creado por este ingeniero, que su autómata era capaz de cruzar una de las calles de Toledo (a la que posteriormente dio nombre) y pedir limosna para la construcción de un hospital.

En el siglo XVIII, y en plena fiebre por los autómatas, Wolfgang Von Kempelen llevó a cabo uno de los fraudes más conocidos en la historia de los autómatas inventando El Turco, una máquina capaz de jugar al ajedrez. Más tarde se comprobó que un maestro del ajedrez, escondido en el interior de la caja, era el encargado de manejar el maniquí que movía las fichas. La máquina viajó por toda Europa y Estados Unidos, derrotando a personajes tan ilustres como Napoleón y Benjamín Franklin.

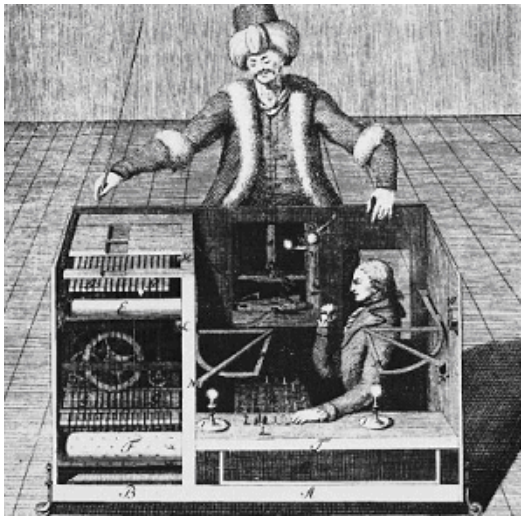


Figura 4: El turco

En 1738, Jacques de Vaucanson construyó uno de los autómatas más famosos de la historia, el *Pato con aparato digestivo*, considerado su obra maestra. El pato tenía más de 400 partes móviles, y podía batir sus alas, beber agua, digerir grano e incluso defecar. Los alimentos los digería por disolución y eran conducidos por unos tubos hacia el ano, donde había un esfínter que permitía evacuarlos. Vaucanson también construyó otros autómatas, entre los que destaca *El Flautista*, un pastor que tocaba la flauta capaz de tocar hasta 12 melodías diferentes. El ingenio consistía en un complejo mecanismo de aire que causaba el movimiento de todos las diferentes partes del engranaje interno del muñeco. El resultado era una música melodiosa, que tenía el mismo encanto y precisión en las notas que un flautista de carne y hueso.

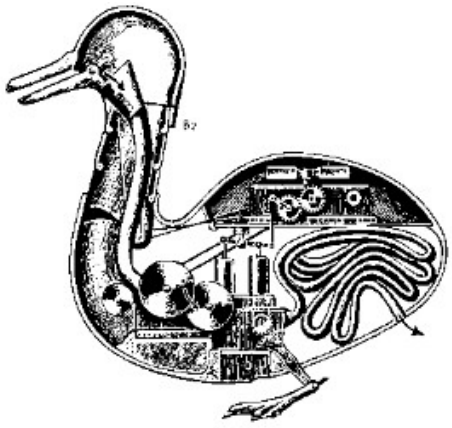


Figura 5: Pato con aparato digestivo

A finales del siglo XVIII y principios del XIX, en plena Revolución Industrial, se desarrollaron diversos ingenios mecánicos utilizados fundamentalmente en la industria textil. Entre ellos se puede citar la hiladora giratoria de Hargreaves (1770), la hiladora mecánica de Crompton (1779), el telar mecánico de Cartwright (1785), el telar de Jacquard (1801), que constituyó con sus tarjetas perforadas los primeros precedentes históricos de las máquinas de control numérico. Más tarde se incorporaron los automatismos en las industrias mineras y metalúrgicas.

Pero sin duda, el automatismo que causó mayor impacto por tener un papel fundamental en la Revolución Industrial lo realiza Potter a principios del siglo XVIII, automatizando el funcionamiento de las válvulas en la máquina de vapor atmosférica, creada por el inventor inglés Thomas Newcomen.

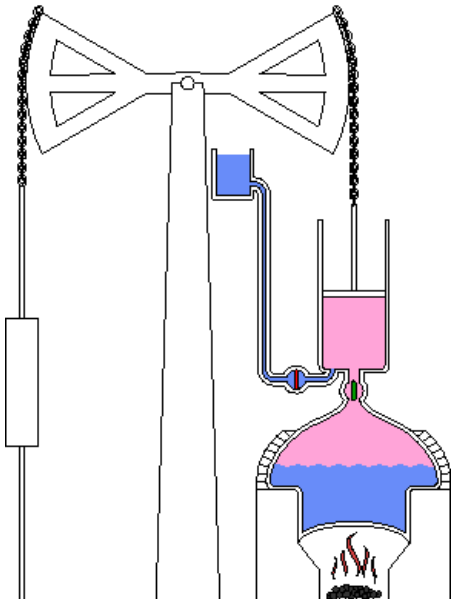


Figura 6: Máquina de vapor atmosférica.

Sin embargo, no fue hasta 1921 cuando se escuchó por primera vez la palabra robot, utilizada por el escritor checo Karel Capek en su obra de teatro *R.U.R (Rossum´s Universal Robots)*. La palabra robot viene del vocablo checo '*Robota'* que significa "trabajo", entendido como servidumbre, trabajo forzado o esclavitud.

Más tarde muchas películas y novelas mostrarían a los robots como máquinas dañinas y amenazadoras y es en el mundo de la ciencia ficción donde aparece por primera vez el término Robótica y este viene acuñado de la mano del escritor Isaac Asimov.

La imagen de robot que aparece en su obra es el de una máquina bien diseñada y con una seguridad garantizada que actúa de acuerdo con tres principios también denominados como las Tres Leyes de la Robótica a las que más tarde añadió la Ley Cero:

- *Ley Cero*: un robot no dañará a la humanidad, ni permitirá por inacción que la humanidad sea dañada.

- *Ley Uno*: un robot no dañará a un ser humano, ni permitirá por inacción que un ser humano sea dañado, a menos que esto contradiga una ley de mayor orden.

- *Ley Dos*: un robot debe obedecer una orden dada por un ser humano, excepto cuando estas órdenes contradigan una ley de mayor orden.

- *Ley Tres*: un robot debe proteger su propia existencia siempre que esta protección no contradiga una ley de mayor orden.

Consecuentemente todos los robots de Asimov son fieles sirvientes del ser humano, de ésta forma su actitud contraviene a la de Capek.

En la década de 1890 el científico Nikola Tesla, inventor, entre muchos otros dispositivos, de los motores de inducción, ya construía vehículos controlados a distancia por radio. Tesla fue un visionario que escribió sobre mecanismos inteligentes tan capaces como los humanos.

Las máquinas más próximas a lo que hoy en día se entiende como robots fueron los "teleoperadores", utilizados en la industria nuclear después de la Segunda Guerra Mundial para la manipulación de sustancias radioactivas. Básicamente se trataba de servomecanismos que, mediante sistemas

mecánicos, repetían las operaciones que simultáneamente estaba realizando un operador.

El desarrollo en la tecnología, donde se incluyen las poderosas computadoras electrónicas, los actuadores de control retroalimentados, transmisión de potencia a través de engranes, y la tecnología en sensores han contribuido a flexibilizar los mecanismos autómatas para desempeñar tareas dentro de la industria. Son varios los factores que intervienen para que se desarrollaran los primeros robots en la década de los 50's. La investigación en inteligencia artificial desarrolló maneras de emular el procesamiento de información humana con computadoras electrónicas e inventó una variedad de mecanismos para probar sus teorías.

Las primeras patentes aparecieron en 1946 con los muy primitivos robots para traslado de maquinaria de Devol. También en ese año aparecen las primeras computadoras: J. Presper Eckert y John Maulchy construyeron el ENAC en la Universidad de Pensilvania y la primera máquina digital de propósito general se desarrolla en el MIT.

El primer robot móvil de la historia, pese a sus muy limitadas capacidades, fue ELSIE (Electro-Light-Sensitive Internal-External), construido en Inglaterra en 1953. ELSIE se limitaba a seguir una fuente de luz utilizando un sistema mecánico realimentado sin incorporar inteligencia adicional.

En 1954, Devol diseña el primer robot programable y acuña el término "autómata universal", que posteriormente recorta a Unimate. Utilizaba los principios de control numérico para el control de manipulador y era un robot de transmisión hidráulica. Así, Engleberger y Devol llamarían a la primera compañía de robótica Unimation (Universal Automation). La comercialización de robots comenzaría en 1959, con el primer modelo de la Planet Corporation que estaba controlado por interruptores de fin de carrera.

Los primeros robots industriales empezaron a producirse a principios de los años 60 y estaban diseñados principalmente para realizar trabajos mecánicos difíciles y peligrosos. Las áreas donde estos robots tuvieron su aplicación fueron trabajos laboriosos y repetitivos, como la carga y descarga de hornos de

fundición. En 1961 el inventor norteamericano George Devol patentaba el primer robot programable de la historia, conocido como *Unimate*, estableciendo las bases de la robótica industrial moderna.

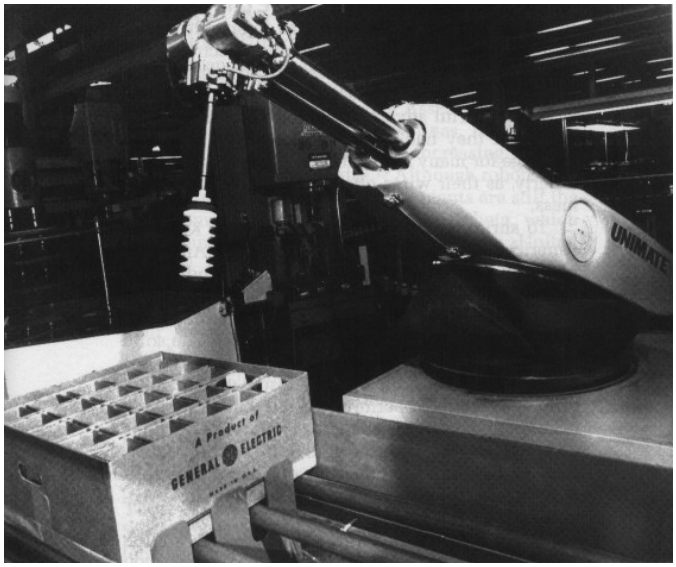


Figura 7: Unimate, robot industrial de Ford Motors Company

Este robot industrial era un manipulador que formaba parte de una célula de trabajo en la empresa de automóviles Ford Motors Company, diseñado para levantar y apilar grandes piezas de metal caliente, de hasta 225 kg, de una troqueladora de fundición por inyección.

Debido a los continuos avances en la informática y la electrónica, a partir de 1970 fueron desarrollados diversos robots programables, siendo de gran importancia en la industria mecánica, tanto en las líneas de ensamblaje como en aplicaciones como la soldadura o pintura.

Durante la década de los 70, la investigación en robótica se centra en gran parte en el uso de sensores externos para su utilización en tareas de manipulación. Es también en estos años cuando se consolida definitivamente la presencia de robots en las cadenas de montaje y plantas industriales en el ámbito mundial.

En 1972 se desarrolló en la universidad de Nottingham, Inglaterra, el SIRCH, un robot capaz de reconocer y orientar objetos en dos dimensiones. Este mismo año, la empresa japonesa Kawasaki instala su primera cadena de

montaje automatizada en Nissan, Japón, usando robots suministrados por Unimation, Inc.

En 1973, Bolles y Paul utilizan realimentación visual en el brazo Stanford para el montaje de bombas de agua de automóvil. También este mismo año, la compañía sueca ASEA (futura ABB), lanza al mercado su familia de robots IRB 6 e IRB 60, para funciones de perforación de piezas.

En esta época, la NASA inició un programa de cooperación con el anterior grupo de trabajo para desarrollar plataformas capaces de explorar terrenos hostiles. El primer fruto de esta alianza seria el MARS-ROVER, que estaba equipado con un brazo mecánico tipo STANFORD, un dispositivo telemétrico láser, cámaras estéreo y sensores de proximidad.

En 1975, Will y Grossman, en IBM, desarrollaron un manipulador controlado por computador con sensores de contacto y fuerza para montajes mecánicos. Este mismo año, el ingeniero mecánico estadounidense Victor Scheinman, cuando estudiaba la carrera en la Universidad de Stanford, California, desarrolló un manipulador polivalente realmente flexible conocido como Brazo Manipulador Universal Programable (PUMA, siglas en inglés). El PUMA era capaz de mover un objeto y colocarlo en cualquier orientación en un lugar deseado que estuviera a su alcance. El concepto básico multiarticulado del PUMA es la base de la mayoría de los robots actuales.

Figura 8: Robot PUMA Unimate

En 1979 Japón introduce el robot SCARA (Selective Compliance Assembly Robot Arm), y la compañía italiana DEA (Digital Electric Automation), desarrolla el robot PRAGMA para la General Motors.

En 1982, el robot Pedesco, se usa para limpiar un derrame de combustible en una central nuclear. También se pone un gran énfasis en los campos de visión artificial, sensorización táctil y lenguajes de programación. Gracias a los primeros pasos dados por compañías como IBM o Intelledex Corporation, que introdujo en 1984 el modelo ligero de ensamblaje 695, basado en el microprocesador Intel 8087 y con software Robot Basic, una modificación del Microsoft Basic, actualmente se tiende al uso de una interfaz (el ordenador) y diversos lenguajes de programación especialmente diseñados, que evitan el "cuello de botella" que se producía con la programación "clásica". Esta puede ser ahora on-line u off-line, con interfaces gráficas (user-friendly interfaces) que facilitan la programación, y un soporte SW+HW que tiende a ser cada vez más versátil.

Actualmente, el concepto de robótica ha evolucionado hacia los sistemas móviles autónomos, que son aquellos que son capaces de desenvolverse por sí mismos en entornos desconocidos y parcialmente cambiantes sin necesidad de supervisión.

En general la historia de la robótica la podemos clasificar en cinco generaciones (división hecha por Michael Cancel, director del Centro de Aplicaciones Robóticas de Science Application Inc. En 1984). Las dos primeras, ya alcanzadas en los ochenta, incluían la gestión de tareas repetitivas con autonomía muy limitada. La tercera generación incluiría visión artificial, en lo cual se ha avanzado mucho en los ochenta y noventas. La cuarta incluye movilidad avanzada en exteriores e interiores y la quinta entraría en el dominio de la inteligencia artificial en lo cual se está trabajando actualmente.

En los últimos años, los robots han tomado posición en todas las áreas productivas industriales. La incorporación del robot al proceso productivo ha representado uno de los avances más espectaculares de la edad moderna. En poco más de cuarenta años, se ha pasado de aquellos primeros modelos, rudos y limitados, a sofisticadas máquinas capaces de sustituir al hombre en todo tipo de tareas repetitivas o peligrosas, y además, hacerlo de forma más

rápida, precisa y económica que el ser humano. Hoy en día, se calcula que el número de robots industriales instalados en el mundo es de un millón de unidades, unos 20.000 en España, siendo Japón el país más tecnológicamente avanzado, con una media de 322 robots por cada 10.000 trabajadores.

2.3 Clasificación de los robots

No resulta sencillo hacer una clasificación de tipos de robots, puesto que ningún autor se pone de acuerdo en cuántos y cuáles son los tipos de robots y sus características esenciales.

Una clasificación podría ser según Generaciones, basada en los hitos conseguidos a través de la historia de la Robótica:

- *1ª Generación. Play-Back*: Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.
- *2ª Generación. Robots con control sensorizado*: Estos tienen un control en lazo cerrado de movimientos manipulados, y hacen decisiones basados en datos obtenidos por sensores.
- *3ª Generación. Robots con control por visión*: El controlador es una computadora que obtiene información a través de un sistema de visión y envía las órdenes al manipulador para que realice los movimientos necesarios.
- *4ª Generación. Robots inteligentes*: Son similares a los anteriores, pero además pueden automáticamente reprogramar sus acciones sobre la base de los datos obtenidos por los sensores.
- *5ª Generación. Inteligencia artificial*: Actualmente en desarrollo, la IA permitiría a los robots acercarse un paso más a la conducta humana proporcionándoles más autogobernación así como sensaciones que actualmente solo puede percibir una persona.

Otra clasificación podría ser según su arquitectura:

- *Androides:* son Robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, destinados fundamentalmente al estudio y experimentación.
  Uno de los aspectos más complejos de estos Robots y sobre el que se centra la mayoría de los trabajos, es el del equilibrio durante la locomoción bípeda.

- *Móviles*: se desplazan mediante una plataforma basada en ruedas; estos robots aseguran el transporte de piezas o su propia movilización de un punto a otro. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sensores.
  Entre otras funciones, a nivel industrial, estos Robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

- *Zoomórficos*: considerados en sentido no restrictivo podrían incluir también a los androides. Constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos.
  A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los Robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los Robots zoomórficos no caminadores está muy poco evolucionado. Cabe destacar, entre otros, los experimentados efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. En cambio, los Robots zoomórficos caminadores multípedos son muy numerosos y están siendo experimentados en diversos laboratorios con vistas al desarrollo

posterior de verdaderos vehículos terrestres, pilotados o autónomos, capaces de avanzar 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.

- *Poliarticulados*: mueven sus extremidades con pocos grados de libertad. Su utilidad es principalmente industrial, para desplazar elementos que requieren cuidados. Suelen ser sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas. En este grupo se encuentran los manipuladores, los Robots industriales y los Robots cartesianos. Se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

- *Híbridos*: Estos Robots corresponden a aquellos de difícil clasificación cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo uno de los atributos de los Robots móviles y de los Robots zoomórficos. De igual forma pueden considerarse 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.

Como clasificación final tenemos la que se basa en la función que suplen los robots, que viene a ser la razón por la que los fabricamos e invertimos en investigación en esta rama:

- *Industriales*: son aquellos empleados en procesos industriales. Suele tratarse de robots fijos con tres o más ejes, multipropósito, controlados automáticamente y reprogramables. Cabe destacar que dentro de la gama de robots industriales podríamos crear una clasificación según su función en la industria, como los robots manipuladores, entre muchos otros.

Figura 9: robot FANUC

- *Personales / Educativos:* en esta categoría entrarían todos los robots empleados con fines pedagógicos y también aquellos destinados al ocio o a realizar tareas personales.

Figura 10: robot LEGO

- *Militares:* como su nombre indica son robots fabricados con fines militares. Suelen ser los robots más avanzados tecnológicamente ya que normalmente son los que disfrutan de mayores inversiones.

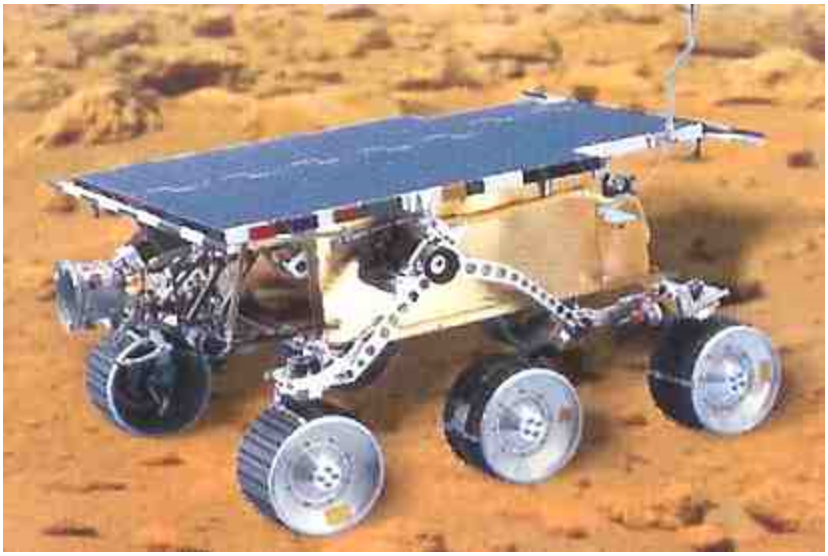


Figura 11: Mars-Rover (NASA)

2.4 Robótica móvil

Los robots móviles surgen de la necesidad de extender el campo de la Robótica. Disponen de mecanismos para la transmisión de movimiento, que permiten su desplazamiento. En el apartado anterior se ha realizado un desglose de los diferentes tipos de robots existentes atendiendo a su aplicación, pero más allá de este aspecto práctico, podemos nombrar los robots móviles como los robots modernos con mayores grados de libertad, y por tanto, mayor utilidad. El movimiento es una característica en el espacio físico, es decir, los robots tienen la posibilidad de desplazarse por el entorno observándolo o interactuando con él. Por tanto eleva muchísimo el grado de complejidad que tiene la programación de estos autómatas.

El primer robot móvil de la historia, pese a sus muy limitadas capacidades, fue ELSIE (Electro-Light-Sensitive Internal-External), construido en Inglaterra en 1953. ELSIE se limitaba a seguir una fuente de luz utilizando un sistema mecánico realimentado sin incorporar inteligencia adicional.

En la industria, se encuentra su aplicación por ejemplo para el transporte de materiales, en terrenos abruptos o peligrosos para el ser humano, así como misiones de rescate o exploración de terrenos desconocidos, incluso la imitación del comportamiento humano.

2.4.1 Clasificación

Dentro de la familia de los robots móviles, podemos hacer una clasificación según su taxonomía:

- *Robots andantes*: son los robots construidos a imagen y semejanza humana, ya que la mayoría están dotados de dos piernas. Este tipo de robots móviles tienen diferentes técnicas de control, aunque todas ellas se caracterizan por algoritmos altamente complejos, para poder mantener el equilibrio y andar correctamente tal y como lo haría un humano. Hoy en día estos robots son capaces de caminar por suelos regulares, bailar, subir escaleras, e incluso practicar deporte pero no están preparados para desplazarse por suelos irregulares. Muchas

veces la programación no está a la altura del hardware del robot y de su capacidad de procesamiento.

Figura 12: Robot humanoide Robonova

- *Robots rodantes*: son robots construidos con ruedas para poder desplazarse. Al ser los más sencillos y fáciles de construir, son con diferencia los más populares, ya que además la carga que pueden transportar es mayor, y por ello, tienen mayor utilidad.

  Incluso dentro de la clasificación de los robots rodantes, podemos considerar diferentes tipos de robots:

  - Configuración de ruedas 2+2.
  - Configuración triciclo.
  - Dotados de orugas, para terrenos con mayor irregularidad.

Figura 13: Robot dotado de oruga

- *Robots trepadores*: Son robots inspirados en animales como las serpientes o gusanos, ya que su forma de desplazarse es una imitación de la utilizada por estos. Están formados por un número elevado de secciones que pueden cambiar de tamaño o posición de forma independiente de las demás pero coordinadas, de forma que en conjunto provoquen el desplazamiento del robot.

Figura 14: Robot trepador

- *Robots nadadores*: Son robots enfocados a tareas de exploración submarina por zonas donde es difícil o peligroso llegar por su complicado acceso, o sus grandes profundidades. Estos robots se han creado basándose en el estudio del movimiento de los peces en el agua, llegando a demostrar que la estructura corporal, así como el movimiento que realizan, es uno de los movimientos más óptimos de movimiento submarino, dado que aprovecha la energía de forma muy eficiente y permite mayor control en la navegación produciendo mucho menos ruido y turbulencias.

Figura 15: Atún robótico

- *Robots voladores*: Suelen ser helicópteros RC dotados de visión artificial y capacitados para la toma de decisiones automáticas. Mayormente son utilizados con fines militares, por ejemplo para tareas de espionaje.

Figura 16: Drone, robot volador militar no tripulado

2.4.2 Componentes de un robot móvil

Llegados a este punto, donde ya tenemos vistos los principales tipos de robots que se construyen en la actualidad, a continuación se detallaran las principales partes y componentes de los robots móviles, tanto mecánicas como electrónicas.

2.4.2.1 Sensorización

Los humanos no damos, a menudo, importancia al funcionamiento de nuestros sistemas sensoriales. Vemos una taza sobre una mesa, la cogemos automáticamente y no pensamos en ello, al menos no somos conscientes de pensar mucho en ello. De hecho, el conseguir beber de una taza requiere una compleja interacción de sentidos, interpretación, conocimiento y coordinación, que, en la actualidad, entendemos mínimamente.

Por tanto, infundir a un robot prestaciones de tipo humano resulta ser tremendamente difícil. Los juegos de ordenador que derrotan a los campeones de ajedrez son comunes en nuestros días, mientras que un programa que reconozca una silla, por ejemplo, en una escena arbitraria aún no existe. El "ordenador paralelo" que todos tenemos en nuestra cabeza dedica grandes cantidades de materia gris a los problemas de la percepción y la manipulación.

Los sensores son los elementos encargados de recoger información del entorno del robot y transmitirla a la unidad de control para su procesamiento. Los sensores constituyen el sistema de percepción del robot, es decir, facilitan la información del mundo real para que el autómata la interprete. Estos son los tipos de sensores más utilizados hoy en día:

- *Sensor de proximidad*: Detectan la presencia de un objeto ya sea por rayos infrarrojos, por sonar, magnéticamente o de otro modo.

Figura 17: Sensores de proximidad

- *Sensor de Temperatura*: Capta la temperatura del ambiente, de un objeto o de un punto determinado.

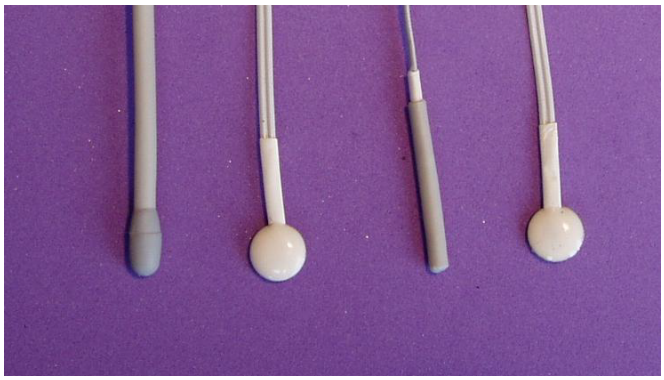


Figura 18: Sensores de temperatura

- *Sensor magnéticos:* Captan variaciones producidas en campos magnéticos externos. Se utilizan a modo de brújulas para orientación geográfica de los robots.

Figura 19: Sensores magnéticos

- *Sensores táctiles*: Sirven para detectar la forma y el tamaño de los objetos que el robot manipula. La piel robótica se trata de un conjunto de sensores de presión montados sobre una superficie flexible. Estos sensores se utilizan en robótica para, por ejemplo, obtener información asociada con el contacto entre una mano manipuladora y objetos en el espacio de trabajo o simplemente evitar colisiones con paredes.

  La información obtenida puede utilizarse para la localización y el reconocimiento de un obstáculo, así como para controlar la fuerza ejercida por un manipulador sobre un objeto dado. Los sensores de contacto pueden subdividirse en dos categorías principales:

  - Binarios
  - Analógicos

  Los sensores binarios son esencialmente conmutadores que responden a la presencia o ausencia de un objeto. Por el contrario los sensores analógicos proporcionan a la salida una señal proporcional a una fuerza local.

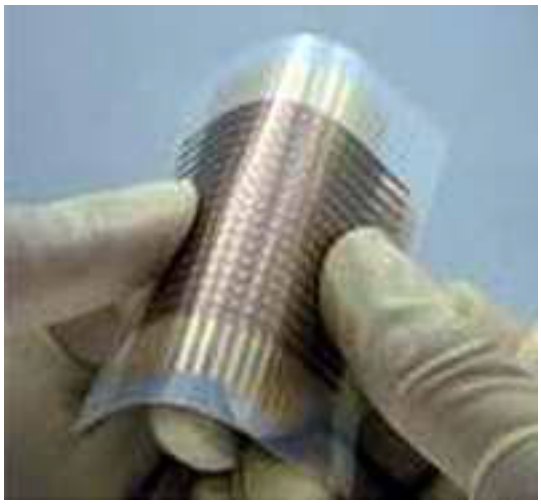


Figura 20: Sensor táctil flexible

- *Sensores de iluminación:* Capta la intensidad luminosa, el color de los objetos, etc. Es muy útil para la identificación de objetos. Es parte de la visión artificial y en numerosas ocasiones son cámaras.

Figura 21: Sensores de luz

- *Sensores de velocidad:* Se emplean para determinar la velocidad de actuación de las distintas partes móviles del propio robot o cuando se produce una vibración. También se detecta la inclinación a la que se encuentra el robot o una parte de él.

Figura 22: Sensores de velocidad de reluctancia variable

- *Sensores de fuerza:* Permiten controlar la presión que ejerce la mano del robot al coger un objeto.

Figura 23: Sensor de presión

- *Sensores de sonido:* Micrófonos que permiten captar sonidos del entorno.

Figura 24: Micrófono

- *Micro interruptores:* Muy utilizados para detectar finales de carrera.

Figura 25: Micro interruptores

2.4.2.2 Actuadores

Los actuadores son los sistemas de accionamiento que permiten el movimiento de las articulaciones del robot. Se clasifican en tres grupos dependiendo del tipo de energía que utilicen:

- *Hidráulicos*: se utilizan para manejar cargas pesadas a una gran velocidad. Sus movimientos pueden ser suaves y rápidos.
- *Neumáticos*: son rápidos en sus respuestas, pero no soportan cargas tan pesadas como los hidráulicos.
- *Eléctricos*: son los más comunes en los robots móviles. Un ejemplo son los motores eléctricos que permiten conseguir velocidades y precisión necesarias.

2.4.2.3 Alimentación

Parte fundamental para garantizar la autonomía del robot, dado que al ser móvil generalmente no va a tener energía externa, solamente contará con

las reservas internas que pueda transportar, salvo en el caso que se utilicen placas solares.

Son muchas y muy diversas las formas en que un autómata móvil pueda transportar y almacenar energía, por lo que vamos a ver la principales y más utilizadas:

- *Baterías*: sin duda la forma más comúnmente utilizada como fuente de alimentación en todo tipo de robots. Son económicamente asequibles, fiables se pueden encontrar en una gran variedad de formatos, voltajes y dimensiones. Desde la pila de botón más pequeña y ligera para alimentar un micro robot hasta las pesadas y potentes batería de plomo utilizadas en los vehículos pesados necesitados de un aporte importante de energía.

  Otro aspecto importante a tener en cuenta es el formato y el conexionado de las baterías. Habitualmente encontramos las batería recargables tanto sueltas como en packs preparados para soldar o para conectar directamente a un PCB. La elección de un formato u otro es más relevante de lo que parece. Pensemos en el ejemplo de un robot móvil todoterreno, si llevase pilar individuales colocadas en un porta-pilas, podría fácilmente que alguna de ellas dejase de hacer contacto o incluso se saliera de su sitio. Pero también tendríamos problemas con las pilas soldadas, por ejemplo a la hora de reemplazarlas o de cargarlas, ya que tendríamos que hacerlo con todas al mismo tiempo. Parece por tanto, que lo más adecuado sería utilizar pilas en packs con conectores, ya que son las que mayor flexibilidad nos aportarían para este tipo de robots.

  Si el presupuesto no es un problema se pueden usar tecnologías más avanzadas como las baterías de ion de litio que últimamente están sufriendo una gran expansión gracias a tecnologías móviles de otras índoles como ordenadores portátiles y teléfonos móviles.

  La diferencia de rendimiento de unos tipos de pilas respecto a otros es notable, así como su tamaño y precio, por lo tanto es importante tener claros los requerimientos del robot respecto a estos

aspectos antes de decidir qué tipo de batería se va a utilizar para alimentarlo.

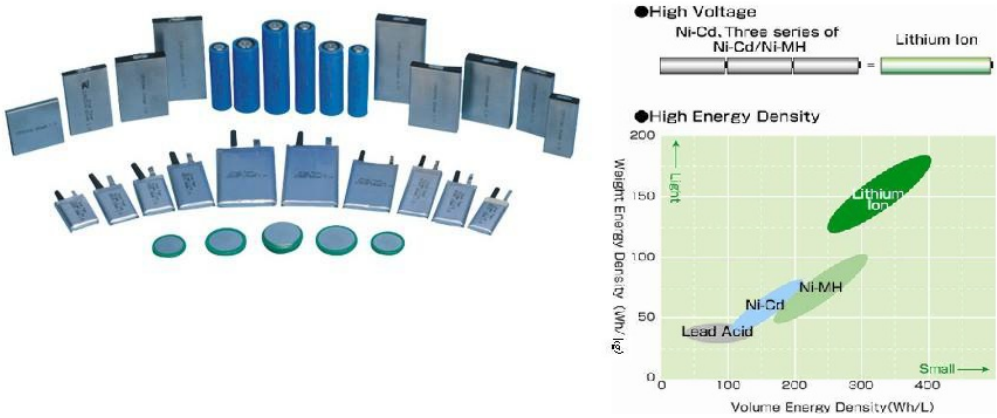


Figura 26: Diferentes tipos de baterías de Ion Litio (izquierda), y gráfico comparativo con diferentes tipos de baterías (derecha)

- *Bombonas de aire*: usadas tanto para el movimiento de actuadores neumáticos como para la propulsión de motores de aire. Este tipo de motores tiene una larga historia, pese a que su difusión es reducida. Su principio es equivalente al de los motores de explosión, con la salvedad que en lugar de producir la combustión de gasolina y oxígeno para generar los gases que impulsan los pistones, se usa directamente aire almacenado en un depósito a alta presión.

Figura 27: Bombonas de aire

- *Baterías inerciales*: este curioso mecanismo tampoco tiene una difusión amplia en el terreno de los robots móviles, pero si tiene aplicación e entornos donde la alimentación es crítica, como centros de proceso de datos. Fue probada en vehículos durante sus albores,

concretamente en autobuses urbanos de Zúrich durante los años 50 y algunas locomotoras de tren durante los 70, pero no se ha extendido su uso de forma masiva.

El principio de funcionamiento es sencillo: se acelera un rotor (habitualmente mediante electricidad) confiriéndole así una elevada energía rotacional, típicamente de varias decenas de miles de rpm. Posteriormente se decelera poco a poco, recuperando así la energía eléctrica de nuevo.

La clave del aprovechamiento energético, que oscila entre el 90% y el 98%, consiste en reducir al máximo el rozamiento. Esto se consigue mediante usando un rotor de carbono (que presenta menos resistencia que el acero) estabilizado magnéticamente en un entorno controlado, generalmente un contenedor sellado tras realizarle el vacío.

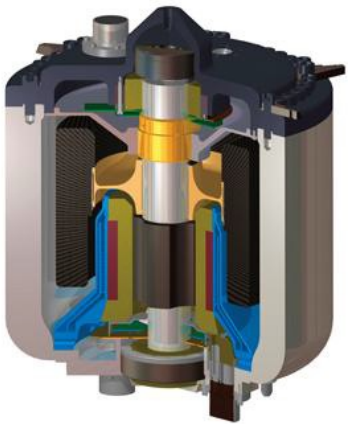


Figura 28: Sección de batería inercial

- *Células de hidrogeno*: Otra tecnología emergente que poco a poco está demostrando su validez son las baterías formadas a base de células que usan el hidrógeno como combustible para producir electricidad.

  Su funcionamiento se basa en una reacción química al igual que en las baterías tradicionales pero a diferencia de ellas, aquí no se producen cambios de estado en los electrodos o en los reactantes. Simplemente se consumen los reactantes y la reacción se produce prácticamente de forma indefinida mientras haya suficiente combustible para sustentar el proceso.

Las principales ventajas son su elevado rendimiento así como su larga vida, que va asociada a la práctica ausencia de mantenimiento que necesitan. En contraposición el combustible a día de hoy todavía es relativamente caro y difícil de conseguir aunque esta situación mejorará cuando haya una mayor demanda de estas células.

Figura 29: Célula de hidrógeno

- *Motor de combustión*: El más conocido entre los sistemas de propulsión. Se usan motores muy similares a los de los vehículos, sobre todo los que llevan las motocicletas. Como combustible una mezcla de aceite y gasolina u otros elementos volátiles como el etanol. Se pueden utilizar tanto para obtener energía mecánica como energía eléctrica si añadimos un generador.

Figura 30: Motor de combustión

- *Biomasa:* Una fuente de energía muy interesante y que aun se encuentra por explotar. Pequeñas cantidades de basura orgánica

permitirían alimentar durante horas un dispositivo dotado de esta tecnología. A parte del ahorro económico que supone, tiene especial interés en zonas sin acceso directo a fuentes de energía o en momentos de conflicto militar donde frecuentemente escasean los suministros energéticos.

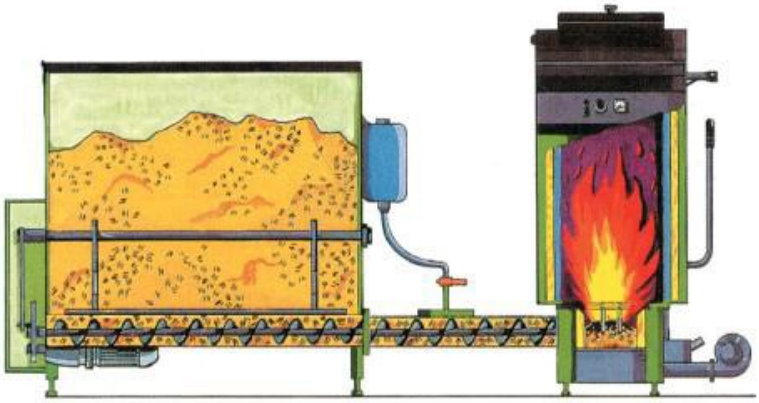


Figura 31: Caldera de biomasa

2.4.2.4 Sistemas de control

El control de un robot puede realizarse de muchas maneras, pero generalmente se realiza por medio de un ordenador industrial altamente potente, también conocido como unidad de control o controlador. El controlador se encarga de almacenar y procesar la información de los diferentes componentes del robot industrial.

La definición de un sistema de control es la combinación de componentes que actúan juntos para realizar el control de un proceso. Este control se puede hacer de forma continua, es decir en todo momento o de forma discreta, es decir cada cierto tiempo. Si el sistema es continuo, el control se realiza con elementos continuos. En cambio, cuando el sistema es discreto el control se realiza con elementos digitales, como el ordenador, por lo que hay que digitalizar los valores antes de su procesamiento y volver a convertirlos tras el procesamiento.

Existen dos tipos de sistemas, sistemas en lazo abierto y sistemas en lazo cerrado:

- *Sistemas en bucle abierto*: son aquellos en los que la salida no tiene influencia sobre la señal de entrada.

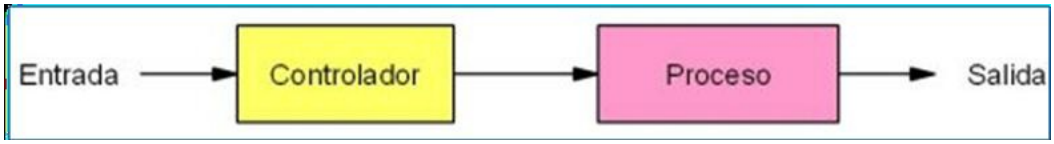


Figura 32: Esquema de un controlador de bucle abierto

- *Sistemas en bucle cerrado:* son aquellos en los que la salida influye sobre la señal de entrada.

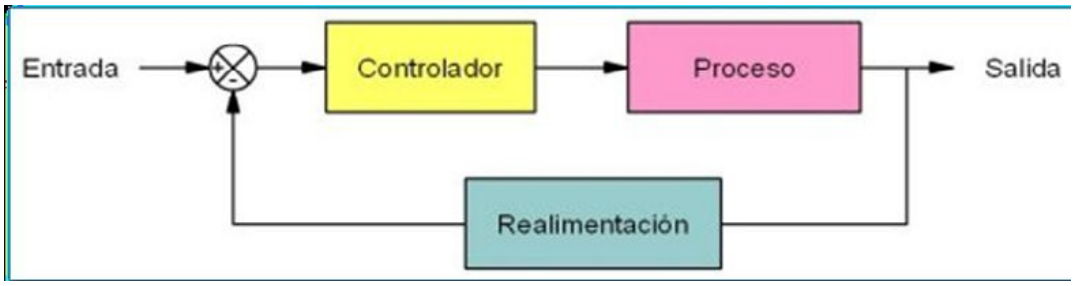


Figura 33: Esquema de un controlador de bucle cerrado

- *Sistemas discretos:* son aquellos que realizan el control cada cierto tiempo. En la actualidad se utilizan sistemas digitales para el control, siendo el ordenador el más utilizado, por su fácil programación y versatilidad. Generalmente, el control en los robots se realiza mediante sistemas discretos en lazo cerrado, realizados por computador. El ordenador procesa la información captada por los sensores y activa los actuadores en intervalos lo más cortos posibles, del orden de milisegundos.

2.5 Lego NXT

LEGO ha sido desde siempre un juguete de construcción sencilla y versátil que casi permitía atravesar los límites de la creatividad. Esta andadura empezó en el año 1934 en Dinamarca y con el tiempo fue cobrando popularidad entre padres, hijos y sobretodo educadores que veían en estos juguetes una forma muy interesante de divertir y a la vez potenciar la creatividad de sus hijos y alumnos.

La oferta de juguetes va desde los pocos meses de edad hasta la adolescencia e incluso un poco más allá, contando con temáticas urbanas, submarinas e incluso espaciales. Entre todas estas líneas cabe destacar una que desde el principio ha contado ha despertado la curiosidad de determinado sector de público, en el que podían englobarse ingenieros y apasionados de la técnica. Hablamos de la denominada línea *Technic*. La principal característica de esta gama de juguetes y la que sin duda la ha hecho tan famosa, es el peculiar enfoque de construcción que ha concebido Lego con ella.

Este enfoque se aparta del juguete tradicional y experimenta con elementos propios de la tecnología actual, tanto en mecánica como en electrónica. Así, entre las piezas de estos 'kits' se encuentran engranajes, ejes, motores eléctricos, cables y hasta elementos tan concretos como levas y pistones, juntas cardan y amortiguadores.

Figura 34: Motor de Lego

En 1998 LEGO anunció el primer conjunto de la gama Mindstorms, el Robotics Invention System (RIS 1.0), añadiendo una nueva dimensión al universo de LEGO. En él, además de las piezas de LEGO Technic, el kit proporcionaba motores de corriente continua, sensores y, lo más importante, el RCX.

El RCX era el "brick" (ladrillo) programable de LEGO que permitía no sólo controlar los actuadores para dotar al robot de movimiento, sino que también manejaba la sensorización y la consecuente respuesta al entorno. Estaba basado en el microprocesador H8 de Hitachi y proporcionaba convertidores analógico/digital, comunicación serie y temporizadores.

El RCX se desarrolló en colaboración entre LEGO y el MIT. En la primera versión el RCX tenía 6 puertos de entrada y 6 de salida, aunque posteriormente se limitaron ambas entrada y salida a los 3 actuales por razones de consumo.

Debido a su relativamente barato coste y a las grandes posibilidades que ofrecía cuando se combinaban de forma ingeniosa sus componentes no tardó en hacerse un sitio en el mercado. Al mismo tiempo se ganó también la simpatía de numerosos aficionados de la robótica que intercambiaban por Internet sus montajes y aplicaciones contribuyendo con diferentes empresas a ampliar tanto el hardware como el software desarrollando (entre otras cosas) entornos de programación para el sistema o fabricando nuevos sensores para poder sacarle un mayor partido a su robot.

En Enero del 2006 se presentó en el International Consumer Electronics Show la siguiente generación de esta exitosa gama: el LEGO Mindstorms NXT. La nueva versión mejora notablemente los motores y trae otros pequeños cambios en los sensores electrónicos y en las piezas de construcción pero, lo más importante otra vez, es que incorpora una unidad de control nueva: el NXT.

Al igual que pasaba con la versión anterior, combinando los bloques de construcción, la fácil programación del NXT y su interfaz de entrada y salida se puede obtener un sistema de prototipado rápido para el desarrollo de una gran variedad de actividades, lo que ha permitido que este sistema haya sido ampliamente aceptado como una herramienta para la investigación y la educación universitaria.

Figura 35: Componentes básicos del NXT

Los modelos de LEGO pueden ser ahora algo más que un simple conjunto de piezas de montaje. Gracias a la funcionalidad que proporcionan los LEGO Mindstorms, los resultados del montaje de estas piezas aisladas pueden sentir y responder a su entorno, así como ser programados para llevar a cabo casi cualquier función.

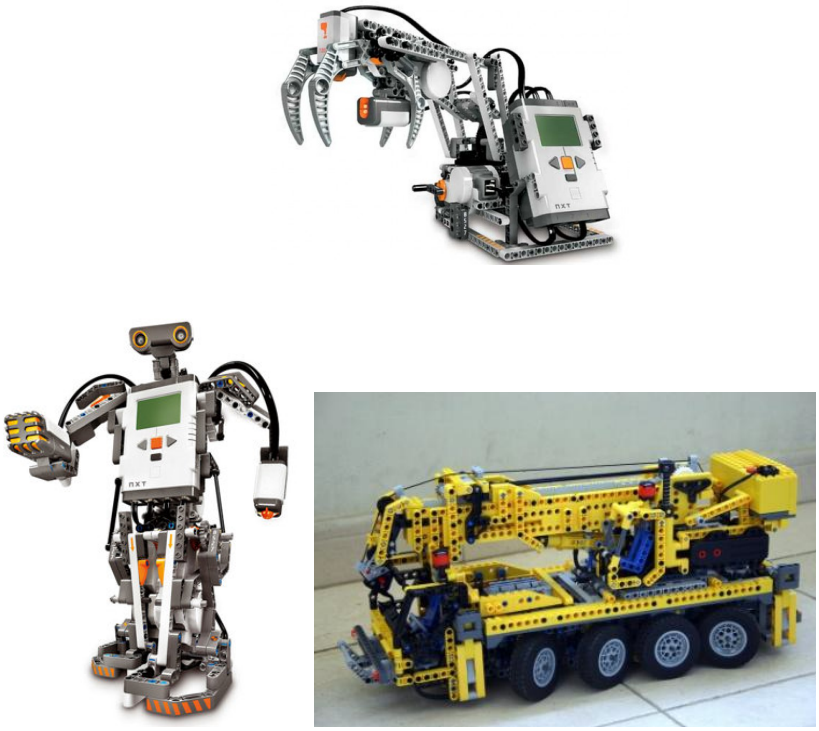


Figura 36: Distintos tipos de montajes con NXT

Como hemos estado comentando, los componentes más importantes del LEGO Mindstorms NXT, los cuales vamos a ver a continuación, son la unidad

de control basada en un micro controlador (brick), los sensores electrónicos y los actuadores.

Por otro lado tenemos las piezas de montaje que aunque no tan sofisticadas, dotan al robot de una estructura y de su buen montaje dependerán muchas características del mismo.

2.5.1 Ladrillo NXT

Posiblemente se trate de la pieza más importante del robot. Prácticamente todas las universidades a nivel mundial poseen ya kits de LEGO con estos bricks inteligentes y se usan en las ramas de la enseñanza y la investigación como las relacionadas con la robótica o la mecatrónica.

El brick vendría a ser el cerebro del NXT, provee de una unidad de control para robots o sistemas electrónicos avanzados que permite implementar escenarios hasta el momento impensables. Su elemento principal está formado por un micro controlador o procesador, que en sus últimas versiones ha alcanzado grados de potencia sorprendentes si pensamos en el Lego como un juguete.

Figura 37: Versión NXT del ladrillo de control

Se alimenta con 6 pilas 1,5 V AA o con una batería recargable de Litio cuya autonomía dependerá del número de actuadores y de sensores que estén trabajando en ese momento.

Tanto para la lectura de los 4 puertos de entrada como para el acceso a los 3 puertos de salida se utilizan cables RJ12, muy similares pero incompatibles a los RJ11 utilizado en la telefonía.

En NXT dispone también de una pantalla LCD de escala de grises de 100x64 pixeles y 26x40.6 mm. Cuenta además con 4 botones que permite a los usuarios navegar por una interfaz jerárquica de menús. Además, cuenta con un altavoz y un canal de salida de sonido que puede reproducir ficheros de sonido con periodos de muestreo de más de 16 kHz y 8 bits de resolución. Para las comunicaciones, el NXT tiene un puerto USB 2.0 full speed de 12 Mbits/s y un puerto inalámbrico Bluetooth CSR (BlueCoreTM 4 v2.0 clase II serial port profile 47Kbyte RAM, 8Mbit FLASH, 26MHz).

El brick del NXT puede conectarse inalámbricamente a otros tres dispositivos a la vez, pero cada intervalo de tiempo sólo lo puede hacer con uno exclusivamente.

Como se ha citado anteriormente, el NXT está equipado con un microprocesador AT91SAM7S256. Además dispone de un coprocesador matemático basado en el micro controlador ATmega48.

El AT91SAM7S256 es un microprocesador de 32 bits, 256KB de memoria flash y 64KB de memoria RAM. Implementa una arquitectura ARM v4T, que le permite superar los 130 MIPS, pudiéndose encontrar en dispositivos como el IPod de Apple, la Nintendo DS, la mayoría de los móviles de Nokia, etc.

A continuación una lista resumida de las características electrónicas del *brick* NXT, extraídas del manual de LEGO:

*** Procesador principal: Atmel 32-bit ARM processor, AT91SAM7S256:**

- 256 KB FLASH
- 64 KB RAM
- 48 MHz

* **Co-procesador: Atmel® 8-bit AVR processor, ATmega48:**

- 4 KB FLASH
- 512 Byte RAM

- 8 MHz

*** Unidad de comunicaciones Bluetooth CSR BlueCoreTM 4 v2.0 + EDR System:**

- Soporte de Serial Port Profile (SPP)
- 47 KByte RAM Internos
- 8 MBit FLASH Externos
- 26 MHz

*** Comunicación vía USB 2.0 (12 Mbit/s)**

*** 4 puertos de entrada con interfaz de 6 hilos y soporte para conexiones AD y DA**

- 1 puerto de alta velocidad, IEC 61158 Tipo 4/EN 50170 compatible
- todos los puertos de entrada cuentan con soporte del bus **I²C**

*** 3 puertos de salida con interfaz de 6 hilos y soporte para lectura desde encoders**

*** Conectores de 6 hilos estándar industrial, RJ12 con ajuste a derechas**

*** Display grafico LCD de 100 x 64 píxel (blanco y negro)**

- Área de visión de 26 x 40.6 mm

***Altavoz de salida con resolución de 8-bit**

- Soporta tasas de muestreo de 2 a 16 KHz

*** 4 botones de goma para interacción con el usuario**

***Alimentación: 6 pilas de tipo AA**

- Se recomienda usar pilas alcalinas
- También se encuentra disponible una batería de Ion de Litio de 1400 mAH
- Cargador opcional disponible para la batería anterior

2.5.2 Actuadores

La gama de motores de que consta la línea LEGO Technic es bastante elevada, contando entre sus filas con aproximadamente 13 modelos diferentes. Estos motores van desde los pequeños micro motores hasta la joya de la corona de esta familia que no es otro que el motor incluido en el set del NXT.

El motor incluido en el set NXT posee unas características muy buenas en cuanto a respuesta y rendimiento. Cuenta con un juego de engranajes

reductores y un encoder rotacional en su interior. La velocidad máxima de los motores es de 200RPM.

Figura 38: Actuador del LEGO Mindstorms: motor de corriente continua

El juego de engranajes reductores garantiza un par de fuerza elevado para el motor, lo que se traduce en la capacidad de realizar un esfuerzo mayor sin la necesidad de elementos externos como era necesario en modelos anteriores, y más importante aún, impidiendo el sobreesfuerzo del motor y la consecuente reducción de su vida útil.

Por ejemplo, alimentado a 9V y funcionando a 117 rpm proporciona un par de 16.7N/cm con un consumo de corriente de 0.55A.

El encoder o tacómetro, por su parte, permite conocer el número de vueltas que ha dado el eje del motor con una precisión de 1 grado. Es decir, si el motor realiza un giro completo el encoder verá incrementada su cuenta en 360 unidades. Esto es cierto sea cual sea el sentido de movimiento del motor, incrementándose en un sentido y decrementandose al girar en sentido contrario.

El NXT es principalmente un dispositivo digital, y como tal únicamente puede proporcionar dos valores: nivel alto ó 1 lógico (9 voltios) o nivel bajo ó 0 lógico (0 voltios). Por ello el NXT necesita alguna forma de crear estos 100 niveles de potencia intermedios para los motores desde la señal digital. La forma de obtener una potencia variable se realiza mediante el mecanismo de Modulación de Anchura de Pulso (PWM).

Este método consiste en arrancar por un periodo de tiempo y luego pararlo. El tiempo durante el cual está en funcionamiento el motor se denomina pulso, repitiendo esto periódicamente conseguiremos que el motor funcione con una potencia menor a la máxima. Si variamos la duración de los pulsos podemos obtener diferentes velocidades.

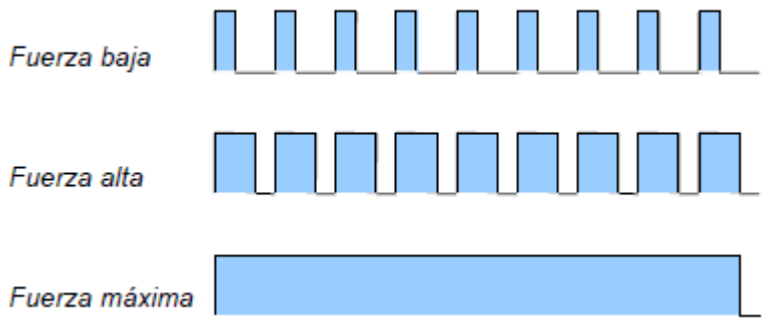


Figura 39: Señal PWM para el control del motor de corriente continua

Uno de los problemas que tiene el PWM es que en lugar de estar suministrando continuamente potencia parcial, esta suministrándola de forma completa en parte del tiempo. En muchos casos, esta sutil diferencia tiene grandes efectos en los dispositivos, sin embargo los motores de LEGO son la excepción ya que están diseñados para ser muy eficientes en potencia y tienen un volante de inercia interno, el cual actúa como un pequeño tanque de almacenaje de energía. El volante de inercia es más efectivo cuando el motor tiene una resistencia muy pequeña, es decir cuando no esta sometido a una carga grande. Cuando arranca el motor consume mucha energía hasta alcanzar la velocidad necesaria. Una vez en marcha, sin embargo, un motor sin una carga excesiva (por ejemplo un motor que únicamente tiene conectado a su salida un engranaje y nada más) puede mantener su velocidad con muy poca energía externa. En este caso, PWM no afecta al motor demasiado ya que los pulsos cortos mantienen al volante girando, y el volante mismo mantiene al motor girando cuando cesan los pulsos. Bajo una carga fuerte (como por ejemplo un robot atravesando una superficie con fuerte rozamiento, como una alfombra), el volante es menos efectivo y el cambio de los niveles de potencia puede tener unos efectos más notables sobre la velocidad del motor.

Para proteger el motor de intensidades muy altas, están equipados por una resistencia PTC montada en serie con el motor, de manera que su valor incrementa rápidamente cuando la temperatura aumenta, limitando de esta forma la intensidad suministrada al motor.

Por último también es necesario tener en cuenta que la velocidad de régimen permanente varía de un motor a otro, con una diferencia que puede incluso alcanzar el 11%. Además se ha comprobado que la velocidad varía en un mismo motor alrededor de un 2% entre el sentido de avance y retroceso. Este fenómeno se debe tener en cuenta a la hora de la programación de los algoritmos de control puesto que puede afectar de una forma considerable en la respuesta del sistema (robot móvil).

2.5.3 Sensores

A continuación vamos a realizar un repaso por el conjunto de sensores disponibles para el LEGO Mindstorms NXT, tanto los de marca propia de LEGO como aquellos ensamblados y distribuidos por terceros pero que son plenamente compatibles con la plataforma NXT.

Son cuatro los sensores incluidos en el kit original de LEGO Mindstorms: de contacto, de luz, de sonido y de distancia.

- *Bumper:* detecta la presión realizada sobre la punta del sensor. Útil para detectar colisiones, presencia de objetos o como mecanismo final de carrera.
- *Luz/Color*: este sensor cumple una doble función, es capaz de detectar la presencia de luz en un entorno y a su vez es capaz de identificar un pequeño rango de colores de los objetos que 've'.
- *Sonido:* consta de un pequeño micrófono capaz de captar sonidos. Los nuevos sensores de sonido permiten medir niveles de sonido tanto en decibelios (dB) como en decibelios ajustados (dBA), de manera que el sensor se adapta a la sensibilidad del oído humano (frecuencias entre 3 y 6 kHz).

- *Ultrasonidos:* permite detecta la presencia de sonidos a una cierta distancia. Útil en las mismas situaciones que el bumper pero sin llegar al contacto. Se trata de sensores basados en ultrasonidos, que permiten medir distancias entre 0 y 255 cm, con una precisión de +/- 3 cm.

Figura 40: Sensores del LEGO Mindstorms NXT

Fabricantes como Hi-Technic comercializan sensores compatibles con el NXT, de entre los cuales destacamos los siguientes:

- *Brújula:* permite conocer la orientación del robot respecto al norte magnético.
- *Acelerómetro:* detecta aceleración en tres ejes y giro en uno.
- *Seguidor de infrarrojos:* es capaz de detectar el haz de un emisor en un ángulo de 135º, indicando así mismo la dirección desde la que proviene el haz de luz.

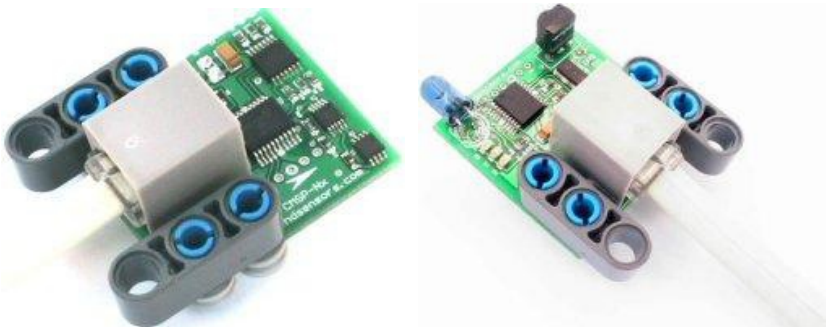


Figura 41: Seguidor de infrarrojos (izquierda), brújula (derecha)

Es posible crear sensores propios o accesorios personalizados para el NXT compatibles con el bus I2C como MindSensors que dispone incluso de micro cámaras de visión que se conectan directamente a los puertos de entrada del ladrillo.

2.5.4 Piezas de montaje

Para montar y realizar diseños propios se incluyen en el set un conjunto de 672 piezas entre las que se incluyen ruedas, engranajes, soportes, etc. que son compatibles con otras piezas de Lego que podamos tener de otros productos. También podemos usar o fabricar nuestras propias estructuras sin necesidad de que se trate de piezas de Lego para montar el robot que deseemos, aunque los sensores, los actuadores y el brick están diseñados para ser montados con piezas de Lego.

2.6 Conexiones

Dado que durante el desarrollo de este proyecto se hemos utilizado dos tipos de conexiones, en este apartado se presenta una breve descripción de los dos campos que se hemos utilizado:

- USB para conectar el robot con el PC.
- Bluetooth para sincronizar las tareas de los robots.

2.6.1 USB

El Universal Serial Bus (bus universal en serie) o Conductor Universal en Serie (CUS), abreviado comúnmente USB, es un puerto de comunicación de periféricos a un computador. Fue creado en 1996 con la finalidad de eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos de bus ISA o PCI y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados sin necesidad de reiniciar el sistema. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos los buses PCI o PCIe salen ganando e igual sucede si la aplicación requiere robustez industrial.

Se pueden clasificar en cuatro tipos según su velocidad de transferencia de datos:

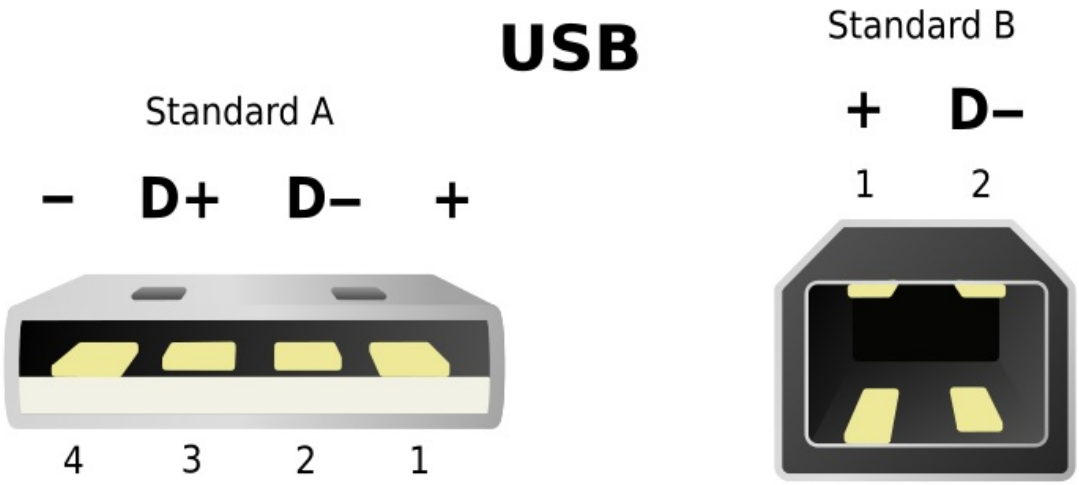
- *Baja velocidad (1.0)*: Tasa de transferencia de hasta 1,5 Mbps (192KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana como los teclados, ratones e incluso artículos del hogar.

- *Velocidad completa (1.1):* Tasa de transferencia de hasta 12Mbps (1,5MB/s), según el estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones.

- *Alta velocidad (2.0):* Tasa de transferencia de hasta 480 Mbps (60MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los ordenadores actuales. El cable USB 2.0 dispone de cuatro líneas, una para datos, una para corriente y otra de toma de tierra.

- *Súper alta velocidad (3.0):* Actualmente se encuentra en fase experimental y tiene una tasa de transferencia de hasta 4.8 Gbps (600MB/s). Esta especificación será diez veces más veloz que la anterior 2.0 y prevé su lanzamiento a corto plazo. Se han incluido cinco conectores extra, desechando el conector de fibra óptica propuesto inicialmente y será compatible con los estándares anteriores.

Las señales del USB se transmiten en un cable de par trenzado con impedancia característica de 90 Ω ± 15%, cuyos hilos se denominan D+ y D-.[4] Estos, colectivamente, utilizan señalización diferencial en half dúplex excepto el USB 3.0 que utiliza un segundo par de hilos para realizar una comunicación en full dúplex. La razón por la cual se realiza la comunicación en modo diferencial es simple, reduce el efecto del ruido electromagnético en enlaces largos. D+ y D- suelen operar en conjunto y no son conexiones simples. Los niveles de transmisión de la señal varían de 0 a 0'3 V para bajos (ceros) y de 2'8 a 3'6 V para altos (unos) en las versiones 1.0 y 1.1, y en ±400 mV en alta velocidad (2.0). En las primeras versiones, los alambres de los cables no están conectados a masa, pero en el modo de alta velocidad se tiene una terminación de 45 Ω a masa o un diferencial de 90 Ω para acoplar la impedancia del cable. Este puerto sólo admite la conexión de dispositivos de bajo consumo, es decir, que tengan un consumo máximo de 100 mA por cada puerto; sin embargo, en caso de que estuviese conectado un dispositivo que permite 4 puertos por cada salida USB (extensiones de máximo 4 puertos), entonces la energía del USB se asignará en unidades de 100 mA hasta un máximo de 500 mA por puerto.



Pin	Nombre	Color del cable	Descripción
1	VCC	Rojo	+5v
2	D–	Blanco	Data –
3	D+	Verde	Data +
4	GND	Negro	Masa

Figura 42: Salidas USB

2.6.2 Bluetooth

Bluetooth proviene de la palabra escandinava "*Blåtand"* que significa "hombre de tez oscura" pero en los tiempos que corren el significado original se ha perdido y ahora se asocia a las comunicaciones inalámbricas, un estándar global que posibilita la transmisión de voz, imágenes y en general datos entre diferentes dispositivos en un radio de corto alcance y lo que le hace más atractivo, muy bajo coste.

Los principales objetivos que este estándar quiere lograr son:

- Facilitar las comunicaciones entre equipos.
- Eliminar cables y conectores entre aquéllos.
- Facilitar el intercambio de datos entre los equipos.

Bluetooth funciona bajo radio frecuencias pudiendo atravesar diferentes obstáculos para llegar a los dispositivos que tenga a su alcance.

Opera bajo la franja de frecuencias 2.4 – 2.48 GHz o como también es conocida como "Banda ISM" que significa "Industrial, Scientific and Medical"

que es una banda libre para usada para investigar por los tres organismos anteriores. Pero esto tiene sus consecuencias, ya que al ser libre puede ser utilizada por cualquiera y para ello, para evitar las múltiples interferencias que se pudieran introducir (microondas, WLANs, mandos, etc.) Bluetooth utiliza una técnica denominada salto de frecuencias.

El funcionamiento es ir cambiando de frecuencia y mantenerse en cada una un “slot” de tiempo para después volver a saltar a otra diferente. Es conocido que entre salto y salto el tiempo que transcurre es muy pequeño, concretamente unos 625 microsegundos con lo que al cabo de un segundo se puede haber cambiado 1600 veces de frecuencia.

Cuando coinciden más de un dispositivo bluetooth en un mismo canal de transmisión se forma lo que se llaman “piconets” que son redes donde hay un maestro que es el que gestiona la comunicación de la red y establece su reloj y unos esclavos que escuchan al maestro y sincronizan su reloj con el del maestro.

Dicho alcance puede variar según la potencia a la que se transmite (a mayor potencia mayor consumo y menor autonomía para el dispositivo) y el número de repetidores que haya por el medio. Así el alcance puede estar entre los 10 y 100 metros de distancia (los repetidores provocan la introducción de distorsión que puede perjudicar los datos transmitidos).

Bluetooth puede conectar muchos tipos de aparatos sin necesidad de un solo cable, aportando una mayor libertad de movimiento. Por esta razón ya se ha convertido en una norma común mundial para la conexión inalámbrica. En el futuro, es probable que sea una norma utilizada en millones de teléfonos móviles, PC, ordenadores portátiles y varios tipos de aparatos electrónicos, como por ejemplo:

- Domótica (activación de alarmas, subida de persianas, etc.).
- Sector automovilístico (comunicación con otros vehículos).
- Medios de pago.

En una comunicación Bluetooth se pueden alcanzar tasas de transmisión de datos de 720 kbps (1 Mbps de capacidad bruta) con un rango óptimo de 10 metros (como se ha comentado antes 100 metros con repetidores).

2.7 RobotC

Como se ha comentado anteriormente los sistemas Mindstorms han tenido una gran aceptación y debido a esto se pueden encontrar una gran variedad de entornos de desarrollo y programación que permiten trabajar tanto en Linux como en Mac y Windows, ya que el firmware es código libre de distribución.

El set de Mindstorms viene acompañado del entorno de programación NXT-G, que está basado en Labview. Es un lenguaje en el cual los programas se crean por bloques, muy intuitivo, rápido y sencillo de programar producido en colaboración con Nacional Instruments.

Existen también muchas alternativas de software libre tales como BricxCC, NBC, NXC, leJOS (NXJ), pbLua o NXT# pero la herramienta que se ha utilizado en el presente proyecto es RobotC. En la siguiente tabla podemos ver que adolece cada uno y que lenguajes nos ofrecen más facilidades y rapidez que otros:

Características	NXT-G Retail	NXT-G Educational	RoboLab 2.9	NBC	NXC	Robot C	NI LabVIEW Toolkit	leJOS NXJ	pbLua	LEJOS OSEK
Tipo de lenguaje	Gráfico	Gráfico	Gráfico	Ensamblador	Como C	C	Gráfico	Java	Lua	ANSI C
Firmware	Estándar	Standard	Estándar(#1)	Estándar	Estándar	Estándar (#1)	Estándar	Modificado	Modificado	Modificado
IDE (¿incluido?)	Si	Si	Si	Si	Si	Si	No (#6)	plugins para Eclipse y NetBeans	No (#7)	Eclipse CDT(GCC+ ATMEL SAM-BA)
Windows	Si	Si	Si	Si	Si	Si	Si	Si	Si (#7)	Si
Mac OSX	Si	Si	Si	Si	Si	Aún no	Si	Si	Si (#7)	No
Linux	No	No	No	Si	Si	No	No	Si	Si (#7)	No (puede)
Eventos	No	No	Si	No	No	Si	No	Eventos estándar de java		Si (OSEK RTOS)
Multitarea	Si	Si	Si	Si	Si	Si	Si	Si		Si (OSEK RTOS)
Bluetooth Brick hacia PC	Si	Si	No	Si	Si	Si	Si	Si	Si	Si
Bluetooth Brick hacia Brick	Si	Si	No	Si	Si	Si	Si	Si	Si	Aún no
Bluetooth Brick hacia otro dispositivo	No	No	No	No	No	Si	No	Si	Si	No
I2C Support	(#5)	(#5)	Si	Si	Si	Si	Si	Si	Si	Si (EEUU sólo)
File System	Si	Si	Si	Si	Si	Si	Si	Si	Aún no	Sin planear
Floating Point	No	No	Si	No	No	Si	¿No?	Si	(#8)	Si
Datalog	No	No	Si	No	No	Si	¿No?	Si	No	Aún no

Figura 43: tabla comparación diferentes entornos

Si se observa la tabla, RobotC es un lenguaje de programación que supera al resto de lenguajes en la mayoría de aspectos.

Pero además, RobotC se ha elegido para realizar este proyecto por la cantidad de posibilidades que tiene para programar con sensores, tanto los propios de LEGO como los de otras marcas.

Esto es porque tiene compatibilidad con el protocolo I2C que nos va a permitir trabajar con sensores digitales (que son la mayoría de los fabricados por “Third Parties” como Hitechnic).

Aunque prácticamente todos los lenguajes de desarrollo presentan también esta compatibilidad, RobotC presenta una interfaz bastante cómoda para el estudio de los sensores que funcionen con dicho protocolo.

Aún así, si nos resultara complicado programar utilizando este protocolo, RobotC incluye librerías para los sensores más importantes tanto de Lego como de otras compañías para poder trabajar con ellos sin tener que preocuparnos de no saber manejar correctamente I2C.

RobotC es un lenguaje muy parecido a C, de hecho parece un micro C que hace uso de un firmware (más rápido que el original de LEGO) como sistema operativo.

Incluye una interfaz de programación típica de texto, con ayudas de rellenado, sintaxis en color, una barra lateral donde se describen las funciones que tiene implementadas RobotC para ayudar a controlar los sensores, etc. y además un depurador en tiempo real para facilitar la labor a los programadores. Dentro de la aplicación se pueden encontrar numerosos ejemplos escritos en dicho lenguaje para probarlos directamente con el robot para comprobar su funcionamiento

Para mostrar cómo resulta el entorno de programación se adjuntan diferentes capturas donde se observan diferentes partes, como puede ser la parte de las funciones y variables que tiene intrínsecas RobotC, la parte de edición de texto con su resaltado de texto y la ventana de depuración donde se muestran los errores y los warnings del compilador.

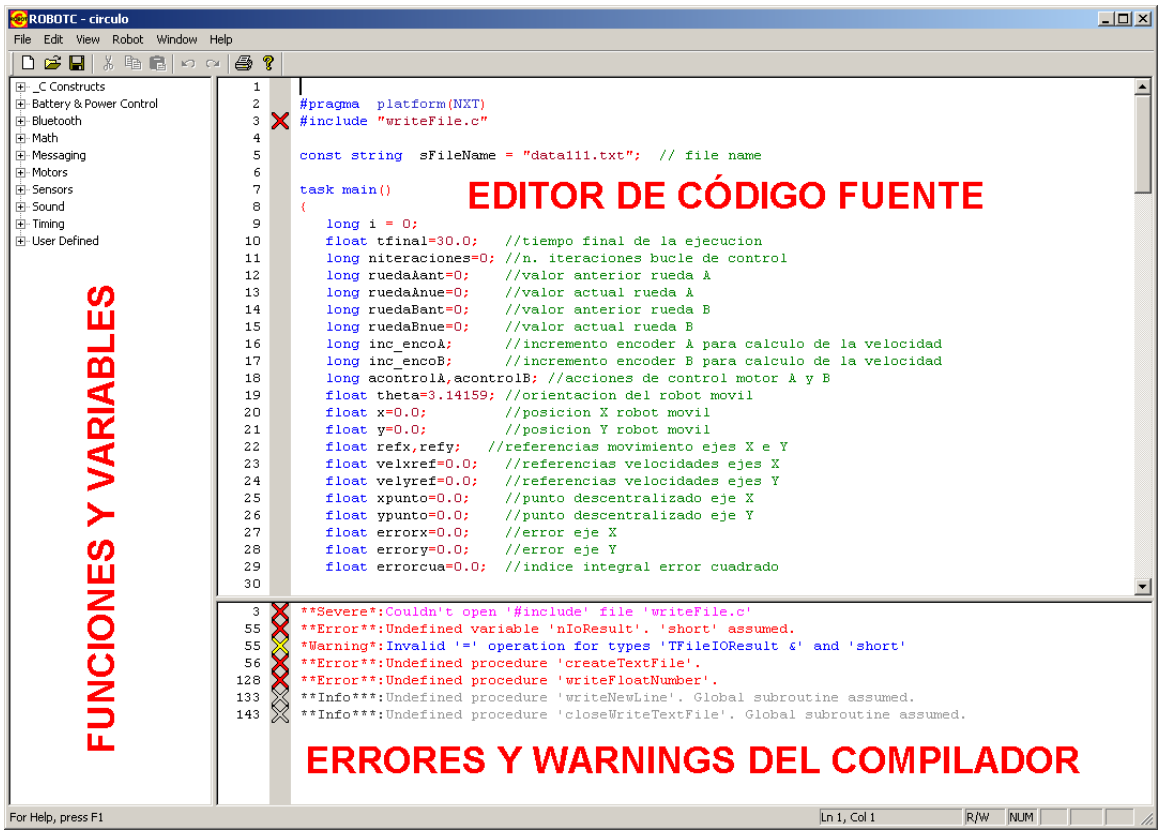


Figura 44: Interfaz de RobotC

El entorno de programación dispone de una serie de utilidades muy interesantes puesto que, por ejemplo, permite configurar de una forma muy simple los actuadores y sensores a utilizar, dispone de un administrador de archivos que permite cargar y descargar ficheros en el NXT, nos facilita la monitorización en tiempo real del estado de los motores, sensores y las variables más importantes del sistema, etc.

ROBOTC facilita también una serie de funciones de bajo nivel que permiten, entre otras cosas:

- Verificar y controlar el estado de las baterías del robot
- Establecer el control de las tareas que se ejecutan concurrentemente
- Programar relojes que permiten especificar los periodos de muestreo requeridos en las aplicaciones
- Control de los motores de corriente continua del sistema
- Obtener los valores de los sensores analógicos y/o digitales
- Establecer las comunicaciones Bluetooth

- Facilitar una librería con las funciones matemáticas más importantes
- Permitir la interacción con el usuario a partir de los botones y la pantalla del NXT, leer y escribir ficheros de texto, etc.

Una vez desarrollado y validado el programa fuente, el compilador de ROBOTC genera el código máquina que puede descargar mediante un cable USB en el ladrillo NXT.

Una vez cargado en el robot móvil se puede lanzar la ejecución del programa. Esto se puede hacer desde el propio entorno de ROBOTC (si el cable USB está todavía conectado al robot) o desde los botones del NXT.

Debido a que con el entorno de programación que se acaba de comentar se trabaja con un lenguaje de programación parecido a C, la complejidad de las tareas a realizar por el robot y de los algoritmos de control pueden ser mucho mayores.

2.8 MATLAB

MATLAB es la abreviatura de *MATrix LABoratory*, es una herramienta o software matemático que ofrece un entorno de desarrollo integrado incluyendo su propio lenguaje de programación el lenguaje M. Hoy en día, podemos instalar esta potente herramienta en plataformas Unix, Windows y Apple Mac OS X.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las *cajas de herramientas* (*toolboxes*); y las de Simulink con los *paquetes de bloques* (*blocksets*).

Es un software que dadas todas sus prestaciones es muy utilizado tanto en universidades como en centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

MATLAB fue creado por *Cleve Moler* en 1984, surgiendo la primera versión con la idea de emplear paquetes de subrutinas escritas en Fortran en los cursos de álgebra lineal y análisis numérico, sin necesidad de escribir programas en dicho lenguaje. El lenguaje de programación M fue creado en 1970 para proporcionar un sencillo acceso al software de matrices *LINPACK* y *EISPACK* sin tener que usar Fortran.

En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales.

En los siguientes apartados veremos más a fondo que nos proporciona la programación de scripts en lenguaje m.

2.8.1 Lenguaje de programación

MATLAB además de la potentísima herramienta matemática que presenta, es un lenguaje de programación, un conjunto de reglas para escribir programas de ordenador. MATLAB es un lenguaje de programación orientado al Cálculo Numérico y es difícil encontrarle cualquier otra aplicación. Desde un punto de vista estético y práctico MATLAB es un buen lenguaje de programación para realizar programas breves y simples. Pero por el contrario, no sería el más idóneo adecuado para:

- Implementación de algoritmos complejos que requieran de modelos de datos complejos organizados de forma jerárquica. Aunque con MATLAB podemos programar utilizando la orientación a objetos no puede considerarse un buen lenguaje para ello.
- Computación de alto rendimiento. El HPC es un caso de uso extremo de los recursos de cálculo. MATLAB tiene un rendimiento razonable en la mayoría de los casos pero un buen programador puede multiplicar entre

diez y cien veces la velocidad de ejecución de un programa utilizando C o Fortran.

- Grandes proyectos de software. MATLAB no es una buena elección para un programa que crece más allá de unos cuantos miles de líneas. No hay una razón única para ello pero se podría decir que la complejidad del código escala mal.

Pero lo realmente valioso de MATLAB no son sus capacidades como lenguaje sino las siguientes:

- Existe un uso generalizado de MATLAB en Ingeniería, es una herramienta de gran popularidad y es útil para una carrera profesional. Esto lo ha convertido en un estándar de-facto para la escritura de pequeños programas de simulación.
- MATLAB cuenta con una extensa biblioteca de funciones que cubren casi todas las disciplinas de la Ciencia y la Ingeniería extensamente documentada y de fácil uso.

2.8.2 Lenguaje interpretado

Los lenguajes de programación, como los lenguajes naturales escritos, no son más que una serie de normas para transmitir conceptos. Mientras el lenguaje escrito sirve para que los seres humanos se comuniquen entre ellos los lenguajes de programación se crearon para comunicarse con los ordenadores mediante una serie finita de claves.

Los lenguajes de programación también tienen gramática y léxico pero son mucho más simples que, por ejemplo, los de la lengua castellana. Los seres humanos estamos educados para convertir palabras y frases en sonidos. Hay que dotar a los ordenadores de un método para convertir el código implementado en un lenguaje de programación en órdenes que sea capaz de cumplir. Hay casi una infinidad de maneras de lograr este objetivo. A diferencia de la mayoría de los cursos sobre lenguajes de programación los describiremos por orden cronológico, aunque no rigurosamente.

Cuando apareció el ordenador programable la única manera de comunicarse con él era describir sin ambigüedad qué sucedía con cada posición de memoria. Este código de bajo nivel, llamado comúnmente ensamblador, es traducido a lenguaje máquina que ya un ordenador es capaz de entender. Aunque hoy este método de programación pueda parecer inverosímil es la mejor manera de mover máquinas lentas y con poca memoria como las de entonces.

El paso siguiente llegó con la aparición de los compiladores. A medida que los ordenadores se hacían más potentes escribir los programas en ensamblador empezó a hacerse una tarea muy laboriosa. El número de direcciones de memoria crecía exponencialmente y las arquitecturas, aunque seguían el modelo de Von Neumann, se hacían más complejas. El siguiente paso fue utilizar el mismo ordenador para traducir desde un lenguaje más humano, de alto nivel, a ensamblador. El ensamblador pasó de ser un lenguaje de uso a un léxico intermedio. El programa que convierte este código de alto nivel se llama compilador.

Este planteamiento tiene una ventaja adicional. El código ensamblador no es el mismo para todas las arquitecturas.
Un programa compilado para arquitecturas x86 no puede ejecutarse en SPARC o POWER pero el código es el mismo. El programa de Kernighan y Ritchie:

```
#include "stdio.h"
int main()
{
printf("Hello, world!\n");
}
```

Produce exactamente el mismo resultado en cualquier ordenador siempre que disponga de un compilador de lenguaje C. Esto asegura la portabilidad a nivel de código, no a nivel de ejecutable.

El paso siguiente es poder utilizar un ensamblador independiente de cada arquitectura mediante un traductor de código propio a código máquina. Esta aplicación se llama máquina virtual. Una máquina virtual es tan lista como se desee (mucho más lista que un procesador) y realizará tareas como la declaración de variables, la liberación de memoria o la gestión del flujo de ejecución. El conjunto compilador y máquina virtual se denomina intérprete y los lenguajes que soportan este funcionamiento se llaman lenguajes

interpretados. Que el código sea ejecutado por un programa y no por el propio ordenador es mucho más lento, por este motivo las máquinas virtuales no se popularizaron hasta finales de los noventa.

El paso siguiente es hacer desaparecer incluso este ensamblador intermedio y con él el compilador. Ya no existe un compilador y una máquina virtual sino que sólo un programa, el intérprete, realiza todo el trabajo. Este último planteamiento no es necesariamente superior en eficacia o rendimiento a una máquina virtual, simplemente es más fácil de diseñar e implementar. MATLAB pertenece a este último grupo.

2.8.3 Lenguaje dinámico

En muchos lenguajes de programación como C o Fortran es imprescindible declarar cada una de las variables que se van a utilizar en la programación. La definición estricta de declaración es la de identificar un determinado espacio en la memoria del ordenador con un nombre.

Volviendo otra vez a un C que cualquiera pueda entender la declaración **int** a; significa que un espacio en la memoria física lo suficientemente grande como para almacenar un entero va a recibir el nombre de a. Estos lenguajes, los que asocian variables a memoria, se llaman estáticos

La llegada de los lenguajes interpretados permitió manejar la memoria de una manera mucho más versátil. Java, que aunque es interpretado es también estático, incluye un recolector de basura que descarga al programador de la tarea de limpiar la memoria. Pero la mayoría de los lenguajes interpretados modernos como Python o Ruby son además dinámicos. En un lenguaje dinámico no existen declaraciones porque el concepto de variable es distinto, ya no es el nombre que se asocia a un espacio en la memoria, es el nombre de un valor. De esta manera la variable tiene un sentido mucho más natural, más matemático. MATLAB es un lenguaje dinámico aunque no puede considerarse moderno.

Desde el punto de vista del intérprete cualquier variable o estructuras de variables son mutables en tiempo de ejecución complicando significativamente el manejo de memoria.

Programar con un lenguaje dinámico es completamente distinto hacerlo con uno estático. La mayor versatilidad suele venir acompañada de mayor coste computacional o de nuevos errores de programación. No debemos perder nuca de vista que la programación es la manipulación de datos almacenados en la memoria de un ordenador y con un lenguaje dinámico estamos más lejos de los mismos.

2.8.4 Interfaz gráfica

La interfaz gráfica de MATLAB es prácticamente idéntica en cualquiera de sus versiones independientemente del sistema operativo.

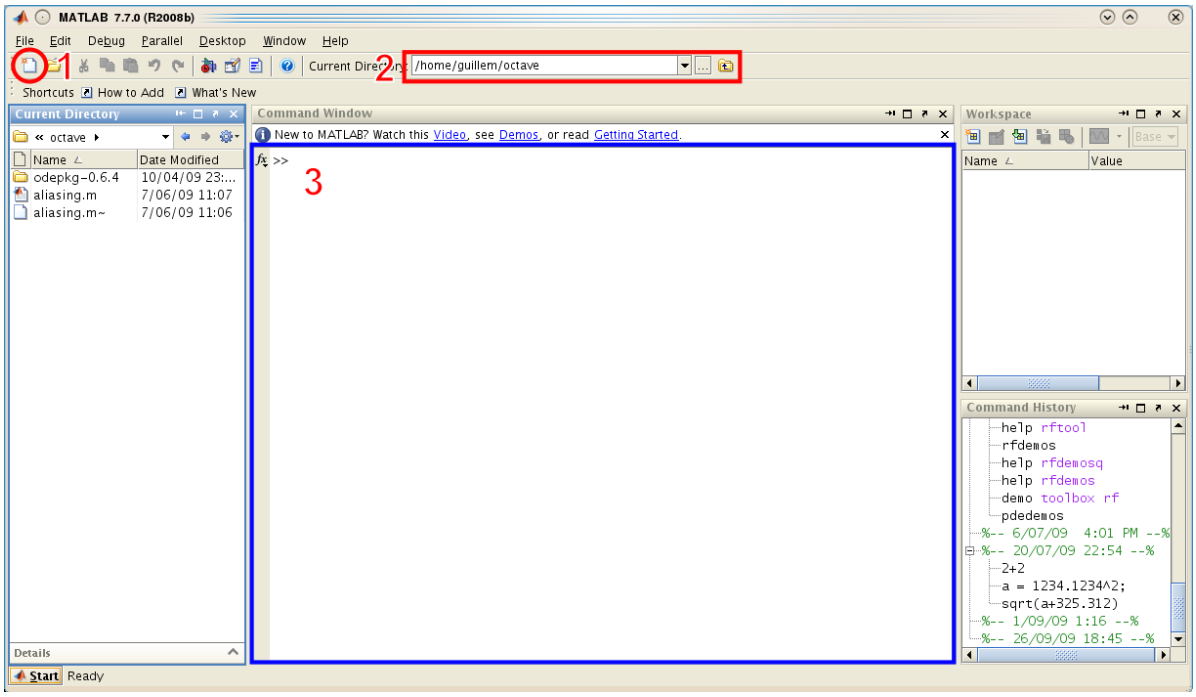


Figura 45: Captura de la interfaz gráfica de MATLAB R2008b

Vemos que la ventana principal está dividida en apartados con una función específica. Cada uno de estos apartados, a excepción del menú, es una ventana que puede moverse dentro de la propia aplicación. Esto permite que ordenemos MATLAB para ajustarlo mejor a nuestras necesidades. Las tres únicas zonas que de momento nos interesan están marcadas con un número en la imagen.

El icono señalado con el número 1 significa nuevo archivo y sirve para abrir el editor de MATLAB.

El recuadro señalado con el número 2 es el diálogo para seleccionar el directorio de trabajo. A medida que vayamos escribiendo código lo guardaremos en algún lugar del ordenador. Para poder utilizarlos en un futuro es importante que MATLAB sepa dónde lo hemos dejado. MATLAB ya sabe, porque así viene configurado de fábrica, dónde tiene guardadas las funciones propias de la aplicación y de los distintos toolkits pero no sabe dónde están las que hemos escrito.

Por último, pero no menos importante, el número 3 es la consola de MATLAB. Como quedó claro en la introducción, en realidad MATLAB no es más que un intérprete para un lenguaje de programación y nuestra vía directa de comunicación con el mismo es la consola. De hecho, no existe ninguna acción de la interfaz gráfica que no pueda ejecutarse también mediante la consola. De hecho, cuando ejecutamos un programa no es ni siquiera imprescindible que la interfaz gráfica esté abierta.

La siguiente pieza es el editor:

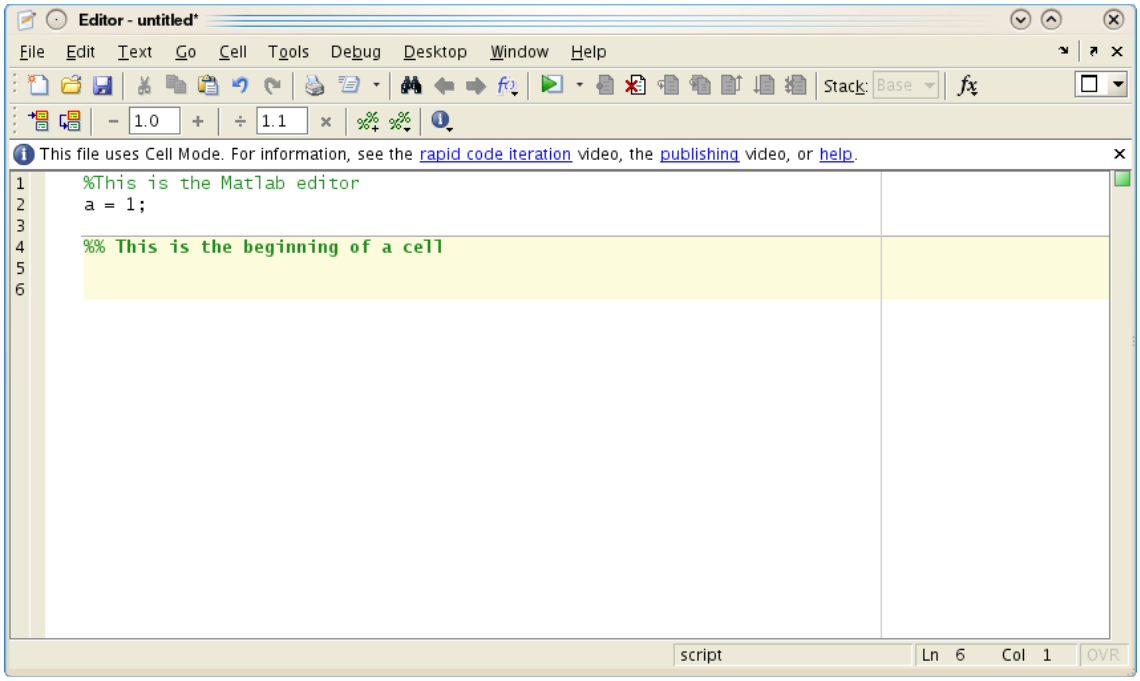


Figura 46: Captura del editor de MATLAB R2008b

La definición de programar es escribir código y para ser realmente productivos es importante utilizar una buena herramienta y conocerla. No es ni

mucho menos necesario utilizar el editor de MATLAB para escribir nuestros scripts pero se trata de una buena opción.

El editor cuenta con casi todas las capacidades que se esperan de una herramienta de programación moderna:

- Coloreado de código
- Análisis sintáctico capaz de detectar errores antes de ejecutar el código
- Depurador integrado

Una de las características que ha integrado en las últimas versiones es el modo celda que nos ayudará a dividir grandes archivos en partes ejecutables independientemente sólo comentando el código de una manera particular.

La interfaz gráfica nos sirve también para consultar la documentación del programa. Es completa, extensa y de calidad.

3.- DESARROLLO PRÁCTICO

3.1 Preámbulos y configuración

Llegados a este punto y una vez finalizado la exposición de las bases teóricas necesarias para el desarrollo de nuestros objetivos, se continúa con la descripción de la parte práctica del proyecto.

3.1.1 Montaje del robot NXT

Siguiendo el guión de las finalidades de este proyecto, el montaje del robot NXT no puede ser de cualquier manera. Se necesita que cumpla unas necesidades de movilidad, de tracción y de dirección. De las opciones planteadas en la parte teórica, se opta por la forma de triciclo: dos ruedas motrices que controlan tracción y dirección al mismo tiempo con un mecanismo diferencial y una rueda trasera más pequeña y libre.

Figura 47: Robots LEGOS de frente

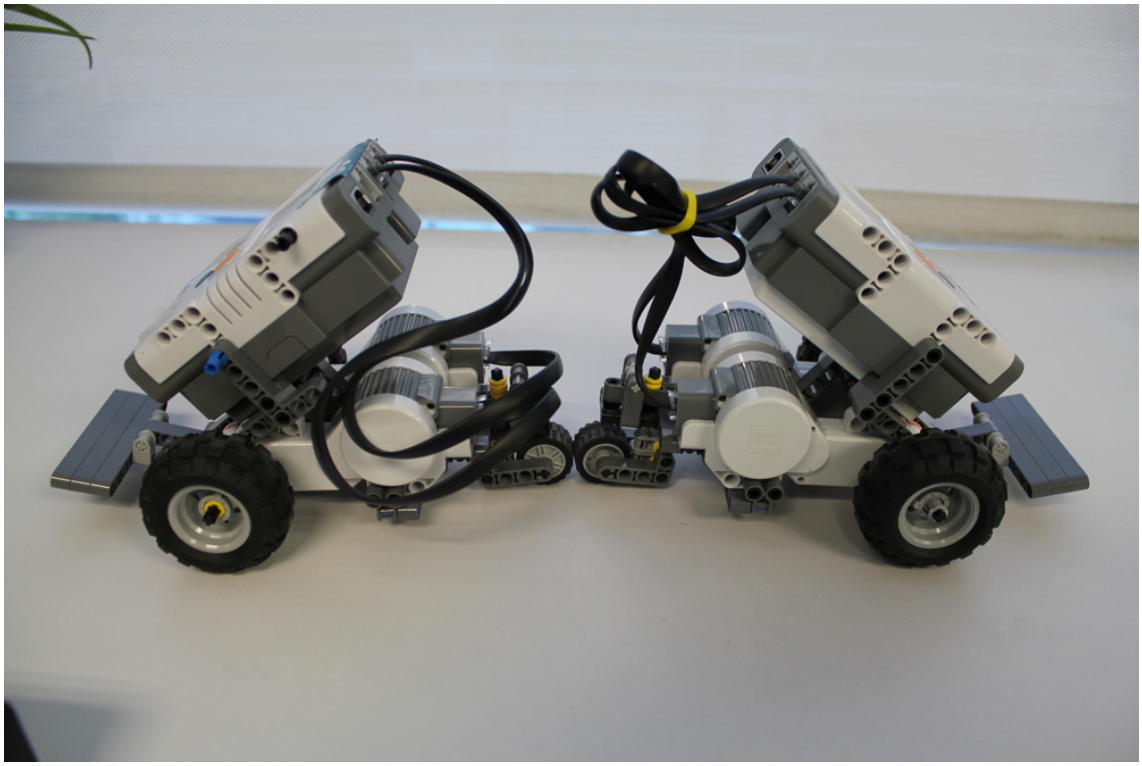


Figura 48: Robots LEGOS de perfil

3.1.2 Instalación del driver USB de Lego

Para que el ordenador reconozca el robot, se debe descargar desde http://mindstorms.lego.com/en-us/support/files/default.aspx el driver de Lego Mindstorms NXT e instalarlo. Ahora cuando se conecte el robot al ordenador por USB y se encienda, aparecerá entre los dispositivos del sistema.

3.1.3 Instalación de RobotC y firmware del robot

En el presente proyecto, se ha trabajado con el entorno de desarrollo RobotC, tal y como hemos mencionado en apartados anteriores. La primera tarea que deberemos realizar es descargar el producto desde su propia web: http://www.robotc.net/download/nxt/ y posteriormente instalarlo en el ordenador.

Una vez instalado RobotC, es bueno hacer algunos cambios en la configuración inicial, por ejemplo, que la interfaz del programa sea para súper usuario, ya que de esta manera, tendremos muchas más opciones y funciones

para poder utilizar en caso de necesitarlas. Esto lo haremos desde: Window -> Menu Level -> Super User.

A continuación conectaremos el robot al ordenador por USB y deberemos instalarle el firmware correspondiente a la versión de RobotC dado que Mindstorms NXT cuenta con su propio firmware, es necesario reemplazar el original que tienen los robots. Para ello entramos al menú: Robot -> Download Firmware y se nos abrirá la siguiente ventana:

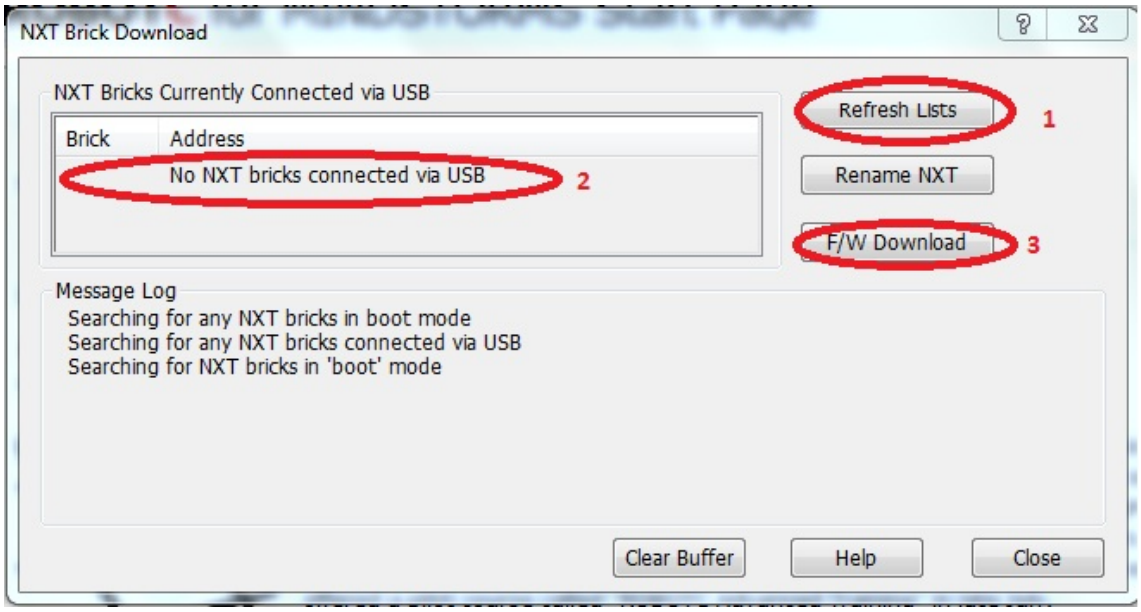


Figura 49: Ventana para instalar firmware en el NXT

Primero actualizamos la lista de bricks conectados al ordenador, pulsando en el botón Refresh Lists. A continuación deberá salir en el panel NXT Bricks Currently Connected via USB el nombre y la dirección MAC del robot al que queremos instalar el firmware, se selecciona y pulsamos F/W Download, se abrirá una nueva ventana en la que aparecerá un archivo con formato .rfw y lo seleccionamos. Seguidamente comenzará a instalarse el nuevo firmware en el ladrillo del robot.

Advertencia: Aunque es un proceso aparentemente sencillo el que se acaba de describir, corremos el peligro de realizar algo algún error y dejar el Brick bloqueado e inutilizable. En ningún caso deberemos desconectar el robot del USB durante el proceso de descarga del firmware.

3.1.4 Instalación y configuración de MATLAB

Como se ha comentado anteriormente en el apartado teórico, para facilitar el estudio de la generación de curvas de Bézier, se ha utilizado la herramienta MATLAB, deberemos instalarla en el ordenador y configurar la interfaz a gusto del usuario para poder trabajar cómodamente con ella.

3.1.4.1 Java Developement Kit

Por supuesto para trabajar con MATLAB se necesita instalar la Máquina Virtual Java en el ordenador. Desde su página web http://www.java.com/en/download/manual.jsp se puede descargar la última versión e instalarla. Hay que asegurarse de que se instala al menos la versión 5 ya que es la mínima necesaria para que funcione la herramienta de trabajo seleccionada.

3.2 Curvas de Bézier

3.2.1 Introducción

Se denomina curvas de Bézier a un sistema que se desarrolló hacia los años 1960, para el trazado de dibujos técnicos, en el diseño aeronáutico y de automóviles. Su denominación es en honor a Pierre Bézier, quien ideó un método de descripción matemática de las curvas que se comenzó a utilizar con éxito en los programas de CAD.

Las curvas de Bézier fueron publicadas, por primera vez en 1962 por el ingeniero de origen francés Pierre Bézier, que las usó posteriormente, con profusión, en el diseño de las diferentes partes de los cuerpos de un automóvil, en sus años de trabajo en la Renault. Las curvas fueron desarrolladas por Paul de Casteljau usando el algoritmo que lleva su nombre. Se trata de un método numéricamente estable para evaluar las curvas de Bézier.

Posteriormente, los inventores del PostScript, lenguaje que permitió el desarrollo de sistemas de impresión de alta calidad desde el ordenador, introdujeron en ese código el método de Bézier para la generación del código de las curvas y los trazados. El lenguaje PostScript sigue empleándose ampliamente y se ha convertido en un estándar de calidad universal; por ello, los programas de diseño vectorial como Adobe Illustrator, el extinto Macromedia FreeHand, Corel Draw, tres de los más importantes programas de dibujo vectorial y otros como Inkscape, denominan como "bézier" a algunas de sus herramientas de dibujo, y se habla de "Trazados bézier", "pluma bézier", "lápiz bézier", etc. Su facilidad de uso la ha estandarizado en el diseño gráfico, extendiéndose también a programas de animación vectorial como Adobe Flash, y retoque fotográfico como Photoshop y Gimp, donde se usa para crear formas cerradas o selecciones.

La idea de definir geométricamente las formas no es demasiado compleja: un punto del plano puede definirse por coordenadas. Por ejemplo, un punto A tiene unas coordenadas (x_1, y_1) y a un punto B le corresponde (x_2,y_2). Para trazar una recta entre ambos basta con conocer su posición.

Si en lugar de unir dos puntos con una recta se unen con una curva, surgen los elementos esenciales de una curva Bézier: los puntos se denominan puntos de anclaje o nodos. La forma de la curva se define por unos puntos invisibles en el dibujo, denominados puntos de control, manejadores o manecillas.

3.2.2 Definición y tipos de curva

Los polinomios de Bernstein, se emplean en teoría de la aproximación para demostrar el Teorema de Weierstrass de aproximación uniforme de funciones continuas por polinomios. Su construcción es muy sencilla a partir de la fórmula del binomio de Newton:

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}, \qquad \binom{n}{i} = \frac{n!}{i!(n-i)!}.$$

Si tomamos a = t, b = 1 − t en la expresión anterior, obtenemos:

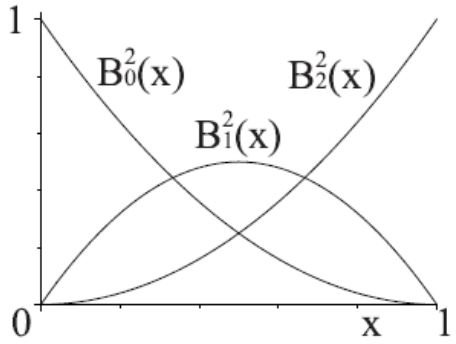
$$1 = (t + 1 - t)^n = \sum_{i=0}^{n} B_i^n(t), \qquad B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}$$

donde B (t) es el polinomio i-ésimo de Bernstein de grado n.

Por ejemplo los polinomios de Bernstein de grado dos son:

$$B_0^2(t) = (1-t)^2, \qquad B_1^2(t) = 2t(1-t), \qquad B_2^2(t) = t^2.$$

Estos polinomios forman una base alternativa {Bn0 (t),. . ., Bnn(t)} de los polinomios de grado n o inferior en una variable t y, frente a la base canónica, presentan la ventaja de ser todos del mismo grado.



La demostración de que forman base es sencilla. Echemos una ojeada al polinomio n-ésimo, Bnn(t) = tn. Coincide con el último polinomio de la base canónica. El anterior, Bnn−1(t) = n(tn−1 − tn), involucra tan sólo dos polinomios de la base canónica. Y así sucesivamente el polinomio Bn i (t), por el término (1 − t) ^n−i, tiene coeficientes no nulos sólo para los monomios ti,. . ., tn, por lo cual la matriz de cambio de base es triangular superior:

$$\begin{pmatrix} B_0^n(t) \\ \vdots \\ B_n^n(t) \end{pmatrix} = \begin{pmatrix} 1 & \cdots & (-1)^n \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ t^n \end{pmatrix}$$

y, por ello, su determinante es el producto de los términos de la diagonal principal, que corresponden al coeficiente del monomio de menor grado de cada polinomio Bni (t), que es no nulo. Con lo cual, el determinante es no nulo y los polinomios de Bernstein forman base de los polinomios de grado n o inferior.

Podremos representar, pues, las curvas polinómicas de grado n como combinación de estos polinomios:

$$c(t) = \sum_{i=0}^{n} c_i B_i^n(t), \qquad t \in [0,1]$$

Donde todos los coeficientes ci son puntos del plano o del espacio afín, según que la curva sea plana o espacial. A estos coeficientes los denominaremos vértices del polígono de control, {c0,. . ., cn}, de la curva de Bézier c(t). Una curva de grado n tiene, pues, un polígono de control de n + 1 vértices.

La curva de Bézier de grado *n* puede ser generalizada de la siguiente manera. Dados los puntos $\mathbf{P}_0$, $\mathbf{P}_1$,..., $\mathbf{P}_n$, la curva de Bézier es del tipo:

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} \mathbf{P}_i (1-t)^{n-i} t^i = \mathbf{P}_0 (1-t)^n + \binom{n}{1} \mathbf{P}_1 (1-t)^{n-1} t + \cdots + \mathbf{P}_n t^n \ , \ t \in [0,1]$$

Ejemplos de diferentes curvas:

Una curva cuadrática de Bézier es el camino trazado por la función B(*t*), dados los puntos: P_0, P_1, y P_2:

$$\mathbf{B}(t) = (1-t)^2 \mathbf{P}_0 + 2t(1-t)\mathbf{P}_1 + t^2 \mathbf{P}_2 \ , \ t \in [0,1].$$

Las fuentes de letras TrueType usan curvas de Bézier desdobladas compuestas por curvas cuadráticas de Bézier.

Una Curva cúbica de Bézier está definida por cuatro puntos del plano o del espacio tridimensional, P_0, P_1, P_2 y P_3 definen una curva cúbica de Bézier. La curva comienza en el punto P_0 y se dirige hacia P_1 y llega a P_3 viniendo de

la dirección del punto P_2. Usualmente, no pasará ni por P_1 ni por P_2. Estos puntos sólo están ahí para proporcionar información direccional. La distancia entre P_0 y P_1 determina "qué longitud" tiene la curva cuando se mueve hacia la dirección de P_2 antes de dirigirse hacia P_3.

La forma paramétrica de la curva es:

$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^3 + 3\mathbf{P}_1 t(1-t)^2 + 3\mathbf{P}_2 t^2(1-t) + \mathbf{P}_3 t^3 \ , \ t \in [0,1].$$

El ejemplo de una curva de orden cinco (*n* = 5) como las que se han utilizado para la generación de las curvas de los robots quedaría como:

$$\mathbf{B}(t) = \mathbf{P}_0(1-t)^5 + 5\mathbf{P}_1 t(1-t)^4 + 10\mathbf{P}_2 t^2(1-t)^3 + 10\mathbf{P}_3 t^3(1-t)^2 + 5\mathbf{P}_4 t^4(1-t) + \mathbf{P}_5 t^5 \ , \ t \in [0,1].$$

3.2.3 Aplicaciones

Las curvas de Bézier han sido ampliamente usadas en los gráficos generados por ordenador para modelado de curvas suaves. Como la curva está completamente contenida en la envolvente convexa de los puntos de control, dichos puntos pueden ser visualizados gráficamente sobre el área de trabajo y usados para manipular la curva de una forma muy intuitiva. Las transformaciones afines tales como traslaciones y rotaciones pueden ser aplicadas, con gran facilidad, a las curvas, aplicando las transformaciones respectivas sobre los puntos de control.

Las curvas cuadráticas y cúbicas son muy corrientes. Las curvas de grados superiores son más difíciles de evaluar. Cuanto más complejas son las superficies que se necesitan, las curvas de bajo orden son menos apropiadas. Para garantizar la suavidad de las curvas el punto de control en el que se juntan dos curvas y el punto de control sobre cualquiera de los lados debe ser colineal. Esta opción está frecuentemente desactivada en programas como Adobe Illustrator o Inkscape. Estas curvas poli-Bézier pueden ser observadas en el formato de archivo SVG.

El método más simple para rastrerizar una curva de Bézier es evaluarla en muchos puntos espaciados, muy próximos entre sí, y escanearla aproximando la secuencia de segmentos lineales.

Esta manera de proceder no garantiza un resultado con la suficiente suavidad porque los puntos pueden estar espaciados demasiado separados. A la inversa, se pueden generar bastantes puntos de control en áreas donde la curva esta cercana a la forma lineal.

Un método adoptado, muy común, es la subdivisión recursiva, en el que los puntos de control de la curva son ajustados para ver si la curva se aproxima a segmentos lineales sin pequeñas tolerancias. Si esto no se logra, la curva es subdividida paramétricamente en dos segmentos y el mismo procedimiento se aplica por recursividad a cada mitad.

También hay métodos que usan la diferenciación, pero se debe tener cuidado y analizar los errores de propagación. Los métodos analíticos donde un desdoble es intersecado con cada línea escaneada hallando raíces de polinomios de grado tres, por segmentación cúbica, y con múltiples raíces, pero no son frecuentes en la práctica.

En el presente proyecto hemos utilizado las curvas de Bézier como referencias para la generación de trayectorias de los robots, de manera que introduciendo una serie de puntos, bien sean 4 o 5, se calcule y se genere la curva para que uno, dos o tres robots las realicen de manera coordinada y sincronizada a fin que puedan transportar algún objeto, como por ejemplo una viga.

3.3 Estudio del movimiento

En el apartado del estudio del movimiento, abordamos los aspectos tanto de la coordinación de las trayectorias de los robots, así como las funciones que nos ofrece RobotC para utilizar los motores de los Legos para aplicarles movimiento.

3.3.1 Coordinación de trayectorias

El problema de la coordinación de curvas es de aspecto más matemático que mecatrónico. Es por ello que recurrimos a MATLAB para poder generar trayectorias y poder comprobar qué es lo que trazaría cada uno de los robots.

Los cálculos realizados eran los siguientes:

- Para cada una de las iteraciones calcula la diferencia entre el punto anterior y el nuevo punto calculado:

```
deltax(i)=x(i)-x(i-1);
deltay(i)=y(i)-y(i-1);
```

- Mediante la función atan2 se calcula el arco tangente de los cuatro cuadrantes de la parte real de los elementos deltax y deltay.

```
alfa=atan2(deltay(i),deltax(i));
```

- Sabiendo ya la orientación y la situación de los puntos, calculamos los mismos puntos pero a una distancia determinada de la curva central, el valor de la distancia se puede modificar según la aplicación que se desee realizar:

```
posx1(i)=x(i)+d*sin(alfa+oinicial);
posy1(i)=y(i)-d*cos(alfa+oinicial);
posx2(i)=x(i)-d*sin(alfa+oinicial);
posy2(i)=y(i)+d*cos(alfa+oinicial);
```

Donde posx1 y posy1 son las posiciones X e Y del robot 1, es decir, el que se encuentra a la derecha, y posx2 y posy2 son las posiciones X e Y del robot situado a la izquierda de la curva central.

Una vez calculados los puntos solo queda generar una gráfica para poder observar el comportamiento tendrá la generación de la curva y la coordinación de ambos robots. De nuevo, con la misma herramienta de MATLAB, es muy sencillo graficar dichos puntos, ya que posee funciones dedicadas a dicha tarea, en concreto la función plot(x,y); donde x e y son vectores de puntos. Podemos observar algunos de los resultados obtenidos:

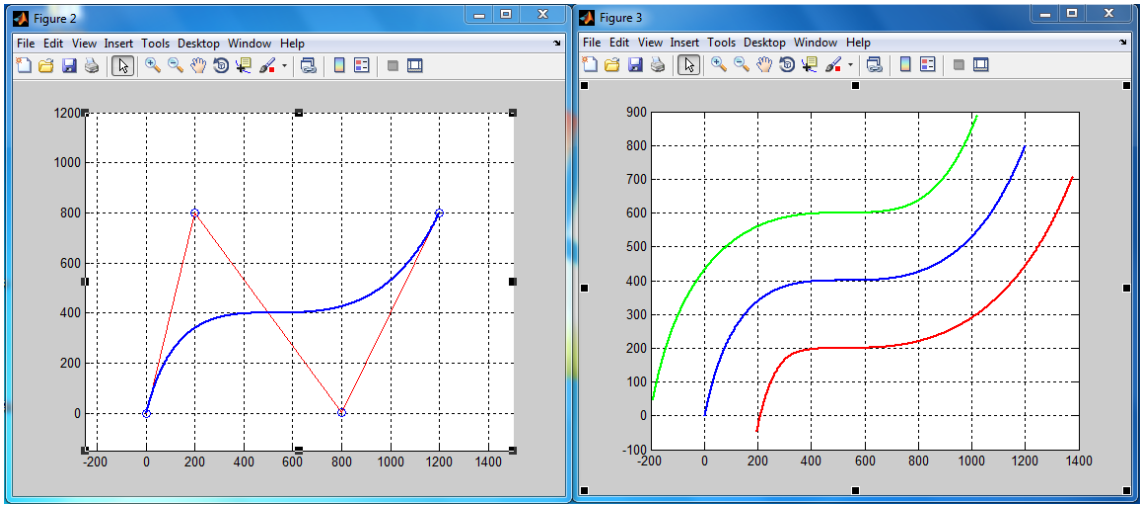


Figura 50: Gráficas de la coordinación de curvas para 4 Puntos

En estas gráficas, se observa que primero se genera una curva con 4 Puntos, y a su lado tenemos otra gráfica que sería el resultado de los cálculos realizados para la coordinación de los robots. En color azul se presenta la curva central, de color rojo el robot situado a la derecha, y de color verde el robot situado a la izquierda de la curva. La distancia de cada una de las curvas a la curva central en este ejemplo es de 200 unidades, ya que de momento, no trabajamos con ninguna medida de longitud en concreto.

3.3.2 Control de los motores

Como se ha dicho en la presentación de este punto, se necesita conocer las funciones que tratan el movimiento de los robots que harán girar las ruedas de los robots móviles. Aunque en la aplicación de control final que se ha desarrollado solamente se han utilizado dos de estas funciones, era de gran importancia saber que herramientas nos ofrece RobotC para trabajar con los motores de los robots.

Con la función motor[], se introduce la velocidad que queremos que tenga el motor. Se pueden introducir valores del rango de -100 a 100 obteniendo los siguientes resultados:

- 0 -> El motor está parado, no hay movimiento.
- [1, 99] -> El motor se mueve hacia adelante.

- 100 -> Máxima potencia hacia adelante.
- [-1, -99] -> El motor se mueve hacia atrás.
- -100 -> Máxima potencia hacia atrás.

Como ya se ha comentado anteriormente, en este proyecto se ha utilizado una configuración triciclo. Por ejemplo, para hacer que el robot gire en una dirección, simplemente hay que aumentar la velocidad de la rueda contraria al sentido que queremos girar.

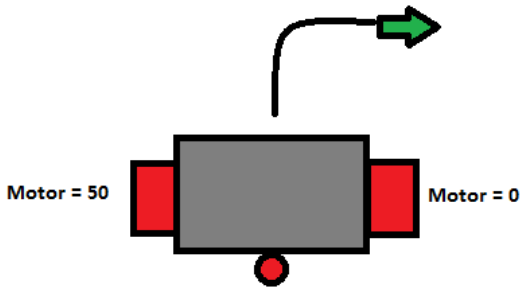


Figura 51: Ejemplo de giro a derecha del robot

Otra de las funciones que se ha utilizado para el desarrollo de control es nMotorEncoder. Gracias a esta función, se puede obtener el valor del encoder de cada motor para saber exactamente cuanto a girado la rueda:

```
ruedaAnue=nMotorEncoder[motorA];
ruedaBnue=nMotorEncoder[motorB];
```

Con estas dos líneas de código, guardamos el valor de los encoders, para poder saber cuanto a girado la rueda, se compara con el resultado de la iteración anterior y calculamos la velocidad angular de cada una de las ruedas.

Otra función importante pero que no hemos tenido la necesidad de utilizar es por ejemplo nSyncedMotors, que nos ofrece la posibilidad de sincronizar la velocidad de los motores, ya que a veces, aunque se introduzcan los mismos valores a los motores, puede que uno corra más que otro. Esta es una buena función si hay que describir líneas rectas.

3.4 Estrategia del control cinemático

Llegados al punto en el que ya tenemos generadas las trayectorias que deseamos que realicen los robots y como aplicarles movimientos, se debía elegir algún método geométrico para realizar el control de la cinemática del autómata. En este caso, se abordó el seguimiento de trayectorias mediante algoritmos de Control de Posición por Punto Descentralizado.

3.4.1 Punto Descentralizado

El control de posición por punto descentralizado se establece a partir de la posición y velocidad de un punto que está separado una distancia ***e*** del eje de tracción del robot.

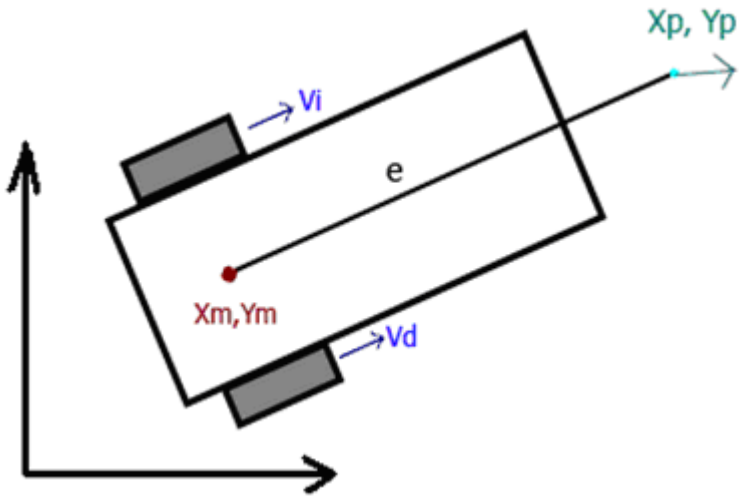


Figura 52: Punto descentralizado

De este modo las coordenadas del punto descentralizado vienen definidas por:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m + e\cos(\theta) \\ \dot{y}_m + e\sin(\theta) \end{bmatrix}$$

Se calcula la derivada (velocidad) de ese punto:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m & -e\cos(\theta)\dot{\theta} \\ \dot{y}_m & +e\sin(\theta)\dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix}$$

Para conocer la posición del robot en todo momento se parte de la ecuación cinemática directa diferencial:

$$\begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & 1/2b \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

Donde sustituyendo queda:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & 1/2b \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \frac{1}{2b} \begin{bmatrix} b\cos(\theta)+e\sin(\theta) & b\cos(\theta)-e\sin(\theta) \\ b\sin(\theta)-e\cos(\theta) & b\sin(\theta)+e\cos(\theta) \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

Las velocidades de las ruedas se pueden obtener despejando de la ecuación anterior:

$$\begin{bmatrix} v_i \\ v_d \end{bmatrix} = \frac{1}{e} \begin{bmatrix} e\cos(\theta)+b\sin(\theta) & e\sin(\theta)-b\cos(\theta) \\ e\cos(\theta)-b\sin(\theta) & e\sin(\theta)+b\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix}$$

Para el desarrollo del control cinemático de los NXT se ha utilizado la siguiente ecuación:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} k_{v_x} \dot{x}_{ref} \\ k_{v_y} \dot{y}_{ref} \end{bmatrix} + \begin{bmatrix} k_{p_x} & 0 \\ 0 & k_{p_y} \end{bmatrix} \begin{bmatrix} x_{ref} - (x_m + e\cos\theta) \\ y_m - (y_m + e\sin\theta) \end{bmatrix}$$

En definitiva, el algoritmo de control de Posición por Punto Descentralizado quedaría definido de la siguiente manera:

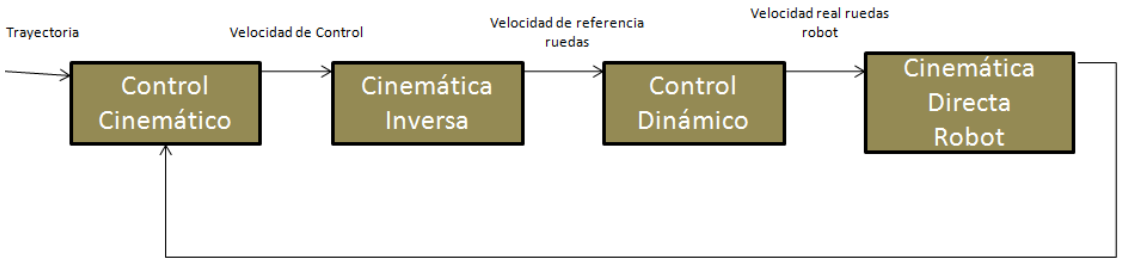


Figura 51: Esquema de control por Punto Descentralizado

Entre las ventajas de la estrategia del punto descentralizado cabe destacar que no es necesario conocer la trayectoria completa para aplicarla sino que al mismo tiempo que se calcula la velocidad necesaria para que el robot alcance la siguiente posición, se calcula el siguiente punto en función de la posición actual donde se encuentra el robot en ese momento. Lo cual ofrece precisión y una continuidad uniforme en el movimiento.

3.5 Estudio de comunicación mediante Bluetooth

En este apartado vamos a desarrollar la comunicación entre dos robots LEGOS Mindstorms NXT mediante tecnología Bluetooth. Era una necesidad que se presentó a la hora de sincronizar el inicio de las carreras de los robots, ya que por el contrario, si uno de los robots móviles empezaba antes que el otro, aunque sea por un intervalo de tiempo muy pequeño, y que aparentemente la percepción humana no lo consigue apreciar, se nos

presentaban descoordinaciones a lo largo de las trayectorias recorridas, provocando que la viga transportada por las máquinas cayera al suelo.

Dado que era un campo nuevo a tratar en RobotC, era aconsejable buscar en los samples que proporciona este software alguna aplicación donde se utilizara la tecnología Bluetooth, para así poder entrar más a fondo y comprender de forma eficiente el mecanismo de las diferentes funciones que se presentan para la comunicación mediante Bluetooth.

3.5.1 Sincronización del inicio de carrera

Durante las primeras pruebas que se realizaron con los robots, se ejecutaban los programas de forma manual, dicho literalmente. El procedimiento utilizado era el siguiente:

Se descargaba cada una de las aplicaciones en su correspondiente robot, y una vez colocados correctamente en sus posiciones iniciales, se pulsaban los botones de color naranja que tienen los ladrillos NXT para que comenzase su ejecución.

Como se ha dicho anteriormente, es prácticamente imposible que la mano humana pulse los 2 botones en el mismo intervalo de tiempo, por lo que provocaba siempre un retraso en alguno de los robots, y que a lo largo de la trayectoria perdían notablemente la coordinación provocando la caída del objeto transportado.

Para solucionar este problema, se decidió sincronizar el inicio de las carreras mediante Bluetooth. Detallaremos paso a paso los procedimientos utilizados para este problema:

- Configuración del Bluetooth para cada uno de los robots. Aunque la configuración se puede realizar manualmente con cada uno de los bricks de los robots, es más eficiente utilizar las funciones para Bluetooth que proporciona RobotC, de forma que no tendremos que configurar el dispositivo cada vez que lo encendamos, y se configurará de manera automática al ejecutarse la aplicación.

```
bBTSkipPswdPrompt    = true;          // Simply use default password. No need for user entry.
bBTHasProgressSounds = true;          // Play speaker tones when BT connected / disconnected.
bBTDebugTrace        = true;  // Generate messages on internal BT operation to "debug stream" output.
```

De esta manera desactivamos la petición de contraseñas en el momento de la conexión. También activamos los altavoces para que suenen cuando el Bluetooth se conecte o se desconecte.

- Seguidamente activamos el BT también para cada uno de los robots asignándoles a cada uno un nombre, para que en el momento de la conexión se puedan reconocer. Los nombres utilizados son “Rob1” para el robot situado a la derecha, y “Rob2” para el robot situado a la izquierda.

```
startCommandMessage("BT 'On'");
BluetoothOn();
delayBetweenCommands();


startCommandMessage("Set Brick Name");
SetBrickName("Rob1");
delayBetweenCommands();
```

```
startCommandMessage("BT 'On'");
BluetoothOn();
delayBetweenCommands();

startCommandMessage("Set Brick Name");
SetBrickName("Rob2");
delayBetweenCommands();
```

- La mejor forma de sincronizar esta tarea es que se ejecute uno de los robots, y se quede en un bucle de espera hasta que el otro se ejecute, y le envíe un mensaje para que pueda salir del bucle y comenzar los dos autómatas las trayectorias al mismo tiempo. En este proyecto, se ha decidido que primero se ejecutará el robot de la derecha cuyo nombre, tal y como se ha comentado en el apartado anterior es Rob1, y este se quede esperando al robot de la izquierda, Rob2.

```
//bucle de espera para sincronizarse
while (true)
{
  if (bQueuedMsgAvailable())
  {
    word temp;
    ClearMessage();
    temp = message;

    if (temp != 1)
      continue;

    else if (temp == 1)
      break;
  }
}
```

- Teniendo ya a Rob1 esperando hasta que otro robot le envíe un mensaje su bandeja de entrada podemos ya ejecutar el LEGO de la izquierda, Rob2. Con la función ConnectPort("Rob1"); ya definida al inicio de la aplicación, conseguimos que Rob2 se conecte mediante BT a Rob1 por el puerto número 1. Los robots LEGOS disponen de 3 puertos para conectarse por BT.
- Finalmente cuando los dos robots ya están conectados vía BT, Rob2 le envía a Rob1 el mensaje que está esperando para salir de su bucle de espera y poder comenzar los dos síncronamente las trayectorias.

```
eraseDisplay();
bNxtLCDStatusDisplay=true;
wait1Msec(500);
sendMessageWithParm(1, 0, 0);
wait1Msec(100);
```

3.6 Aplicación de control desarrollada

El principal objetivo era que los dos prototipos de LEGO NXT realizaran trayectorias basadas en curvas de Bézier, suponiendo que tuvieran que llevar una carga muy pesada entre los dos porque uno solo no pudiera con ella, coordinar y sincronizar sus movimientos para desplazarse al unísono y de esta forma mantener la distancia de uno con respecto al otro constante.

Antes de empezar a implementar los algoritmos para conseguir el comportamiento necesario se realizaron pruebas a fin de establecer las

mejores herramientas para que los prototipos describieran unos movimientos deseados.

Esto incluía la creación de funciones que vamos a ir explicando poco a poco a lo largo de este punto.

La primera parte del proyecto, se centró en generar curvas de Bézier para que posteriormente los robots móviles trazaran estas curvas como trayectorias. Para ello y con la herramienta MATLAB, se implementaron una serie de funciones que nos permitía introducir puntos en una gráfica para calcular y procesar la curva de Bézier correspondiente. De esta manera permite observar y analizar rápidamente el comportamiento y las propiedades de dichas curvas al introducir ciertos puntos.

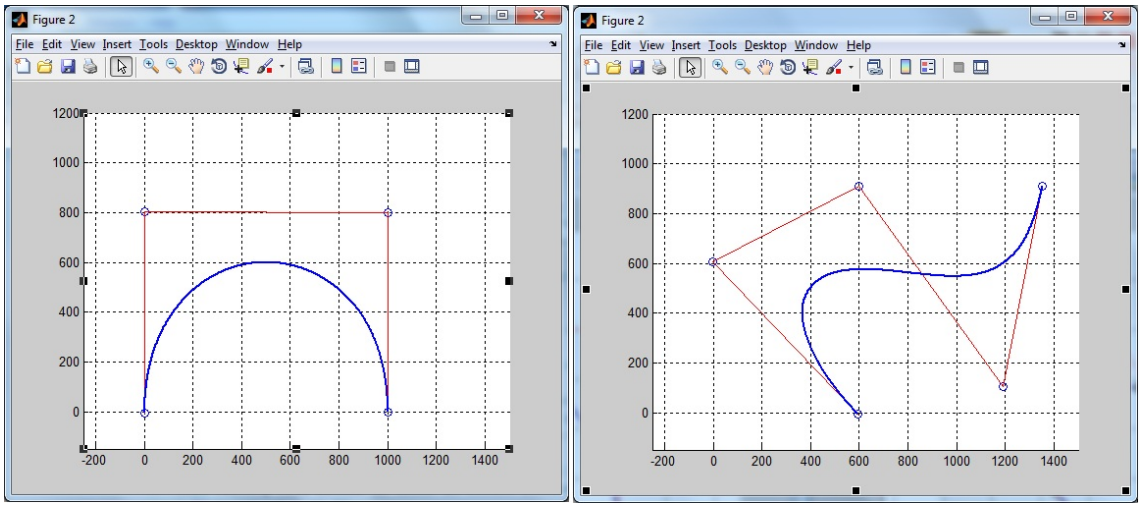


Figura 53: De izqda. A derecha. Gráficas curvas de Bézier de 4 puntos y 5 puntos.

Una vez generadas correctamente las curvas de Bézier tanto para 4 puntos como para 5 puntos, el siguiente paso era calcular y generar las mismas curvas pero a cierta distancia de la curva central, ya que serán estas las que se utilizaran como trayectorias para los robots.

Este proceso ya se ha explicado su implementación en el apartado de coordinación de trayectorias, pero no hemos visto ciertas consideraciones que debíamos tener en cuenta en esta coordinación.

A la hora de trasladar el cálculo de una curva de Bézier como trayectoria de un robot móvil, y más, si tiene que ser para dos robots móviles

que están transportando algún objeto, debemos eliminar por ejemplo curvas en las que se crucen su trazado, sus polígonos de control, o curvas muy cerradas, ya que nos darán problemas más adelante.

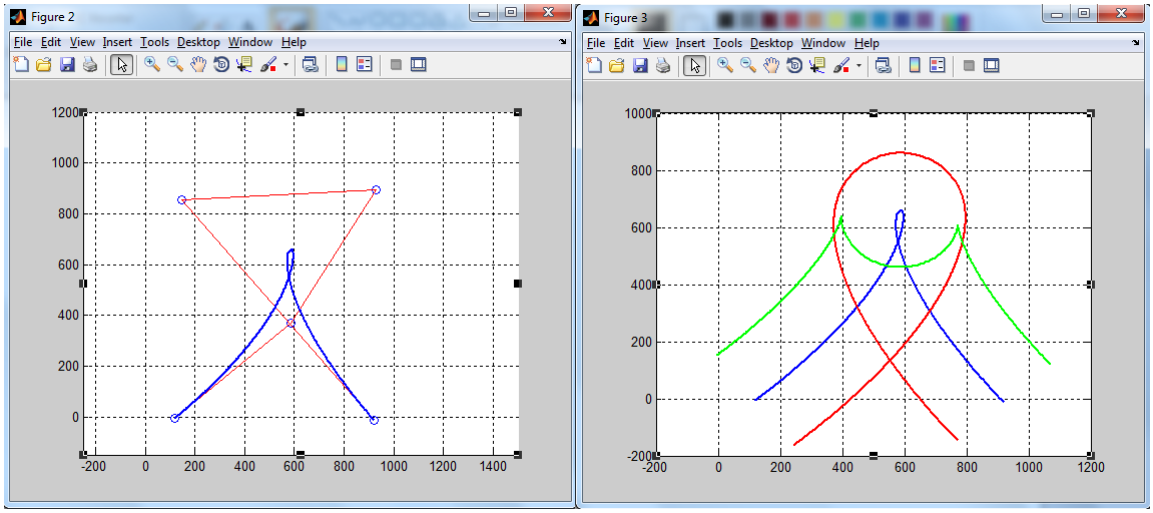


Figura 54: ejemplo de una curva que no se puede utilizar

Como se puede observar en la Figura X, el comportamiento que tendrían los robots no es el deseado en este proyecto, ya que al cruzarse las curvas hace imposible llevar alguna carga entre ambos robots.

Por este motivo, antes de seguir adelante, se hizo una batería de pruebas para calificar y estudiar qué tipo de curvas podríamos utilizar más adelante como trayectorias de nuestros Mindstorms NXT, siempre con curvas de 4 y/o 5 puntos.

Una vez diseñadas las trayectorias que queremos que realicen los robots, pasamos a su implementación en RobotC. El primer aspecto que se presentó para los robots es la forma de introducir los puntos, ya que ahora no tendríamos unos ejes de coordenadas tal y como se había implementado en MATLAB. Para ello la mejor opción era definir variables globales al inicio del programa y mediante un identificador de curvas tener guardadas cada una de las trayectorias que queremos que realicen los robots, a fin de no tener que introducir cada vez los puntos para cada una de las curvas.

Teniendo los puntos de las curvas definidos, se debía implementar el algoritmo de control de trayectorias, que, como se ha dicho en el apartado de la

estrategia del control cinemático, se ha elegido el algoritmo de control de posición por punto descentralizado.

Ya con los robots en mano, las primeras pruebas realizadas se hicieron para observar el comportamiento que ofrecía un solo robot al introducirle una curva. Aparentemente trazaba una trayectoria muy similar a la esperada, pero para poder asegurarnos que el recorrido era el correcto, introducimos en la aplicación una librería con funciones de escritura en fichero, y así de esta manera poder registrar las trayectorias que se calculan y las trazas que realiza realmente el robot, para compararlas entre sí teniendo los datos guardados en un fichero. Estas comparaciones se ploteaban nuevamente con ayuda de MATLAB.

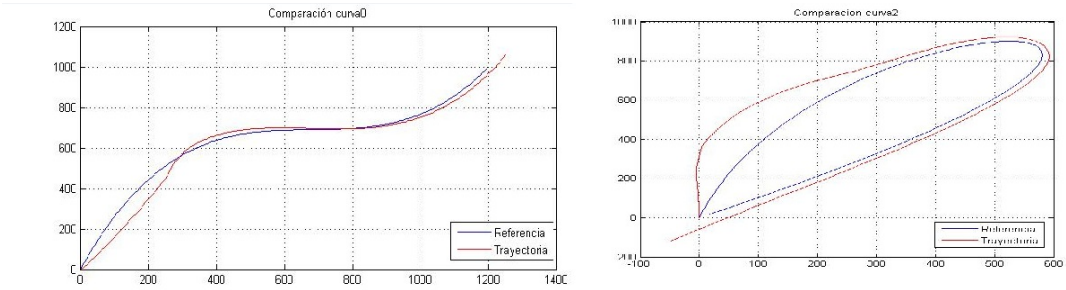


Figura 55: primeras comparaciones entre referencia y trayectoria realizada

En las gráficas anteriores se puede observar claramente que el robot realiza una trayectoria (color rojo) que se desvía notablemente de la curva deseada, pintada de color azul, sobre todo al inicio de las carreras, ya que también hay que decir, que parece que a lo largo del recorrido corrige la traza y se acerca a las referencias calculadas. Estas fueron de las primeras comparaciones que se realizaban, pero se puede deducir que todavía se estaba lejos del objetivo final del proyecto.

Tras las siguientes pruebas realizadas, se observó que el robot empezaba sus carreras realizando movimientos muy bruscos llegando incluso a derrapar las ruedas, por lo que provocaba que perdiera las referencias de posición en la que se encontraba. La primera idea fue reducir la velocidad en la que realizaba las trayectorias, pero de nuevo, incluso con una velocidad mucho menor que la inicial, seguía empezando con movimientos mucho más bruscos

que los que debía hacer. Se hubiera podido dejar este problema para más adelante y empezar a trabajar con la coordinación de varios robots, pero si se solucionaba este problema, funcionaría correctamente con ambos LEGOS, de lo contrario, se produciría derrapes al inicio de la carrera de cada uno de ellos.

¿Cuál era la causa de este problema? ¿Qué es lo que provocaba estos derrapes y este desvío inicial en la trayectoria? Tras varias revisiones a la aplicación y a los algoritmos de control, no se encontraba ningún error en el código programado, pero el robot seguía desviándose.

Finalmente se descubrió cual podía ser una de las causas de estos errores: la posición y orientación inicial del robot. En el algoritmo elegido para el control cinemático, en cada una de las iteraciones el robot compara su posición con las referencias que se van calculando, y en función del resultado se aplican las acciones de control. Es lógico que si el robot inicialmente tiene una posición u orientación distinta a la primera referencia calculada de la trayectoria, aplique una acción de control muy brusca para tratar de llegar lo antes posible a su referencia inicial, consecuentemente esta acción de control provoca los derrapes en las ruedas de los robots. Por lo tanto se calcularon y se modificaron correctamente las posiciones iniciales del robot. En el caso de la orientación inicial, se dibujó la curva en MATLAB y se calculó el ángulo que presentaba el primer segmento de la curva, para así que este sea la orientación inicial del robot. Dependiendo del tipo de curva que se describe, el robot posee una orientación inicial, indicada en radianes.

Tras estas modificaciones, se decide volver a realizar una batería de pruebas para observar y analizar el comportamiento del robot, obteniendo resultados verdaderamente satisfactorios.

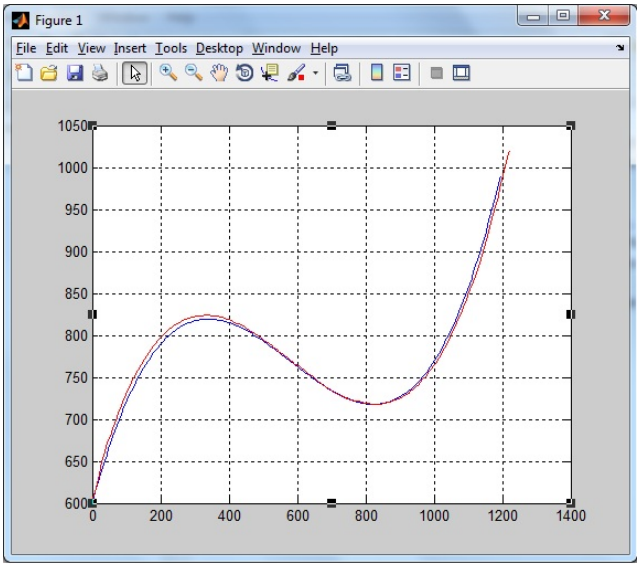


Figura 56: Gráfica con la referencia de 4 puntos y la trayectoria de un robot

En la presente gráfica, con una curva de Bézier de 4 puntos como referencia (color azul), ya se puede apreciar que el robot responde correctamente al algoritmo y realiza correctamente la trayectoria deseada (color rojo).

Con el problema para un robot solucionado, el siguiente paso es coordinar a dos robots para que puedan transportar un objeto, como por ejemplo una viga. Teniendo hecha la implementación matemática en MATLAB, parecía que iba a ser una tarea sencilla, ya que se trataba de trasladar el código de la coordinación de MATLAB a RobotC. Pero a lo largo de la implementación se descubrió que no iba a ser tan trivial como parecía, MATLAB es una herramienta matemática, con infinidad de funciones y de opciones para poder trabajar con vectores o matrices. En cambio, RobotC, a pesar de disponer de algunas de las funciones matemáticas más importantes, encontramos algunas que no tenía, y que además tenían un carácter importante a la hora de la coordinación. Este era el caso de la función atan2.

La función atan2, está definida para que introduciéndole dos parámetros, X e Y, devuelva el arco tangente de los cuatro cuadrantes de la parte real de los elementos introducidos, tal y como se ha descrito en apartados anteriores de la presente memoria. Como RobotC no dispone de dicha función incluso con el nivel de usuario experto, se tuvo que resolver, diseñar e implementar de

manera esta función, ya que sin ella, no se iba a poder calcular la orientación de las curvas paralelas a la curva central.

```
float atan2(float x, float y);


float atan2(float x, float y)
{
  float val;   //val=radianes;

   if (x>0) {val=atan(y/x);}
   else
   if ((x<0)&&(y>=0))  {val=PI+atan(y/x);}
   else
   if ((x<0)&&(y<0))   {val=-PI+atan(y/x);}
   else
   if ((x==0)&&(y>0))  {val=PI/2;}
   else
   if ((x==0)&&(y<0))  {val=-PI/2;}
   else
   if ((x==0)&&(y==0)) {val=0;}

   return val;
}
```

Implementación de la función atan2.

En el programa principal se ha creado una variable de tipo entero para identificar a cada uno de los robots, idRobot, de esta manera, cada robot sabe qué es exactamente que tiene que calcular y qué curvas debe realizar, y así se ahorra un coste computacional bastante elevado al no tener que realizar todas y cada una de las operaciones en la aplicación principal para cada robot.

Con la coordinación aparentemente implementada, es el momento de realizar las pruebas correspondientes con los dos robots al mismo tiempo, y poder analizar el comportamiento que hacen. De nuevo se recurre a MATLAB para poder graficar las trayectorias generadas por nuestros autómatas.

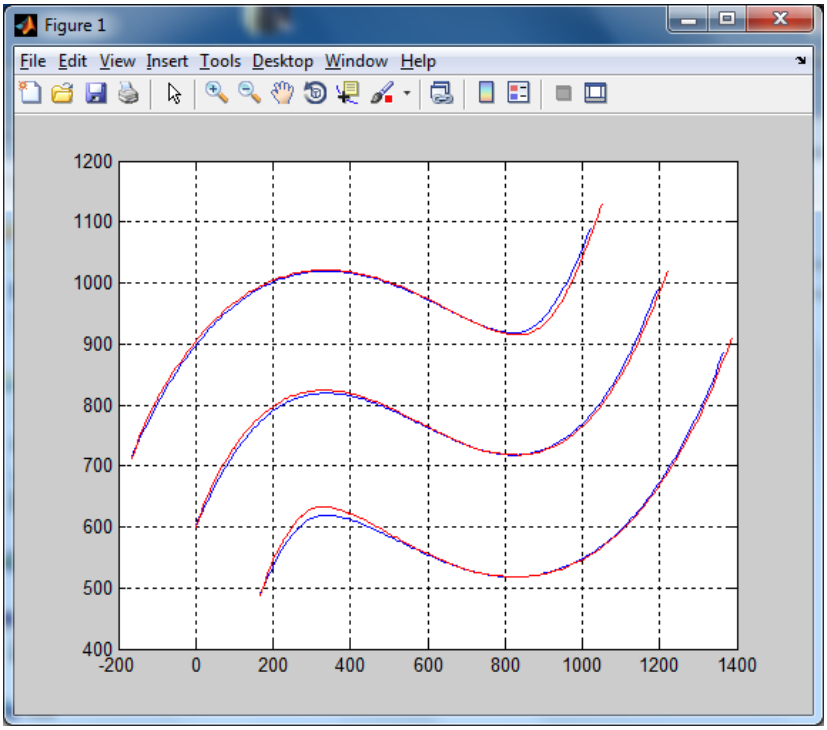


Figura 57: Gráfica con 3 curvas generadas

Ya en esta gráfica se analiza un comportamiento mucho más ajustado al deseado en este proyecto, aunque se puede observar una pequeña desviación en el robot que describe la curva inferior, esto es causado por tener una curva de carácter cerrado y por utilizar un algoritmo de punto descentralizado, pero también hay que destacar lo rápido que se corrige este error, y el robot continua correctamente siguiendo la referencia calculada.

Hasta el momento, todas estas pruebas se han realizado con los robots libres de objetos, es decir, todavía no trasportaban nada. Y además la forma de empezar las carreras era de forma manual. El usuario debía intentar pulsar el botón naranja del ladrillo lo más síncrono posible, para que comenzasen su ejecución al mismo tiempo. Pero como ya se ha explicado, esto es prácticamente imposible para el ser humano. Es entonces cuando se decide introducir el concepto de Bluetooth para realizar la sincronización del inicio de las tareas.

El mecanismo utilizado para sincronizar mediante Bluetooth ya se ha desarrollado en un apartado anterior, por lo tanto, simplemente comentar que antes de introducirlo en la aplicación principal, se investigó y se probó simplemente esperando a que se conectaran los robots y empezaran a

desplazarse en línea recta. Cuando ya comprobamos que funcionaba se introdujo en nuestra aplicación y se consiguió que los robots empezaran a realizar sus curvas de manera síncrona y sin la necesidad de que el usuario pulsara los dos botones a la vez. Ahora se ejecutaba un robot, el cual se queda esperando a que su compañero se ejecute, le envíe el mensaje, y comiencen ambos su trayectoria de manera sincronizada y coordinada.

Por último, cabe destacar que en la práctica fue de vital importancia colocar los robots lo más paralelamente posible entre ellos así como medir perfectamente los puntos iniciales en los que los robots se debían encontrar. De otra forma, aunque los programas funcionaran correctamente, no se conseguirían resultados óptimos en dichas pruebas.

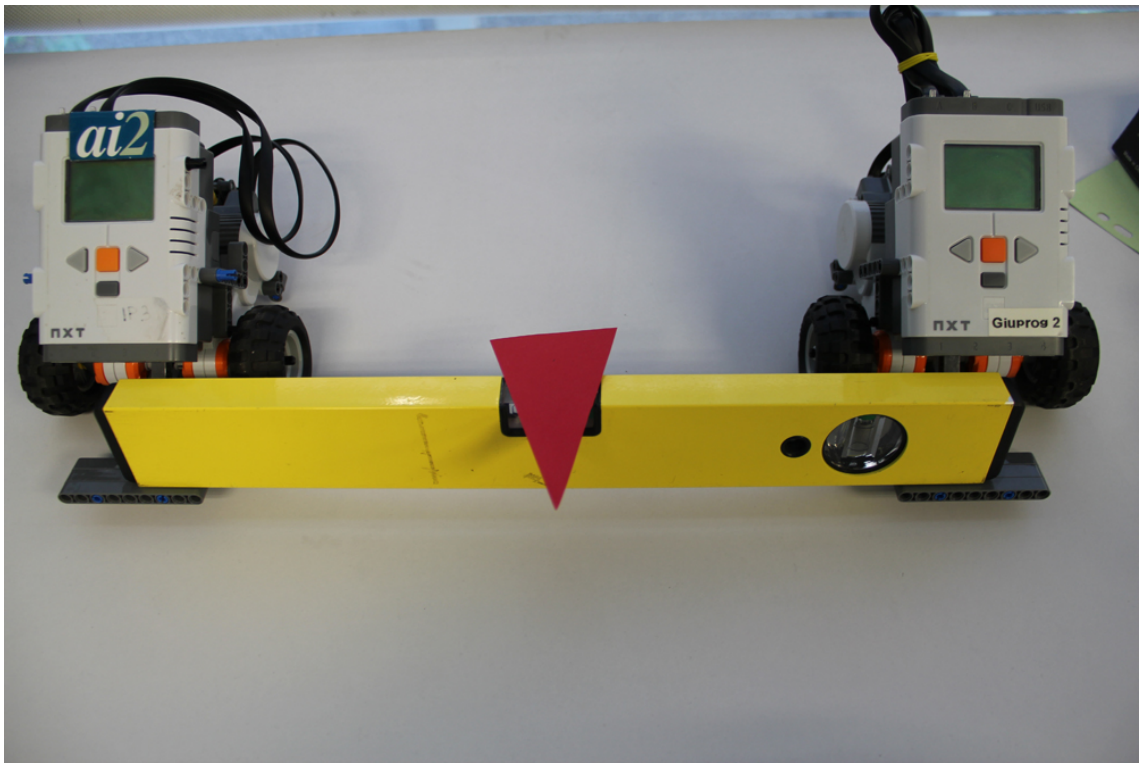


Figura 58: Robots LEGO con el objeto transportado

3.6.1 Resultados obtenidos

Para finalizar el trabajo práctico, se presenta de manera gráfica algunos resultados obtenidos con diferentes tipos de curvas. En las gráficas que se observaran a lo largo de este apartado, primero se presenta la curva obtenida

mediante MATLAB, para comprobar y diseñar la curva que se desea que realicen los robots, y posteriormente se presenta la trayectoria realizada por cada uno de los dos robots.

En los resultados teóricos se pueden apreciar dos gráficas, la primera es la curva que se genera a partir de introducir los puntos, y en la segunda gráfica encontramos tres curvas dibujadas, la curva de color azul se trata de la curva central, que es la trayectoria que debe realizar el punto medio del objeto transportado por los robots. En la parte derecha de la curva azul, encontramos otra curva de color rojo, que en este caso, se trata de la trayectoria que debe realizar el robot de la parte derecha, que en este proyecto se ha nombrado Rob1. Por otra parte, la curva restante, presentada de color verde, es el robot situado a la izquierda llamado Rob2.

Seguidamente tendremos el resultado práctico de cada una de las curva que se habían diseñado de forma teórica. En estas gráficas, encontraremos cuatro curvas, dos curvas por cada una de las trayectorias de los robots. Esto es: En color azul tenemos las referencias de las trayectorias que calcula cada uno de los robots. Estas son las que deben realizar. Y en color rojo tenemos las trayectorias que realmente están trazado los robots. Se puede apreciar que la referencia de la trayectoria calculada y la curva que realiza el robot son coincidentes, con lo que significa que está realizando perfectamente el recorrido deseado.

3.6.1.1 Curvas de Bézier de 4 puntos

Se presentan tres trayectorias distintas para comprobar el comportamiento de los robots móviles:

Curva 0:

Resultado teórico

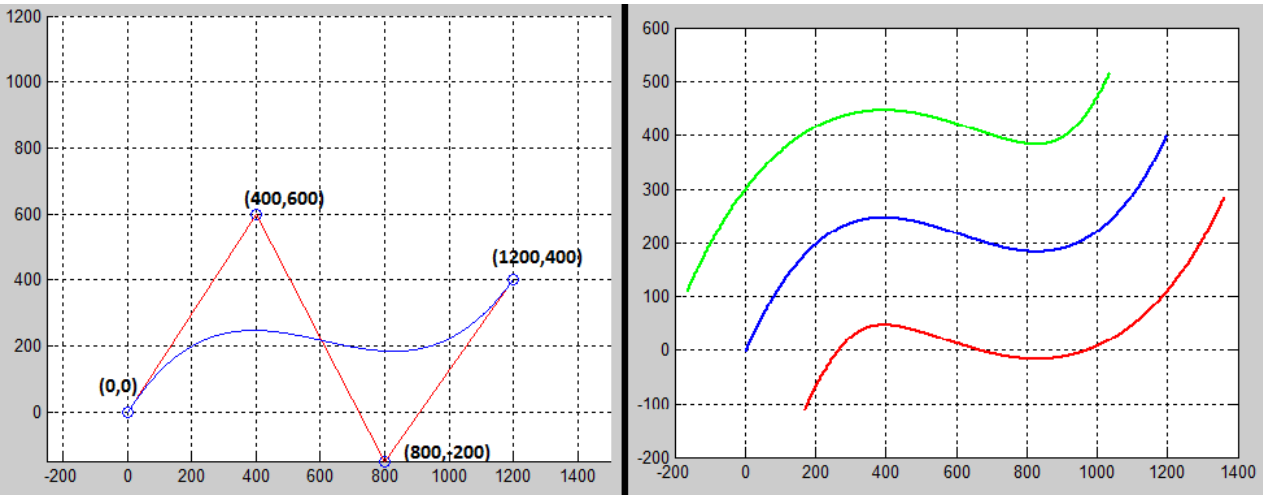


Figura 59: Ejemplo teórico de una curva de 4º grado

Resultado práctico

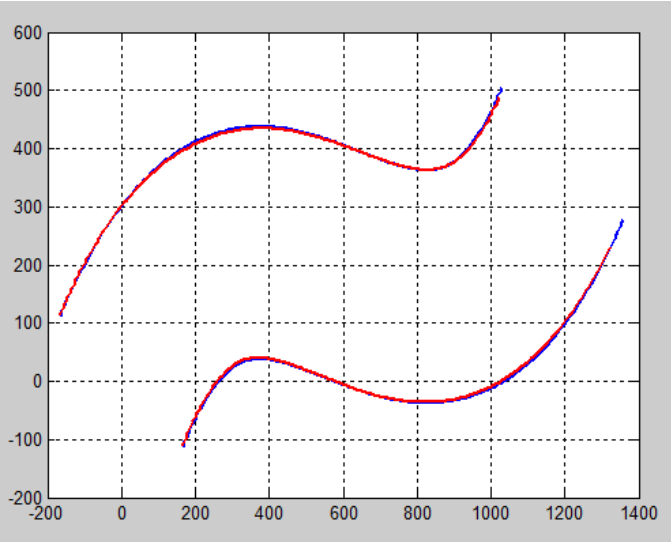


Figura 60: Ejemplo práctico de una curva de 4º grado

Curva1 (línea recta):

Resultado teórico

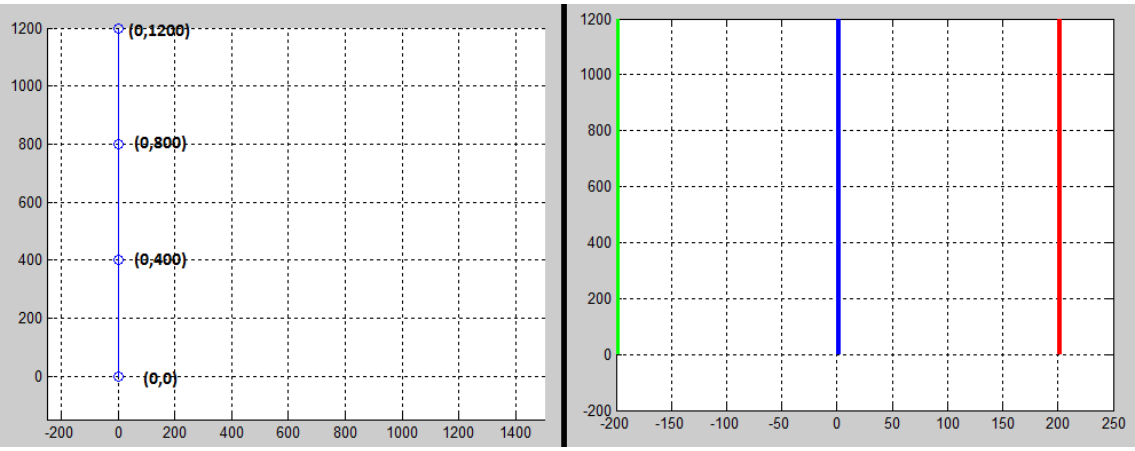


Figura 61: Ejemplo teórico de una curva de 4º grado

Resultados prácticos

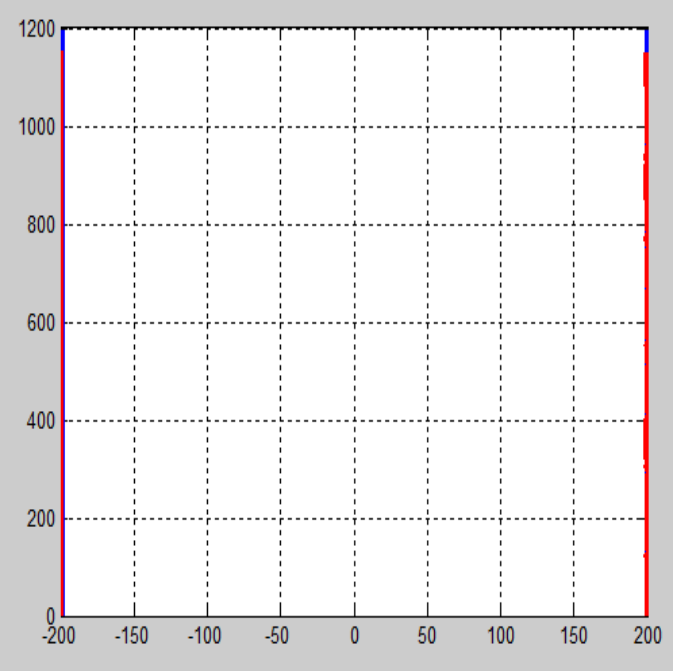


Figura 62: Ejemplo práctico de una curva de 4º grado

Curva2:

Resultado teórico

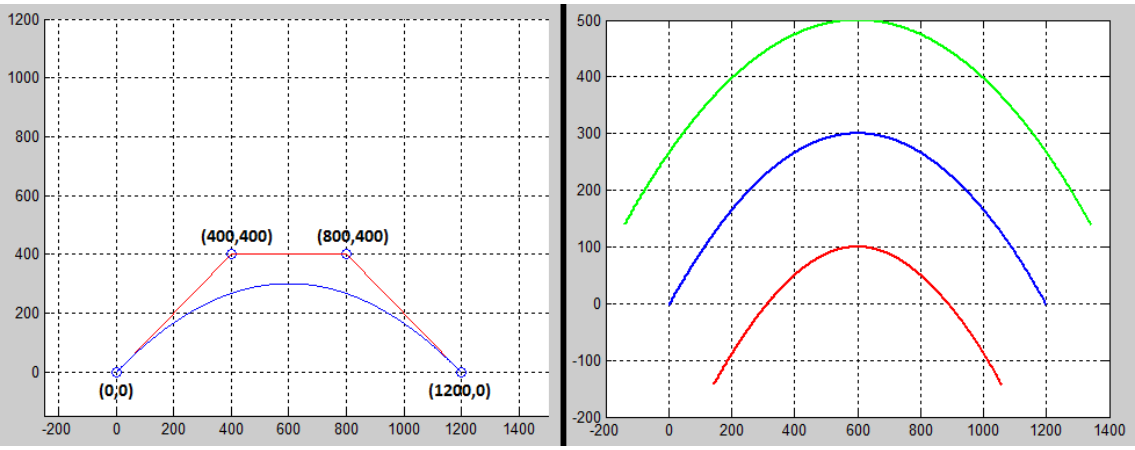


Figura 63: Ejemplo teórico de una curva de 4º grado

Resultado práctico

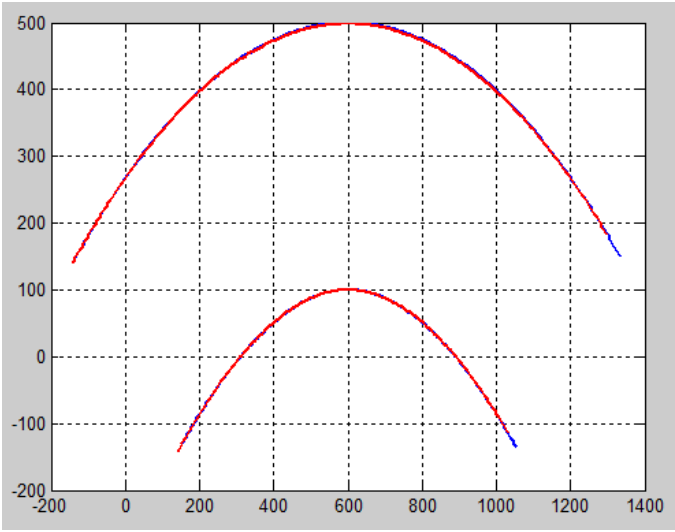


Figura 64: Ejemplo práctico de una curva de 4º grado

3.6.1.2 Curvas de Bézier de 5 puntos

Se presentan dos trayectorias distintas para comprobar el comportamiento de los robots móviles para este grado de curvas:

Curva 0:

Resultado teórico

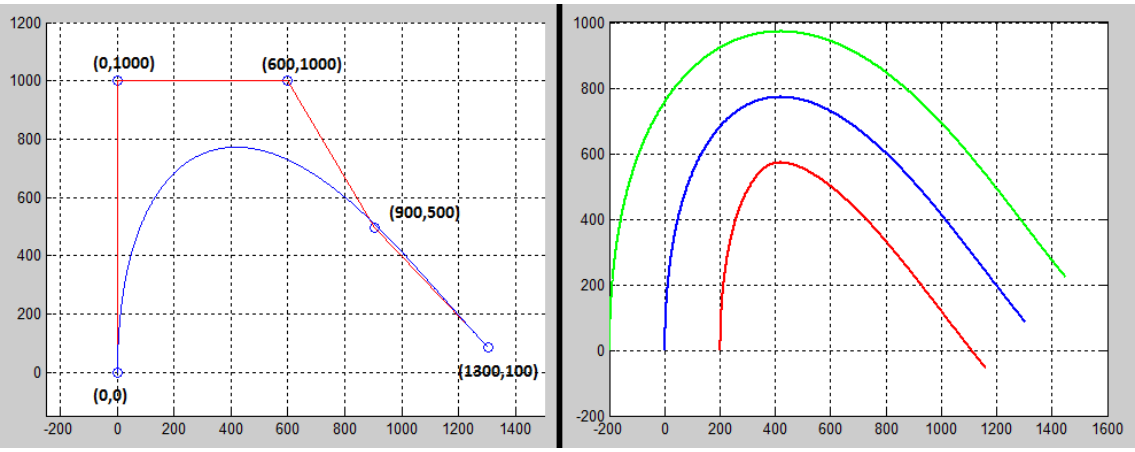


Figura 65: Ejemplo teórico de una curva de 5º grado

Resultado práctico

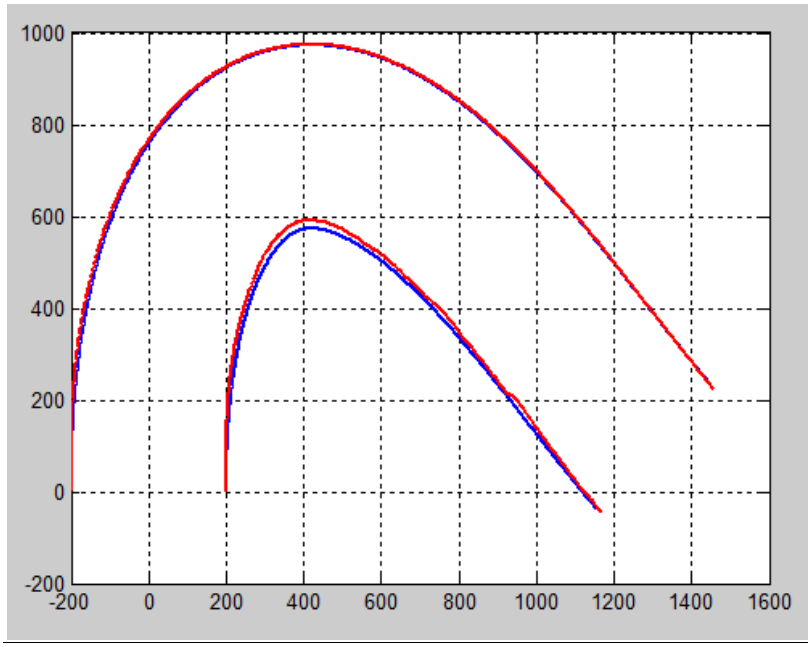


Figura 66: Ejemplo práctico de una curva de 5º grado

Curva 1:

Resultado teórico

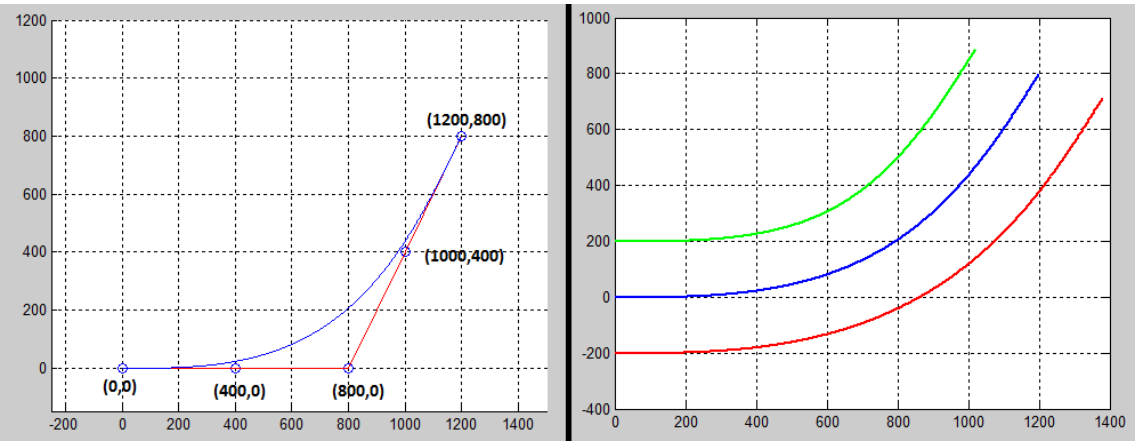


Figura 67: Ejemplo teórico de una curva de 5º grado

Resultado práctico

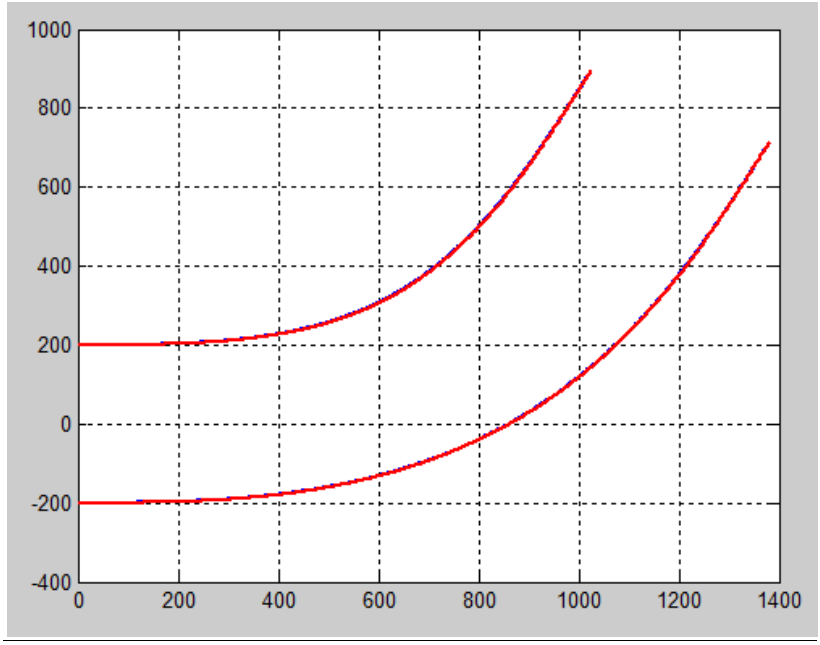


Figura 68: Ejemplo práctico de una curva de 5º grado

3.6.1.3 Resultados mediante cámara cenital

Para poder asegurarnos y comprobar que las trayectorias trazadas por los robots son las deseadas, se han obtenido datos mediante una cámara cenital, la cual detecta un triangulo de color rojo situado justo en el centro del

objeto transportado, y guarda en un fichero de datos los valores de las posiciones x e y, de forma que podemos graficar dichos puntos y así comprobar que la trayectoria realizada es la correcta.

Hay que considerar que los resultados obtenidos han sido satisfactorios, pese a que al dibujar las gráficas se observa que no son curvas con mucha precisión. Esto es debido a la vibración del objeto durante el transporte. Otro dato a tener en cuenta en estas gráficas, es que la orientación de la cámara cenital es distinta a la de los robots, por este motivo algunas de las curvas parece que están desplazadas o ligeramente rotadas, pero el punto principal de este apartado es comprobar que la trayectoria que realiza el móvil transportado por los robots es la curva de Bézier correspondiente a los puntos introducidos.

A continuación se presentan algunos de los resultados obtenidos con estas cámaras:

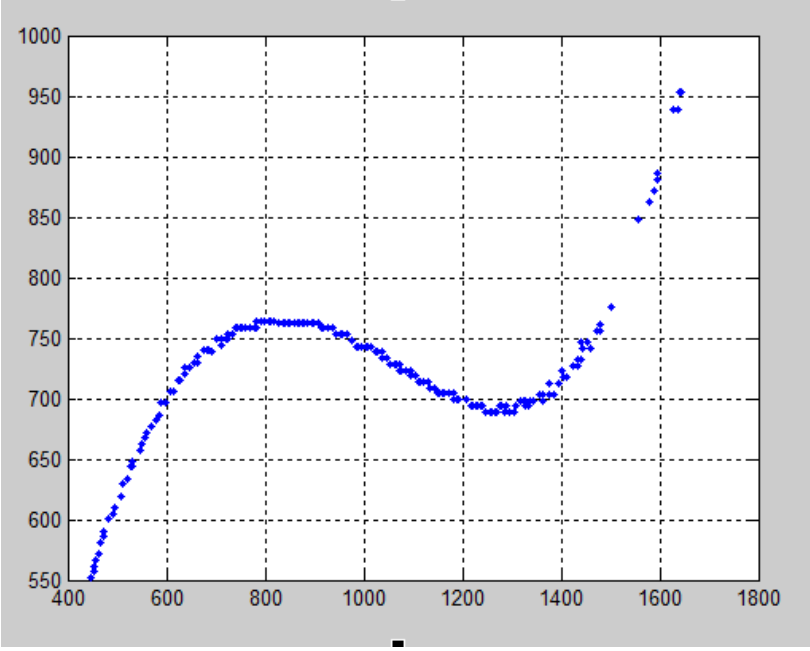


Figura 69: Puntos cámara cenital de la curva 0 de 4 puntos

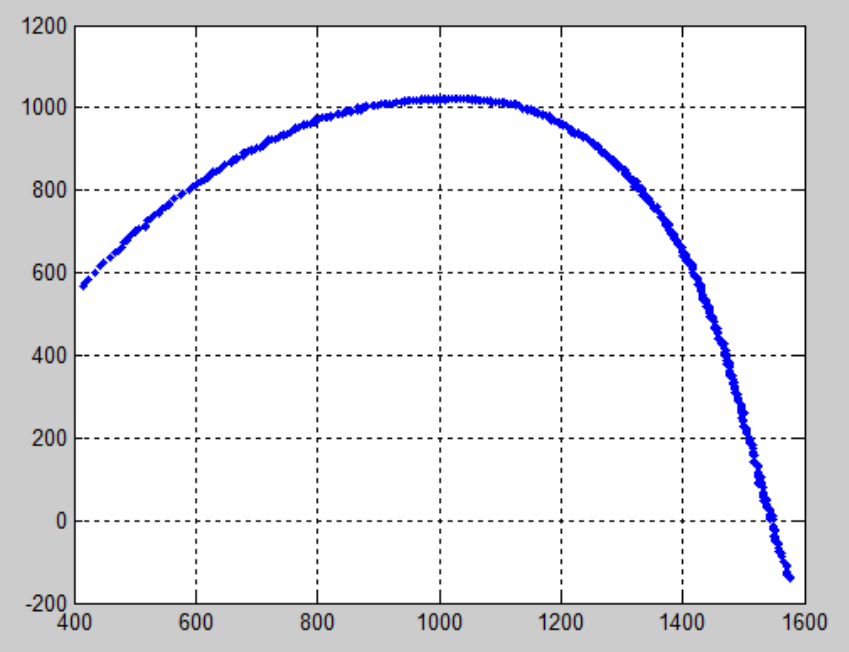


Figura 70: Puntos cámara cenital de la curva 0 de 5 puntos

En la figura 70 podemos observar el aspecto a tener en cuenta comentado anteriormente, la curva descrita es la correcta aunque la orientación de la curva es distinta, por lo que se presenta una curva ligeramente rotada hacia el lado derecho respecto de la curva original (Figura 65).

4.- CONCLUSIONES Y FUTUROS TRABAJOS

Como conclusiones finales de este proyecto de fin de carrera, hay que destacar que se ha cumplido con éxito todos los objetivos propuestos inicialmente. En primer lugar destacar la familiaridad que se ha conseguido con el firmware Mindstorm y con los robots LEGOS NXT. Se ha conocido a fondo las características del entorno de programación RobotC, específica para este tipo de robots. Además de la herramienta matemática MATLAB, con la que ha sido un gran apoyo a la hora de diseñar el tipo de curva que se quería que realizaran los robots, ya que se podía realizar gráficamente y poder observar el comportamiento que debían realizar. También se ha estudiado la gestión de la comunicación mediante Bluetooth que se crea entre ambos robots, para la sincronización del inicio de las trayectorias. Se han estudiado y analizado algoritmos de control cinemáticos para el movimiento de los robots. Se ha trabajado en la gestión y generación de trayectorias basadas en curvas de Bezier de objetos móviles, calibrando tiempos de muestreo o parámetros adecuados.

En definitiva, la tarea de investigar y estudiar los diferentes campos que se proponían se ha visto recompensada y concluida en aplicaciones que cubren con altas expectativas lo estudiado. Los problemas y complicaciones surgidos a lo largo del proyecto, han derivado en soluciones que han motivado la ambición de seguir trabajando e investigando, aún después de haber terminado los objetivos propuestos.

Para futuros trabajos se propone por ejemplo, ampliar la aplicación para que generen las trayectorias de manera dinámica para poder introducir obstáculos a lo largo de la carrera, y que los robots lo puedan detectar, esquivar y seguidamente continuar con la trayectoria definida inicialmente. Esto además de cambiar los algoritmos, requiere la introducción de sensores de ultrasonidos para poder detectar la presencia de objetos a una cierta distancia. También se podría realizar un aplicación para que desde un computador introdujéramos las trayectorias y este se comunique con los robots para que realicen las tareas, de esta manera, se conseguiría dibujar la curva en tiempo

real y además poder ir enlazando el final de una curva con el inicio de una nueva.

5.- BIBLIOGRAFÍA

[1] LEGO Lab, University of Aarhus. http://legolab.daimi.au.dk/

[2] Wikipedia. http://es.wikipedia.org/wiki/Wikipedia:Portada

[3] Proyectos prácticos de electrónica y robótica. Jorge Flores Vergaray.
http://jorgefloresvergaray.blogspot.com/2009/01/sensores-para-robotica.html

[4] Sistemas Robotizados. Universidad de Almería
http://aer.ual.es/docencia_es/sr/

[5] El Rincón del Vago. http://html.rincondelvago.com/historia-de-la-evolucion-de-los-robots.html

[6] Apuntes Diseño por Computador. Curvas de Bézier. E.T.S.I Navales, Universidad Politécnica de Madrid. Leonardo Fernández Jambrina.

[7] ROBOTC.net Blog. http://www.robotc.net/blog/

[8] ROBOTC.net forums. BT with Multiple NXTs.
http://www.robotc.net/forums/viewtopic.php?f=1&t=372&hilit=btConnect

[9] MathWorks España - MATLAB and Simulink for Technical Computing.
http://www.mathworks.es/index.html

[10] MathWorks España. http://www.mathworks.es/videos/matlab/getting-started-with-matlab.html?s_cid=HP_FR_product_mlgetstarted

ANEXO 1: PROPIEDADES DE LAS CURVAS DE BEZIER

Aunque anteriormente hayamos definido la curva en el intervalo [0, 1], es posible utilizar otros intervalos. Sólo hace falta transformar el intervalo por una sencilla aplicación afín para que la parametrización esté definida en el intervalo [a, b]:

$$t(u) = \frac{u-a}{b-a}, \quad u \in [a, b],$$

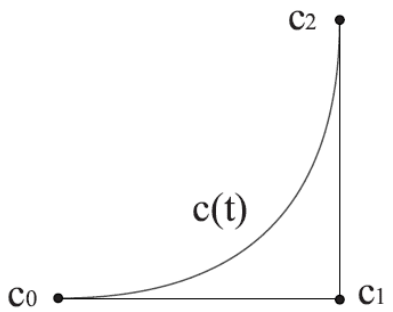


Figura 1: Parábola del polígono de control {c0, c1, c2}

De modo que u = a corresponde a t = 0 y u = b, a t = 1. La nueva parametrización será, pues:

$$\tilde{c}(u) = c(t(u)) := c\left(\frac{u-a}{b-a}\right), \quad u \in [a, b].$$

Nótese que la parametrización c (t) no ve el intervalo [a, b], sino el [0, 1]. Es sólo un ajuste del usuario.

Dos de los vértices del polígono de control tienen una interpretación inmediata. A la vista de la expresión de los polinomios de Bernstein, (1), resulta que Bni (0) = 0, sea cual sea el grado n, salvo para i = 0, para el cual toma el valor Bn0 (0) = 1. Del mismo modo, Bni (1) = 0, salvo para i = n, que no tiene términos (1 − t), por lo cual Bnn (1) = 1.

Así pues, la curva pasa por los vértices c0, cn:

$$c(0) = \sum_{i=0}^{n} c_i B_i^n(0) = c_0, \qquad c(1) = \sum_{i=0}^{n} c_i B_i^n(1) = c_n.$$

De hecho, son los únicos vértices del polígono por los que pasa la curva. Veremos más adelante que el resto de vértices están relacionados con las derivadas sucesivas de la parametrización.

El nombre de polígono de control hace referencia a que sirve para controlar la forma de la curva. Sin embargo, ese control no es local, ya que, desplazando un vértice, se mueve toda la curva, aunque principalmente la

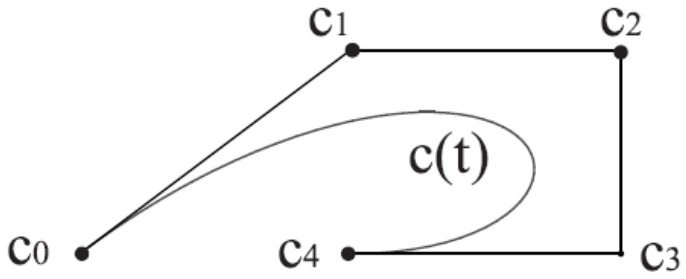


Figura 2: Una curva de Bézier siempre pasa por los vértices primero y último

parte más próxima al vértice en cuestión. Esto se debe a que el máximo del polinomio i-ésimo de Bernstein está en t = i/n,

$$0 = \frac{\mathrm{d}B_i^n(t)}{\mathrm{d}t} = \frac{n!}{i!(n-i)!} t^{i-1}(1-t)^{n-i-1}\{i(1-t) - (n-i)t\} \Rightarrow nt = i$$

con lo cual la variación del vértice i afecta más a los valores de c(t) en las proximidades del máximo

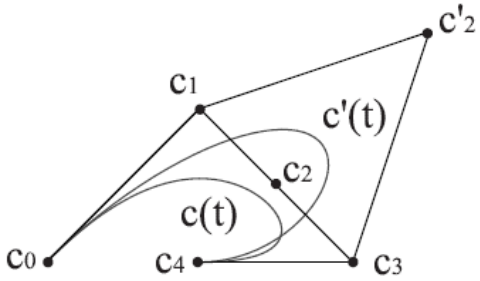


Figura 3: Control local: al mover el vértice c2, se deforma mayormente la parte más próxima a él

Esta propiedad mejorará sustancialmente cuando empleemos curvas polinómicas a trozos.

Una propiedad importante de la representación de Bézier, que es inmediata a partir de la fórmula:

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i}$$

es que la suma de los polinomios de Bernstein de grado n es la unidad. Por tanto, la expresión de la parametrización de la curva:

$$c(t) = \sum_{i=0}^{n} c_i B_i^n(t), \qquad t \in [0,1]$$

es una combinación baricéntrica de los vértices del polígono de control

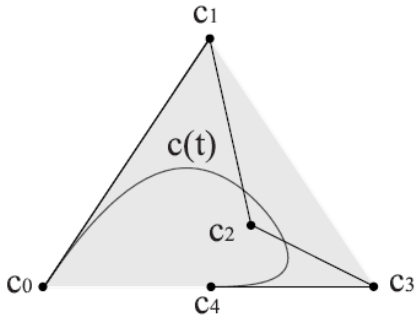


Figura 4: La curva está contenida dentro de la envolvente convexa de su polígono de control

Esta propiedad se traduce en muchas otras importantes. Por ejemplo, la curva está contenida en la envolvente convexa del polígono de control.

Así pues, como su nombre indica, el polígono nos proporciona una primera idea de por donde pasa la curva, lo cual puede ser muy útil, por ejemplo, para

saber si dos curvas se cortan o no: si las envolventes de los polígonos no se cortan, las curvas no se cortan.

Además, el hecho de que la expresión de la curva de Bézier sea una combinación baricéntrica facilita la transformación de la curva por una aplicación afín f, ya que

$$f(c(t)) = f\left(\sum_{i=0}^{n} c_i B_i^n(t)\right) = \sum_{i=0}^{n} f(c_i) B_i^n(t).$$

es decir, la curva imagen tiene por polígono de control la imagen del polígono primitivo, {f(c0), . . . , f(cn)}. Por tanto, no es preciso calcular la imagen de cada punto para construir la curva imagen, sino sólo la imagen del polígono de control y recalcular la curva.

Finalmente, una propiedad sencilla, pero importante, de las curvas de Bézier es su simetría. Si invertimos el polígono de control, {cn, . . . , c0}, la gráfica de la curva es la misma que la correspondiente a {c0, . . . , cn}, sólo que es recorrida en sentido inverso, de cn en t = 0 a c0 en t = 1. El motivo está en la propiedad de simetría de los números combinatorios,

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!} = \binom{n}{n-i}$$

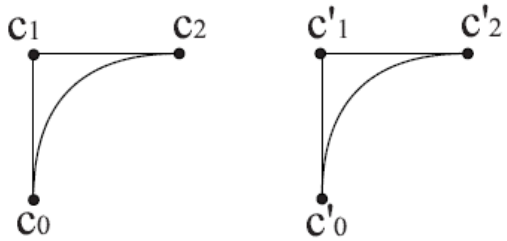


Figura 5: Para trasladar una curva de Bézier, basta trasladar su polígono de control

que implica la relación entre los polinomios de Bernstein, Bni (1−t) = Bnn−i(t), y la simetría de la parametrización de Bézier,

$$c(1-t) = \sum_{i=0}^{n} c_i B_i^n(1-t) = \sum_{i=0}^{n} c_i B_{n-i}^n(t) = \sum_{j=0}^{n} c_{n-j} B_j^n(t),$$

simplemente tomando j = n − i.

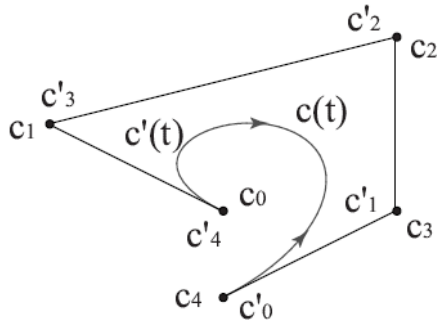


Figura 6: Para invertir el sentido de una curva de Bézier, basta invertir el orden de su polígono de control