DOCUMENTO Nº 1

Memoria descriptiva

Título del proyecto:

Diseño e Implementación de un Sistema de Control para un dispositivo de transporte equilibrado en dos ruedas teledirigido

Autor:

Francisco Javier Antón Díaz

Índice

Índice de ilustraciones

1.1.- Objeto

Este proyecto tiene por objeto ofrecer una solución al problema de una carretilla equilibrada sobre 2 únicas ruedas en configuración de péndulo invertido. Existen ya muchos ejemplos de esta configuración en el mercado. El planteamiento físico de todos ellos es prácticamente el mismo, lo realmente interesante es la obtención de una buena regulación del aparato con diferentes periféricos indispensables.

Ilustración 1 - Segway 1

Ilustración 2 – Segway 2

Ilustración 3 – Segway 3

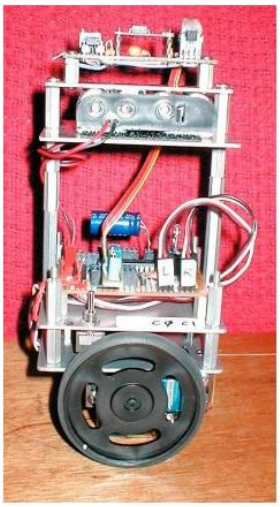


Ilustración 4 –Segway 4

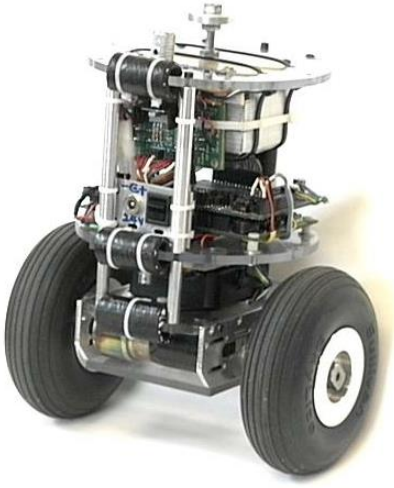


Ilustración 5 – Segway 5

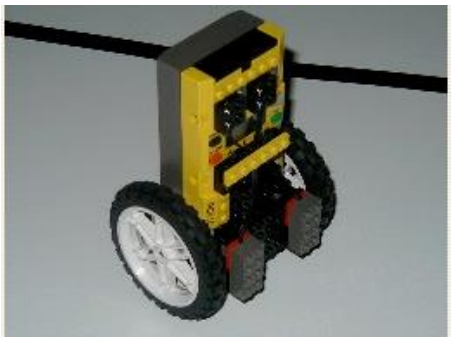
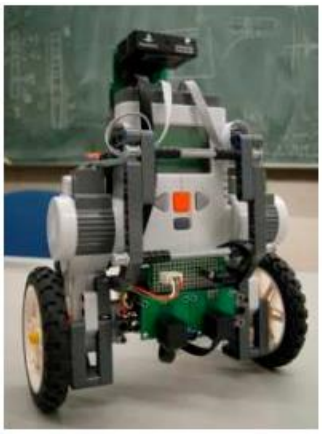


Ilustración 6 – Segway 6



No todas usan la misma tecnología ni las mismas ecuaciones físicas pero lo que sí comparten es la regulación del sistema. El control automático del sistema físico es la base de todos los problemas. El mismo robot con diferente sistema de control automático se comportará de manera radicalmente diferente. Por eso, la elección de la tecnología no será tan rigurosa como el propio cálculo de la regulación del sistema. Es ahí donde se ahondará más profundamente.

El péndulo invertido es conocido por ser uno de los problemas más importantes y clásicos de la teoría de control. El sistema se compone de un carro sobre el cual se monta un péndulo que puede girar libremente. El carro deberá moverse para compensar el desplazamiento del péndulo y mantenerlo así, en equilibrio.

El problema aborda el control de un sistema inestable y no lineal. A menudo es utilizado como ejemplo académico, principalmente por ser un sistema de control accesible, y por otro lado, permite mostrar las principales diferencias del sistema en lazo abierto y de su estabilización en bucle cerrado.

Pese a existir diferentes técnicas a la hora de diseñar el regulador óptimo capaz de estabilizar el péndulo, no todas representan la mejor opción, aunque en muchas de ellas podamos encontrar ventajas que las demás no presentan.

El presente documento describe las instrucciones para la construcción de un robot en configuración de péndulo invertido, así como las estrategias que se pretenden seguir para el diseño e implementación de un controlador que combine diferentes técnicas de control clásicas. Lo más evidente para este modelo de planta es que los sistemas de control aplicables involucran cálculos y toma de decisiones rápidas para mantener los centros de masa de elementos del sistema en la posición y dirección correcta.

El tipo de control que se implementa para cada sistema depende de varios factores, entre ellos, si la planta a controlar es lineal o no lineal, estable o inestable, etc. Hasta hace pocos años el control de sistemas lineales se realizaba principalmente mediante reguladores Proporcional, Integral, Derivativo o una combinación de estos. Para el caso de sistemas no lineales, en especial de varias entradas y salidas, era común utilizar variables de estado. Hoy en día cada vez es más común utilizar controles "inteligentes" para realizar estas tareas.

El objetivo será estudiar la forma de conseguir dejar en perfecto equilibrio nuestro robot. Para conseguirlo se han hecho diferentes pruebas y cálculos con sensores y actuadores.

1.2.- Alcance

El objetivo del proyecto se centra y se limita en adoptar una solución óptima para la regulación de un sistema físico. La construcción de la "maqueta-robot" es parte de este proyecto así como el dimensionado de las variables respecto a este robot pero esto no es el límite del proyecto. Este proyecto se podría rediseñar fácilmente para otro robot o maqueta o para otra máquina. Lo verdaderamente interesante es la regulación y el calibrado de sus sensores para conseguir un equilibrio óptimo y un movimiento estable.

El proceso se ha estructurado en los siguientes apartados:

1. ***Obtención de un modelo teórico:*** Es la descripción mediante ecuaciones de la dinámica del sistema *péndulo invertido* que se propone realizar, la modelización del sistema real.

2. ***Simulación del sistema:*** A partir de las ecuaciones obtenidas para la descripción del sistema, se realizará un ensayo para ver su comportamiento y se obtendrá el sistema de control que permitirá al robot mantener el equilibrio.

3. ***Implementación en el sistema real:*** Comprende la construcción física del modelo y la utilización del control obtenido en la simulación del sistema para actuar sobre el modelo real.

Los tres apartados se realizarán de forma secuencial, aunque entre la simulación y la implementación existirá una estrecha relación bidireccional, puesto que ciertos parámetros de la simulación se obtendrán mediante experimentos con el sistema real, con la finalidad de aumentar la similitud en cuanto a resultados entre ambos apartados.

El resultado será el desarrollo del proceso completo de modelización, simulación e implementación de un sistema de control automático que controle adecuadamente la oscilación de un péndulo sobre una base móvil. Para mostrar el proyecto vamos a utilizar una maqueta dimensionada para realizar las labores con total exactitud. Para realizar la maqueta necesitaremos un sistema digital que sea programable con puertos E/S, varios sensores, varios actuadores, piezas de ensamblaje y cableado adecuado.

1.3.- Estudio de la realidad

Sin duda, un precedente en este proyecto sería la carretilla elevadora. Su funcionamiento es similar, tiene muy en cuenta el centro de gravedad y la carga máxima según altura.

Ilustración 7 - Carretilla elevadora

La carretilla elevadora es una máquina de uso generalizado en el sector industrial. Permite la manipulación mecanizada de las cargas, permitiendo su elevación en altura y su transporte. Dado que es una máquina móvil, de cierto volumen que puede

transportar cargas muy pesadas entraña una serie de riesgos para la seguridad y salud de los trabajadores. Es necesario, por tanto, el establecimiento de ciertas medidas preventivas que ayuden a minimizar los riesgos que su uso genera. La carretilla elevadora es de tracción motorizada. Puede ser eléctrica o con motor de combustión interna.

El funcionamiento teórico y la física del sistema se explica a continuación:

- **Principios de la compensación**

Un vehículo de elevación (balancines o cualquier equipo compensado) tiene un punto de equilibrio o eje fijo. El eje fijo del eje longitudinal (hacia delante y hacia atrás) de la máquina es la línea central de los neumáticos delanteros.

- **Los principios de la compensación se pueden explicar como sigue.**

El peso de la parte del vehículo situada hacia atrás de la línea central del neumático delantero (ilustración 7) representa el peso de un niño situado a la distancia ”D” del eje fijo (ilustración 8). El peso del mástil, el carro, las horquillas u otro medio de activación de cargas, así como la propia carga (ilustración 7) representan el peso de los niños (ilustración 8) situados a la distancia ”d” del eje fijo.

En el balancín, los niños se mueven dentro o fuera del eje fijo, situando su peso a las distancias ”d” o ”D” hasta que están equilibrados. El equilibrio del balancín se puede ilustrar afirmando que el peso ”p” multiplicado por la distancia ”d” equivale al peso ”P” multiplicado por la distancia ”D”. Esto indica que varias combinaciones de peso ”multiplicado por la distancia” pueden crear una condición de equilibrio.

Los principios de la estabilidad en una carretilla elevadora son similares a los de un balancín en el sentido de que el peso de la carga y su distancia del eje fijo determinan los requisitos de compensación.

Cabe recordar que la carretilla elevadora **es diferente** porque el peso hacia atrás de la línea central del eje de accionamiento (eje fijo) multiplicado por la distancia al ”CG” (centro de gravedad) de dicho peso debe ser siempre superior por un amplio margen al peso hacia delante del eje de accionamiento multiplicado por la distancia a su ”CG”. Si nos aproximamos a una condición de equilibrio, las fuerzas dinámicas necesarias para detener, conducir o inclinar pueden provocar un vuelco peligroso del vehículo.

El peso de la carretilla elevadora situado hacia atrás del eje fijo y el ”CG” de ese peso no varían, por lo que la compensación siempre será un valor fijo.

El peso del mástil, el carro y las horquillas o fijación también es un valor fijo; pero la distancia al ”CG” hacia delante del eje fijo varía en función de la inclinación del mástil y la altura del carro. El peso de la carga varía a medida que cambia su posición hacia delante del eje fijo. Su posición depende de si la carga está situada en las horquillas (o u otro medio de sujeción), del ángulo de inclinación del mástil y de la altura del carro y la carga.

El operador debe controlar estas variables y asegurarse de que el peso total hacia delante del eje fijo multiplicado por la ubicación del centro de gravedad más las otras fuerzas dinámicas de funcionamiento nunca superen la compensación de la carretilla elevadora. El cumplimiento de estas normas de seguridad y el uso de un buen sentido común ayudarán a garantizar el funcionamiento seguro de la carretilla elevadora.

Las carretillas elevadoras están diseñadas para mantener el equilibrio adecuado de todos los elementos. El equilibrio se puede ver afectado por los siguientes factores:

- la capacidad de la máquina (en un centro de carga especificado)
- el peso de la carga y su centro de gravedad
- la posición de la carga en las horquillas o la fijación
- el tipo y el peso de la fijación
- la aceleración o el frenado
- las condiciones de la superficie del terreno y los grados de ángulo
- la inclinación del mástil y la altura de la carga
- las condiciones climáticas

Ilustración 8 - Funcionamiento carretilla 1

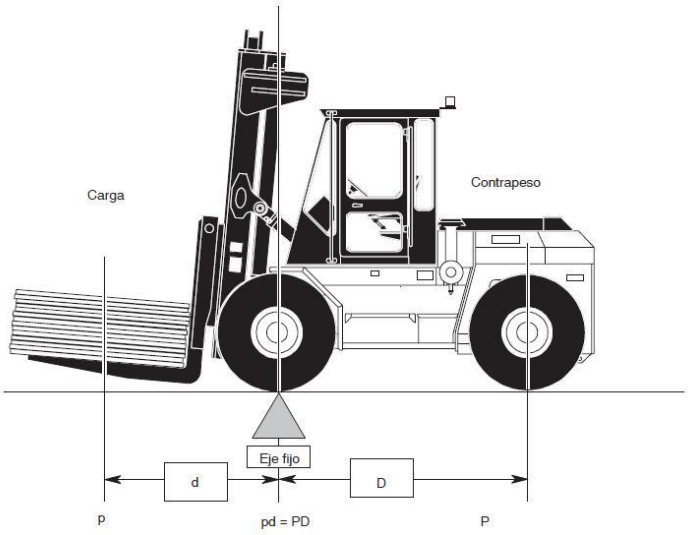
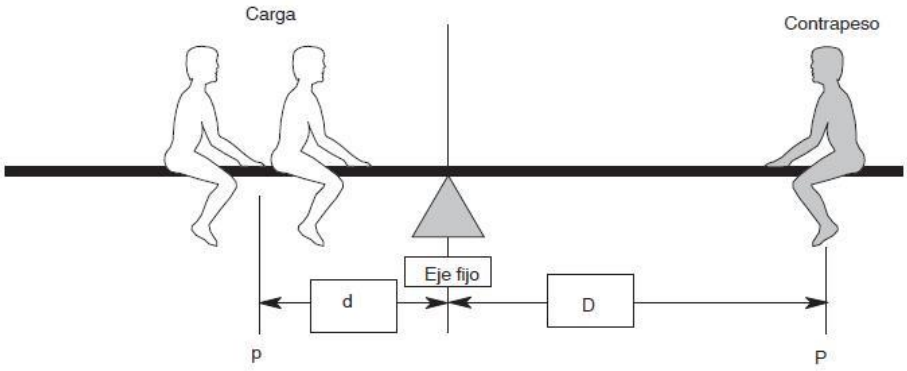


Ilustración 9 - Funcionamiento carretilla 2



En este proyecto se ha ido un paso más allá. Se ha introducido un nuevo elemento que afecta notablemente al equilibrio. El montaje se hará en un vehículo de 2 ruedas por lo que además de todos los factores anteriores se deberá tener en cuenta el cambio en el centro de gravedad debido al peso del propio aparato. Este en reposo tenderá a caer por su propio peso.

En el mercado hay múltiples opciones a la hora de elegir una carretilla pero todas van sobre vehículos de 3 ó 4 ruedas. Por otra parte, sobre vehículos de dos ruedas en equilibrio solo se encuentran "juguetes" robotizados. A modo de maqueta para presentar el proyecto nos valdría cualquiera de ellos con el objeto de explicar su funcionamiento para poder, en un futuro, construir la carretilla elevadora equilibrada en dos únicas ruedas.

1.4.- Soluciones posibles

Vamos a indagar en otros dispositivos de similares características para ver el estado de la industria. Para la configuración de equilibrio en dos ruedas hay algunos robots que tienen el mismo principio de funcionamiento. Un detalle a tener en cuenta es

que en nuestro dispositivo el peso de la estructura está por encima de los ejes de las ruedas. En algunos segway esto no es así por lo que el sistema es diferente. Es más fácil equilibrar un mecanismo que tiene el peso por debajo del eje de las ruedas.

Ilustración 10 – Principio funcionamiento segway

1.4.1.- Estado del arte

1.4.1.1.- Robot equilibrio 1

Se trata de un robot de tipo balancín que debe mantener la vertical (péndulo invertido). El microcontrolador de abordo calcula de forma dinámica los valores de velocidad que debe entregar a los motores dependiendo de las lecturas que va obteniendo de los sensores.

El microcontrolador utilizado es un 18F4550 de microchip. Los sensores son un acelerómetro de tres ejes (se usan dos ejes) MMA7260QT y un giroscopio MLX90609.

El sistema de motricidad es un kit RD01 en el que venían dos ruedas, dos motores con reductora, dos soportes y una placa con el puente en H.

Como metas del proyecto se encuentran la consecución de dos aspectos. El primero, que fuera capaz de mantener el equilibrio incluso en superficies inclinadas. El segundo aspecto era hacer posible que el robot pudiera desplazarse en cualquier dirección, para ello se le instaló un módulo Bluetooth® con el que poder comunicarse usando un teléfono móvil en el que corre un programa escrito en Java. A través de este programa se le puede indicar al robot la velocidad de desplazamiento y el sentido de giro.

Aspectos técnicos:

Para conseguir que el robot se mantenga estable en la vertical es necesario que los motores desarrollen la velocidad justa en cada situación. Así pues el aspecto que toma más importancia es el cálculo de los ángulos de inclinación del robot.

Para calcular dichos ángulos se emplean dos sensores, un giroscopio y un acelerómetro. El giroscopio nos va a medir velocidad angular o velocidad de rotación sobre el eje del sensor, en otras palabras, el espacio recorrido (grados de rotación) en un tiempo determinado. El acelerómetro, como su nombre indica mide aceleración en uno, dos o tres ejes, al ser la fuerza de gravedad de la tierra un tipo de aceleración podemos medir la inclinación del sensor con respecto a la perpendicular del suelo.

Este robot recibe la señal de los dos sensores, el giroscopio y el acelerómetro. El acelerómetro es sensible a las aceleraciones por lo que este sensor montado en un vehículo que se desplaza no va a ser todo lo preciso que necesitamos. Solamente nos dará medidas "buenas" cuando el conjunto del robot no experimente aceleraciones o estas sean muy pequeñas. Por otra parte hay que remarcar que en condiciones de bajas aceleraciones (o nulas) el acelerómetro es muy fiable.

Así pues, ¿cómo medimos el ángulo de inclinación? Básicamente hay que usar el giroscopio. Al contrario que el acelerómetro, este sensor es totalmente inmune a la aceleración y se ve poco afectado por las vibraciones. Sin embargo tiene un problema, la deriva que acumula con el tiempo. Efectivamente, a medida que pasa el tiempo, y a una velocidad angular constante, su lectura tiene cierta variación lo que provoca que el ángulo calculado difiera del real a medida que pasa el tiempo.

De esta forma nos vemos obligados a utilizar los dos sensores. Calcularemos el ángulo con las medidas que obtenemos del giroscopio e iremos "refrescando" este valor con lo obtenido del acelerómetro en los momentos de mínima aceleración.

Partimos de una máquina que es capaz de estabilizarse y moverse con solo dos ruedas con la ayuda de una fuerza externa. La máquina por sí sola evidentemente no será capaz de permanecer en equilibrio durante más de un instante. Eso es algo que debemos conseguir recibiendo información de algún sensor y excitando algún actuador.

Ilustración 11 - Robot equilibrio 1

1.4.1.2.- Robot equilibrio 2

El proyecto consiste en la fabricación de un robot de dos ruedas que consiga mantener el equilibrio por sí mismo con ayuda de un acelerómetro y un giróscopo, además de un filtrado digital de la señal y un control PID y que pueda ser controlado por Bluetooth® desde un smartphone.

Los motores utilizados para este proyecto serán dos servos que habrá que modificar para conseguir que puedan girar 360º.
Para controlar los motores se utiliza un puente en H. El puente se basara en un integrado L298 que soporta 2 amperios de trabajo continuo y picos de hasta 3, y una tensión de hasta 46 voltios.

Para la parte de programación lo primero que se hace es implementar un filtrado digital de la señal del acelerómetro y el giróscopo. El sensor que utiliza es el GY-521, basado en el chip MPU-6050 que incluye acelerómetro y giróscopo en el mismo integrado.

El filtrado digital consiste en un filtro de Kalman que hace mas estable las medidas y evitara medidas erróneas puntuales. El filtro de Kalman trabaja en dos etapas, en la primera realiza una estimación y en la segunda etapa corrige el error. En este caso para corregir la deriva del giróscopo se utiliza el acelerómetro o a la inversa.

Además del filtro de Kalman también es necesario implementar un control PID para el control de los motores. El algoritmo de cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral, y el derivativo.

Ilustración 12 - Robot equilibrio 2

1.4.1.3.- Robot equilibrio 3

El robot MiP de WowWee mantiene el equilibrio sobre dos ruedas**.** El fabricante de juguetes WowWee en asociación con UCSD (Universidad de California en San Diego), han desarrollado el PMI, un robot móvil con forma de Péndulo Invertido, es una máquina que está controlada por gestos con las manos o mediante conectividad Bluetooth® desde cualquier teléfono inteligente o tableta. MiP tiene su propia personalidad a la hora de la interacción con los seres humanos.

El robot viene con accesorios que se pueden montar o desmontar en cualquier momento. El dispositivo robótico tiene la capacidad de equilibrar varios objetos y también interactuar con otro robot MiP mientras se mueve.

Ilustración 13 - Robot equilibrio 3

Lo primero que debemos elegir es el microcontrolador que se encargará de ser el cerebro de operaciones. Se estudian diferentes soluciones que podrían ser válidas para el efecto:

1.4.2.- Sistemas digitales

1.4.2.1.- Basado en PLC

Ilustración 14 - PLC

Los PLC´s modelo S7 conquistan cada vez más campos de aplicación, puesto que es muy potente, su precio es sumamente atractivo y es muy fácil de usar. Este sistema necesita el PLC, un ordenador, cableado y un montaje mecánico de la carretilla en péndulo invertido. El PLC estaría disponible en la universidad para poder usarlo en caso necesario. El ordenador no sería problema puesto que hay muchas existencias en la propia universidad. El problema puede venir a la hora de adecuar todo el montaje hardware del robot en péndulo invertido. Es ahí donde se perdería mucho tiempo y dinero en diseñar y montar algo que no está relacionado con este proyecto. Es algo puramente mecánico, no de un proyecto electrónico.

1.4.2.2.- Robot programable

Ilustración 15 – Robot programable

Existen en el mercado muchos modelos de robot programable, muchas marcas y tiendas a las que acudir. El precio no es tan alto como el del PLC pero el manejo del software no es tan sencillo ni tan standard. Aquí el problema también puede venir a la hora de encontrar los sensores, interconectarlos y usarlos. Realmente el proyecto se centraría en hacer uso y manejo del robot.

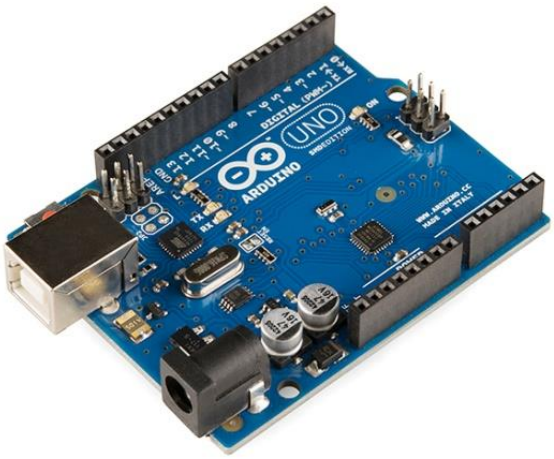
1.4.2.3.- LEGO Mindstorms nxt 2.0

Ilustración 16 – Kit Lego

Lego ofrece una gama amplia de accesorios para la construcción de casi cualquier robot por poco dinero. Además el montaje y el manejo de los sensores es muy sencillo ya que todo se hace en un lenguaje muy coloquial y la compatibilidad del equipo es inmediata. También tiene una ventaja muy atractiva: El “cerebro” de todo es un microcontrolador de la misma marca que hace que el conexionado entre todo sea compatible y muy sencillo.

1.4.2.4.- Basado en microcontroladores. Arduino

Ilustración 17 - Arduino



Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el Atmega168, Atmega328, Atmega1280, y Atmega8 por su sencillez y bajo coste que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa.

Desde octubre de 2012, Arduino se usa también con microcontroladoras CortexM3 de ARM de 32 bits,5 que coexistirán con las más limitadas, pero también económicas AVR de 8 bits. ARM y AVR no son plataformas compatibles a nivel binario, pero se pueden programar con el mismo IDE de Arduino y hacerse programas

que compilen sin cambios en las dos plataformas. Eso sí, las microcontroladoras CortexM3 usan 3,3V, a diferencia de la mayoría de las placas con AVR, que generalmente usan 5V. Sin embargo, ya anteriormente se lanzaron placas Arduino con Atmel AVR a 3,3V como la Arduino Fio y existen compatibles de Arduino Nano y Pro como Meduino en que se puede conmutar el voltaje.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data. Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente. Puede tomar información del entorno a través de sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador. También cuenta con su propio software que se puede descargar de su página oficial que ya incluye los drivers de todas las tarjetas disponibles lo que hace más fácil la carga de códigos desde el computador El proyecto Arduino recibió una mención honorífica en la categoría de Comunidades Digital en el Prix Ars Electrónica de 2006.

1.4.2.5.- Basado en ordenadores. Raspberry Pi+

Ilustración 18 - Raspberry pi+

Raspberry Pi es un ordenador de placa reducida o (placa única) (*SBC*) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

El diseño incluye un System-on-a-chip Broadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (el firmware incluye unos modos "Turbo" para que el usuario pueda hacerle overclock de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM (aunque originalmente al ser lanzado eran 256 MB). El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa.

La fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada deDebian), RISC OS 5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora) y promueve principalmente el aprendizaje

del lenguaje de programación Python. Otros lenguajes también soportados son Tiny BASIC, C, Perl y Ruby.

Además necesitaremos piezas de ensamblaje, sensores y actuadores. En el mercado también existen muchas marcas que fabrican estos elementos.

1.4.2.6.- Sensores y actuadores

En el mercado existen una infinidad de sensores y actuadores válidos para realizar el proyecto.

Ilustración 19.- Servomotor

Ilustración 20.- Giroscopio

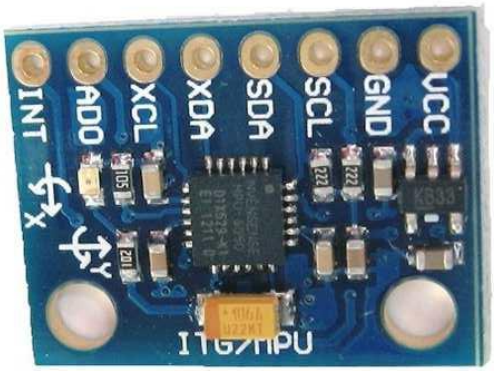


Ilustración 21.- Sensor de posición

Ilustración 22.- Brujula

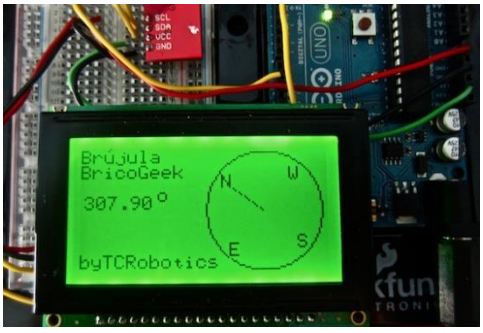


Ilustración 23.- Motores

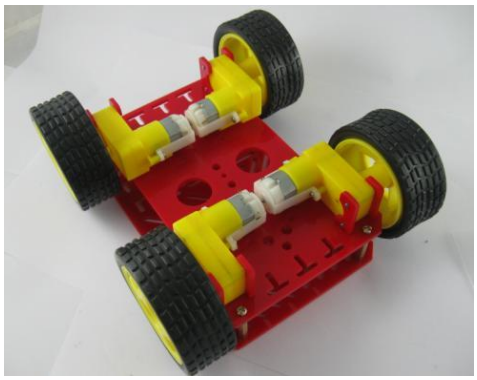
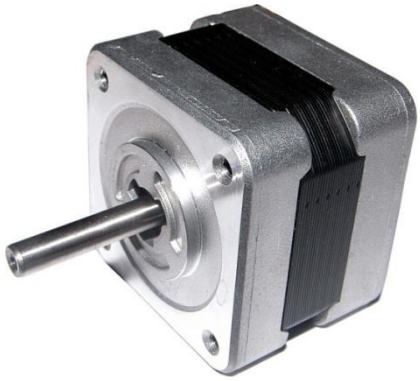


Ilustración 24.- Motor paso a paso



1.5.- Solución adoptada

Las diferentes opciones disponibles dan lugar a una tabla comparativa que ilustra claramente la conveniencia de cada una de ellas. Para poder llegar a una buena elección se han puntuado las distintas características del 1 al 10. El 1 es la puntuación más "mala" y 10 será la más "buena". Así lograremos saber qué microcontrolador es el más conveniente.

	Precio	Integración	Peso	Disponibilidad	Resultado

PLC	1	7	2	8	**18**
Robot	3	3	3	5	**14**
Lego	8	10	9	10	**37**
Arduino	10	5	9	10	**34**
Raspberry pi	10	5	9	10	**34**

Sin duda elegimos el entorno de LEGO por su sencillez, robustez y precio. Dentro de la gama de productos LEGO hay un paquete robótico llamado mindstorms NXT 2.0 sobre el cual trabajaremos para conseguir nuestro objetivo.

Quizá el hecho que más pueda sorprender en un primer momento es la elección de una plataforma como Lego Mindstorms NXT 2.0, puesto que existe una gran diversidad de implementaciones físicas para realizar un montaje con el que llevar a cabo el objetivo de controlar un sistema carro-péndulo. Por ello, el primer paso realizado será conocer esta plataforma para poder justificar su elección.

A pesar de que inicialmente se propuso como un juego de robótica, la plataforma Lego (***http://www.lego.com/en-gb/mindstorms/?domainredir=mindstorms.lego.com***) ha ido adquiriendo un interés creciente en entornos educativos a diferentes niveles. Por ejemplo, Butler y Martin (2001) analizan esta plataforma para impulsar el desarrollo de habilidades tecnológicas entre los alumnos y profesores de educación primaria de Irlanda usando técnicas de aprendizaje basado en proyectos. Por su parte, Atmatzidou y otros (2008) investigan la efectividad de usar LEGO Mindstorms como herramienta para introducir a los alumnos de primaria y secundaria los conceptos básicos de programación a través de actividades lúdicas. También en enseñanza secundaria,

Moundridou y Kalinoglou (2008), describen un estudio empírico relativo al uso de la plataforma Lego Mindstorms en enseñanza secundaria en el campo de ingeniería mecánica. Sin embargo, es en entornos universitarios donde esta plataforma está ganando más aceptación debido a que proporciona una plataforma potente y de coste reducido para ilustrar conocimientos de diferentes áreas de conocimiento. Sólo por mencionar algunos autores, Gómez de Gabriel y otros (2010), describen cómo utilizan la plataforma LEGO para desarrollar competencias en mecatrónica en la enseñanza universitaria de grado. Otros autores la usan para desarrollar competencias en programación en diferentes lenguajes de programación, tanto a nivel básico (Gandy, E.A, 2010) como avanzado (Lew y otros,2010).

El kit LEGO MINDSTORMS NXT (http://mindstorms.lego.com/) es una buenísima elección para ser utilizado como plataforma de bajo coste que permite adquirir las competencias relativas a informática industrial dentro de los estudios de Ingeniería Electrónica.

La plataforma LEGO MINDSTORMS es un juego de robótica desarrollado por la compañía LEGO en colaboración con el MIT. El kit básico proporciona los elementos básicos (sensores, servos y partes mecánicas) para construir robots sencillos.

De la colaboración entre LEGO y el MIT salieron diversos productos que no funcionaron, en gran medida debido a que la programación de estos productos debía hacerse en computadoras, y a principios de los años 90 las computadoras no estaban al alcance de cualquier usuario. A finales de los años 90, cuando se generalizó el acceso a las computadoras, LEGO sacó al mercado un nuevo producto. Este producto fue el RIS y se convirtió en la primera versión de la línea de productos LEGO MINDSTORMS. Salió al mercado en 1998, y contaba con un bloque o ladrillo (brick en inglés) con un microcontrolador de 8 bits (Hitachi H8/3000) sobre el que se ejecutaban los programas que hacían que el robot interactuase con el entorno. El nombre de este primer bloque fue RCX. El kit RIS estaba compuesto por el bloque RCX, sensores y actuadores así como

piezas estándares de LEGO que permitían la creación de un gran número de estructuras. También incluía un software de programación para crear programas de ejecución.

1.5.1.- Microcontrolador LEGO NXT 2.0

En 2006 salió al mercado el producto LEGO MINDSTORMS NXT en sustitución del kit RIS. Este nuevo producto era totalmente distinto en apariencia y prestaciones. El nuevo bloque se comercializó bajo el nombre de NXT. Las prestaciones que ofrecía eran mayores: USB, Bluetooth®, LCD más grande, microcontrolador de 32 bits (con mayor capacidad de cómputo), etc. Además, se incorporaron nuevos sensores más precisos y motores más potentes.

Desde 2009, LEGO comercializa una nueva versión, el LEGO MINDSTORMS NXT 2.0. Este modelo ofrece las mismas características que el modelo anterior pero con algunas mejoras en cuanto al software y sensores.

El controlador, bloque NXT o brick, contiene una CPU con capacidad de ejecutar programas de diferente complejidad. Concretamente, cuenta con un microprocesador Atmel ARM7 de 32 bits a 48 MHz. Se trata de un microprocesador utilizado extensivamente en electrónica de consumo (PDAs, teléfonos móviles, reproductores digitales multimedia,…) así como en otras aplicaciones empotradas como por ejemplo en la industria del automóvil. Además, la plataforma cuenta con un coprocesador, Atmel AVR de 8 bits, y con una memoria Flash de 256 KB y una memoria RAM de 64 KB. Con respecto a las comunicaciones, el controlador soporta tanto comunicaciones inalámbricas con Bluetooth® (especificación v2.0 EDR) como comunicación por cable usando tecnología USB (estándar 2.0) que soportan tasas de transferencia de datos de hasta 2.1 y 12 Mbits/s respectivamente. El bloque NXT cuenta también con 4 puertos de entrada con conexión RJ12 (conector telefónico de 6 hilos), que permiten conectar sensores tanto digitales como analógicos y 3 puertos de salida RJ12 que se utilizan para conectar diferentes tipos de actuadores, normalmente servomotores. Además, el controlador cuenta con una pantalla LCD gráfica de 100x64

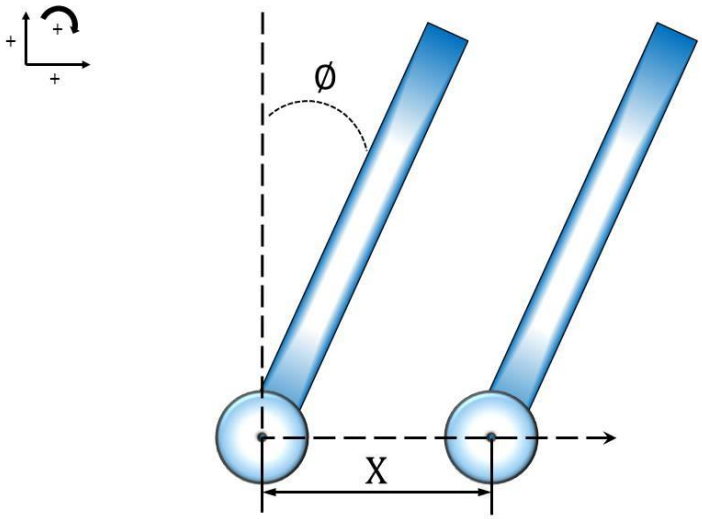
pixeles que se puede utilizar en modo gráfico para dibujar figuras, un altavoz con un canal de sonido con 8 bits de resolución capaz de generar tonos en el rango de 2 a 16 KHz y 4 botones que permiten interactuar con el bloque NXT.

A pesar de que el kit básico de LEGO sólo cuenta con cuatro sensores que miden luminosidad, volumen sonoro, ultrasonidos y un sensor de contacto, existe una amplia gama de sensores compatibles con la plataforma LEGO que miden muchas más variables: Aceleración, inclinación, posición (GPS), dirección (brújula), giróscopo, presión atmosférica, presión neumática, temperatura, magnitudes eléctricas (tensión, corriente, conductividad), magnitudes químicas (pH, salinidad, etc.) y muchas más. Dos de los suministradores más frecuentes de estos sensores son HiTechnic y MindSensors, que no sólo ofertan sensores sino otros componentes compatibles con la plataforma LEGO MINDSTORMS. También existe un adaptador para conectar sensores Vernier que amplía aún más el rango de sensores disponibles para medir magnitudes con la plataforma LEGO.

1.5.2.- Modelo teórico

Antes de abordar el problema es necesario hacer un estudio teórico. El sistema básicamente es un péndulo invertido. Este problema ya ha sido estudiado en muchas ocasiones y hay múltiples interpretaciones y soluciones al respecto. En este caso abordaremos el problema desde cero. Primeramente ilustramos el movimiento del cuerpo rígido y las variables que aparecen.

Ilustración 25 - Modelo teórico



Observando el modelo teórico es lógico deducir rápidamente que las dos variables que nos interesa medir son **el ángulo de inclinación** y **la posición del robot**.

Para medir el ángulo de inclinación necesitaremos un sensor que sea bastante rápido y efectivo. Lego dispone de sensores que son capaces de detectar la inclinación y la velocidad con la que cae. Estos son el acelerómetro y el giróscopo que se adaptan con facilidad al microcontrolador. Para nuestro problema acoplaremos un giróscopo HiTechnic cuyas características se detallan a continuación.

- **Sensor giróscopo**

El **giróscopo** o **giroscopio** es un dispositivo mecánico que sirve para medir, mantener o cambiar la orientación en el espacio de algún aparato o vehículo.

Está formado esencialmente por un cuerpo con simetría de rotación que gira alrededor del eje de dicha simetría. Cuando el giróscopo se somete a un momento de fuerza que tiende a cambiar la orientación de su eje de rotación, tiene un comportamiento aparentemente paradójico, ya que cambia de orientación (o experimenta un momento angular en todo caso, si está restringido) girando respecto de un tercer eje, perpendicular tanto a aquel respecto del cual se lo ha empujado a girar, como a su eje de rotación inicial. Si está montado sobre un soporte de Cardano que minimiza cualquier momento angular externo, o si simplemente gira libre en el espacio, el giróscopo conserva la orientación de su eje de rotación ante fuerzas externas que tiendan a desviarlo mejor que un objeto no giratorio; se desvía mucho menos, y en una dirección diferente.

Presenta, por tanto, dos propiedades fundamentales: la **inercia giroscópica** o "rigidez en el espacio" y la **precesión**, que es la inclinación del eje en ángulo recto ante cualquier fuerza que tienda a cambiar el plano de rotación. Estas propiedades se manifiestan a todos los cuerpos en rotación, incluida la Tierra. El término giróscopo se aplica generalmente a objetos esféricos o en forma de disco montados sobre un eje, de forma que puedan girar libremente en cualquier dirección; estos instrumentos se emplean para demostrar las propiedades anteriores, para indicar movimientos en el espacio, o para producirlos.

Éste fenómeno físico, el **efecto giroscópico**, puede observarse fácil y cotidianamente en peonzas, o monedas lanzadas a rodar, por ejemplo, aunque por supuesto, cualquier objeto giratorio funciona en cierto modo, como giróscopo. El giro en vuelo impartido por el jugador a un balón de rugby, o el de una bala disparada desde un arma de ánima rayada para estabilizar su trayectoria son ejemplos de aplicación del efecto.

Usaremos pues el sensor giroscópico NXT Gyro de Hi-technic. El sensor Giróscopo puede medir rotación en una escala de ± 360°. Con él se pueden construir y controlar robots que se pueden balancear, donde la medición de la rotación es esencial.

Ilustración 26 – Giróscopo Lego

El giróscopo de Lego devuelve la velocidad en rad/s del movimiento en un eje. Hay que colocar el sensor correctamente en el robot para que funcione adecuadamente. Realmente el valor que nos interesa saber es el ángulo de inclinación del robot/giróscopo. Para eso basta con tomar el valor del giróscopo durante un intervalo de tiempo. Si se multiplica ese valor por el tiempo transcurrido obtenemos el ángulo girado.

$$velocidad\ angular * tiempo = ángulo$$

El sensor da variación relativa de la posición, no es absoluta. Por tanto, al comenzar a usar el sensor hay que tomarle un valor de offset para que las siguientes lecturas sean más exactas. El sensor, al dejarlo en reposo, debería devolver un cero ya que la velocidad es cero. En cambio, devuelve un valor bajo pero distinto de cero. Para ello, se implementará una función que lea ese valor en reposo y lo guarde como valor de offset para todas las mediciones futuras del giróscopo. El offset debería ser constante en todo momento, sin embargo no es así. Éste se puede ver afectado por varios factores. Es lo que se llama *deriva* del sensor. Uno de los factores es la temperatura. El sensor no

funciona igual en caliente que en frio. Si queremos que la lectura sea más fiable tendríamos que tomarla una vez que el sensor se ha calentado. Otro factor importante es el voltaje de la batería o pilas. No se obtendrá el mismo resultado con pilas que con baterías, ni con la batería recién cargada que a punto de apagarse. Además, cuando los actuadores se activan, la tensión cae un poco debido al consumo.

Para ser más exacto aún en la lectura del sensor se ha usado la *media móvil exponencial* para introducir el efecto deriva del giróscopo. La media móvil exponencial lo que hace es ir corrigiendo el offset proporcionalmente a lo lejos que se encuentre del punto de equilibrio. Mientras más lejos del punto de equilibrio esté, mayor será en efecto deriva y viceversa.

Para saber en qué posición relativa está el robot recurriremos a un sensor que lleva adaptado el propio servomotor de Lego. El sensor es un encoder que es capaz de contar las vueltas que da el eje del motor. De hecho tiene una precisión de 360 cuentas por vuelta.

- **Servomotores Lego**

Para que el robot disponga de movilidad se instalarán en él dos servomotores de corriente continua construidos en base a una gran cantidad de engranajes internos. Éstos pueden rotar hasta alcanzar 170 rpm, trabajan en un rango de -9V a +9V y tienen un peso de 80 gramos cada uno.

Ilustración 27 - Servomotor Lego

1.6.- Opciones Software

Para programar nuestro kit robot podemos elegir entre varios lenguajes. El kit original cuenta con un lenguaje de programación propio basado en bloques visuales. El funcionamiento es muy bueno pero está limitado por el propio lenguaje. Además contamos con múltiples opciones para programar el NXT 2.0.

A continuación se enumeran los lenguajes que hay disponibles:

- **LeJOS NXJ.** Es una versión basada en Java con soporte para ambas versiones del ladrillo (RCX/NXT).
- ***BrickOS*** *(antes LegOS).- Basado en C.*
- ***RobotC***
- ***NQC****. (Not Quite C).- Lenguaje para el RCX*
- ***IC****. (Interactive C).*
- ***MathLab****. Por medio de una extensión de terceros. Soporta (RCX/NXT)*
- ***ROS****. Si tienes instalado Linux en tu computadora, este lenguaje es lo que buscabas*
- ***IAR*** *Embedded Workbench.- Ensamblador, C y C++*

- ***LUA***. *Se puede utilizar en el mindstorms por medio del firmware pblua*
- ***LeJOS OSEK***. *Basado en ANSI C/C++*
- ***Lisp***. *Para el RCX y NXT*
- ***Ruby NXT***.
- ***NXT ++.*** *No es realmente un lenguaje, sino más bien una buena interface que utiliza el C++*
- ***ADA.*** *Solo para el RCX*
- ***nxcEditor.*** *Editor para lenguaje C que corre en ambiente Linux. Tiene un simulador*
- ***Prolog***. *Lenguaje conocido por ser usado en la investigación de la inteligencia artificial*
- **NXC.** Not Exactly C.

Los siguientes lenguajes trabajan con el firmware original de Mindstorms:

- ***RoboLab***. *Es el lenguaje gráfico que acompañaba al RCX*
- ***LabView***. *Por medio de una extensión diseñada para el Mindstorms.*
- **NBC**. (Next Byte Codes)/NXC (Not eXactly C).- Soportan ambas versiones del ladrillo.
- ***Microsoft Robotics Developer Studio***.
- ***Gostai***. *URBI for LEGO MINDSTORMS NXT*
- ***Visual Basic, Visual C++, Delphi, Visual Java++.*** *Solo para el RCX. Se controla por medio de un control ActiveX*
- ***LOGO***. *Aquél famoso lenguaje, puede ser interpretado por el NXT*
- ***CHERP***. *Creative Hybrid Environment for Robotic Programming. Sirve tanto para el ladrillo RCX como para Lego WeDo*

Como se puede observar, para muchos de los lenguajes se ha recurrido al C como base. La elección del lenguaje a utilizar depende de factores tales como: Complejidad del proyecto, experiencia en programación y hasta gusto personal.

Se han hecho pruebas con el LABVIEW®, con MATLAB® y con NXC. Se ha elegido el **NXC** por su accesibilidad a todas las variables y por preferencia del usuario al usar un entorno parecido a lenguaje C. De hecho, NXC son las siglas de "Not Exactly C", no exactamente C.

1.7.- Resultados

Con todos estos componentes del robot la configuración queda de la siguiente manera:

Ilustración 28 – Maqueta Lego

Ilustración 29 - Dispositivo Alzado

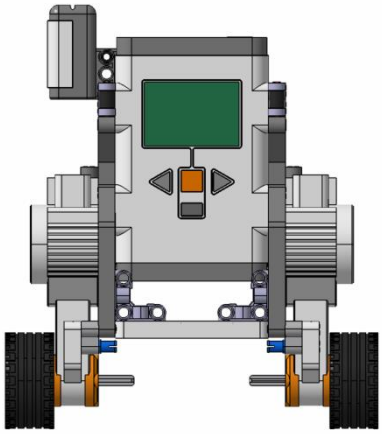


Ilustración 30 - Dispositivo perspectiva

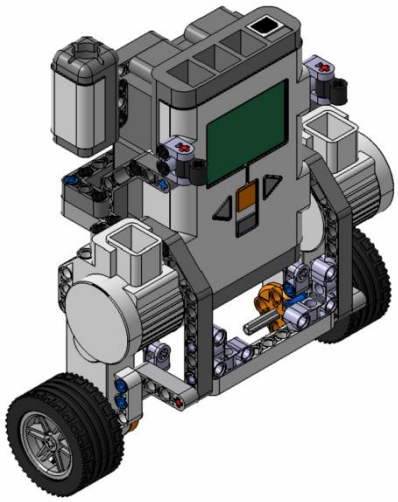
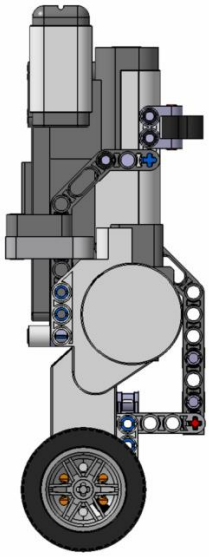


Ilustración 31 - Dispositivo perfil



Ya está lista la maqueta con la que vamos a trabajar. Ahora se va a hacer un análisis del problema desde un punto de vista teórico.

1.8.- Pruebas preliminares

Lo primero que se hará es estudiar bien el entorno del lenguaje y hacer muchas pruebas con los periféricos para obtener un buen manejo de los mismos. Para estudiar el comportamiento de los dos sensores, de los servomotores y del microcontrolador se han creado varios programas-ensayo para cada uno.

1.8.1.- Prueba servomotores

La primera prueba es muy sencilla. Solo se trata de ver el movimiento de un servomotor para estudiar su comportamiento.

El código solo emite un sonido a modo de inicio, a continuación activa el motor en una dirección y después de una espera corta lo activa en la otra dirección posible.

```
task main ()
{
PlaySound(SOUND_CLICK);
Wait (1000);
OnFwd(OUT_AC, 50);
Wait (5000);
OnFwd(OUT_AC, -50);
Wait (5000);
Off(OUT_AC);
}
```

1.8.2.- Prueba encoders servomotores

Este código hace que los motores avancen en una dirección al 20% de su velocidad (`OnFwd(OUT_AC, 20)`), el encoder de cada motor cuenta las vueltas giradas (`Cuenta_I = MotorRotationCount(OUT_C)`) y las muestra en pantalla (`NumOut(15, 8, Cuenta_I)`). Así seremos capaces de ver cómo funciona el encoder, la resolución y la calidad de la medición.

```
task main()
{
   float Cuenta_I = 0.0;
   float Cuenta_D = 0.0;
   long i = 0.0;
   ResetRotationCount(OUT_C);
   ResetRotationCount(OUT_A);
   OnFwd(OUT_AC, 20);
   for(i=0;i<100;i++)
   {
   Cuenta_I = MotorRotationCount(OUT_C);
   Cuenta_D = MotorRotationCount(OUT_A);
   NumOut(15, 8, Cuenta_I);
   NumOut(15, 16, Cuenta_D);
   Wait(30);
   }
```

```
   OnRev(OUT_AC, 20);
   for(i=0;i<200;i++)
   {
   Cuenta_I = MotorRotationCount(OUT_C);
   Cuenta_D = MotorRotationCount(OUT_A);
   NumOut(15, 8, Cuenta_I);
   NumOut(15, 16, Cuenta_D);
   Wait(30);
   }
}
```

1.8.3.- Obtener valor del giróscopo

```
#define GYRO_PORT IN_3
float Lectura_Giroscopio;

task main()
{
SetSensorHTGyro(GYRO_PORT);
while(1)
        {
        Lectura_Giroscopio = SensorHTGyro(GYRO_PORT);
        NumOut(15, 8, Lectura_Giroscopio);
        Wait(100);
        ClearScreen();
        }
```

Conectamos el giróscopo al puerto de entrada 3 del brick ó microcontrolador. Este código lee el valor que dá el sensor conectado al puerto de entrada 3 y lo muestra en pantalla para así poder verificarlo. El valor obtenido es Grados/segundo.

1.8.4.- Recibir mensajes vía Bluetooth®

Este código recibe 2 mensajes vía Bluetooth®, uno por el canal cero y otro por el canal 1. Guarda ambos mensajes en sus respectivas variables y los muestra en pantalla.

```
task main()
{
   string msg0;
   string msg1;

   while(true)
   {
         ReceiveRemoteString(0,true,msg0);
         TextOut(10,LCD_LINE5,msg0);
         ReceiveRemoteString(1,true,msg1);
         TextOut(10,LCD_LINE6,msg1);
         Wait(20);
   }
}
```

1.8.5.- Ensayo con el robot completo

La primera prueba global con el robot consiste en usar un sensor/emisor de luz que tras dejar el robot en equilibrio manualmente, emite una luz roja y acto seguido toma una muestra de la luminosidad recibida a través del sensor. Esto se guarda en una variable y se usa como referencia para mantener nuestro kit siempre alrededor de esa referencia. Cuando el robot se cae hacia delante el sensor se sitúa más cerca del suelo y por tanto la luminosidad recibida por el sensor es mayor. El autómata actuará moviendo las ruedas hacia delante para mantener el equilibrio (la luminosidad “referenciada”). Por el contrario, si el robot se cae hacia atrás la luminosidad desciende y actuará sobre las ruedas en movimiento reverso para conseguir el equilibrio.

Rápidamente se puede ver que existen varios inconvenientes para este montaje. El equilibrio inicial para tomar una primera lectura ha de ser casi perfecto ya que ese es el valor de referencia que nuestro robot intentará mantener siempre. El inicio está hecho por el usuario a mano así que es muy fácil desviarlo 1 o 2 grados sin percibirlo. El resultado es que el robot nunca mantendrá un punto estable, estará continuamente corrigiendo la posición hasta que perderá el equilibrio.

Otro segundo inconveniente es que el suelo debe ser firme y homogéneo en su textura. Cualquier variación hace que la lectura de luminosidad del sensor varíe y se desestabilice.

Por tanto, el robot se queda "más o menos" en equilibrio pero la corrección es constante y en poco tiempo el robot cae. El método no es bueno. Para evitar esto recurrimos a un sensor mucho más efectivo, el giróscopo o giroscopio. El principio de funcionamiento es muy sencillo.

1.8.6.- Ensayo nº1. PID1 sin movimiento

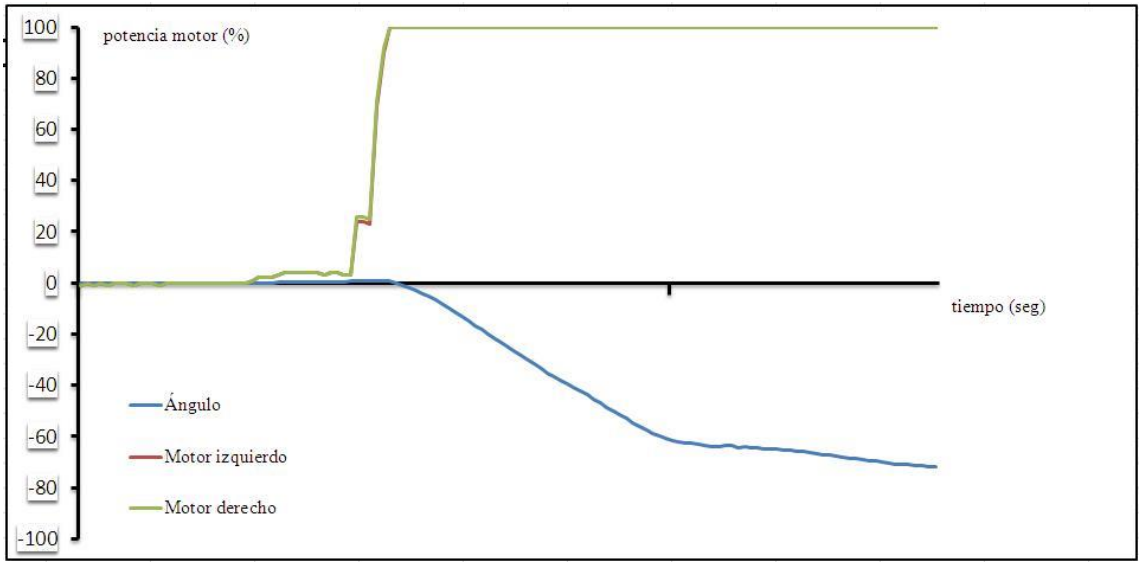
Este es un ensayo que se realiza para ver la calidad del equilibrio, la estabilidad del dispositivo. Los valores de los PI que se han obtenido teóricamente servirán como punto de partida aunque a través de estos ensayos se ajustarán todo lo posible para afinar la calidad del equilibrio todo lo posible.

No vamos a mandarle ningún comando para que avance o retroceda, solo necesitamos ver como de estable es el robot y cuánto tiempo es capaz de mantener el equilibrio.

```
#define Ki_Giro 1
#define Kp_Giro 1
#define Ki_Motores 1
#define Kp_Motores 1
```

```
#define KDRIVE -1
#define K_Direccion 1
```

Ilustración 32 - Gráfica PI 1



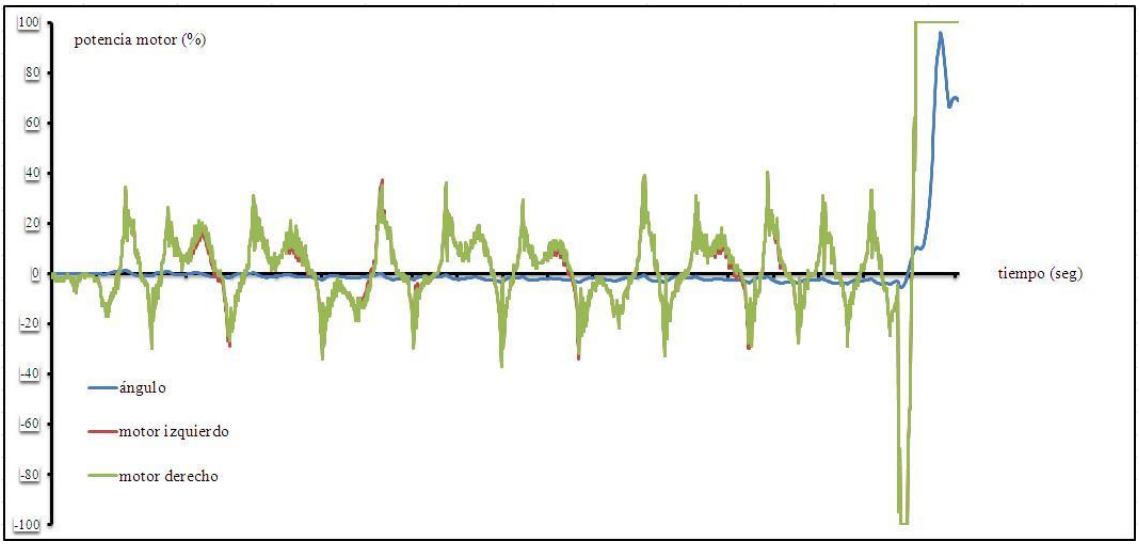
El robot cae inmediatamente después de iniciar el equilibrio. Se observa que cae demasiado rápido, intenta recuperar el ángulo demasiado rápido. Esto nos lleva a realizar los primeros cambios en los valores de PID.

1.8.7.- Ensayo nº2. PID2 sin movimiento

Cambiamos valores de las constantes de los PI para conseguir más estabilidad. El proceso es intuitivo, simplemente observando la reacción del dispositivo puede arreglarse mucho.

```
#define Ki_Giro 10
#define Kp_Giro 1
#define Ki_Motores 0.1
#define Kp_Motores 0.1
#define KDRIVE -0.1
#define K_Direccion 0.1
```

Ilustración 33 - Gráfica PI 2



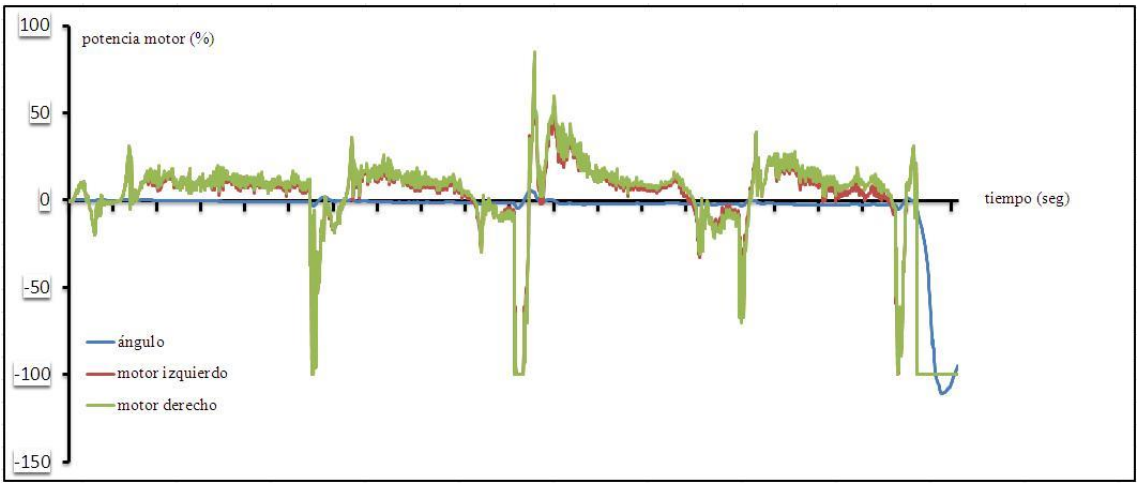
Ya se queda en equilibrio durante un tiempo considerable (cada marca en el eje de abscisas equivale a un segundo). A los 20 segundos de permanecer en equilibrio se le aplica una perturbación, se le dá un golpe suave, y el resultado es la caída. El golpe ha sido bastante suave, debería poder corregir la posición para volver a encontrar el equilibrio. Además, el ángulo del giróscopo es demasiado fluctuante. Se observa una constante corrección en la potencia que se le pasa a los motores. Proseguimos corrigiendo los valores de PID.

1.8.8.- Ensayo nº3. PID3 sin movimiento

Se actualizan los valores de las constantes a criterio personal del programador. Después de retocar muchas veces se llega a unos números obtenidos empíricamente.

```
#define Ki_Giro 17.5
#define Kp_Giro 1.15
#define Ki_Motores 0.07
#define Kp_Motores 0.1
#define KDRIVE -0.02
#define K_Direccion 0.25
```

Ilustración 34 - Gráfica PI 3

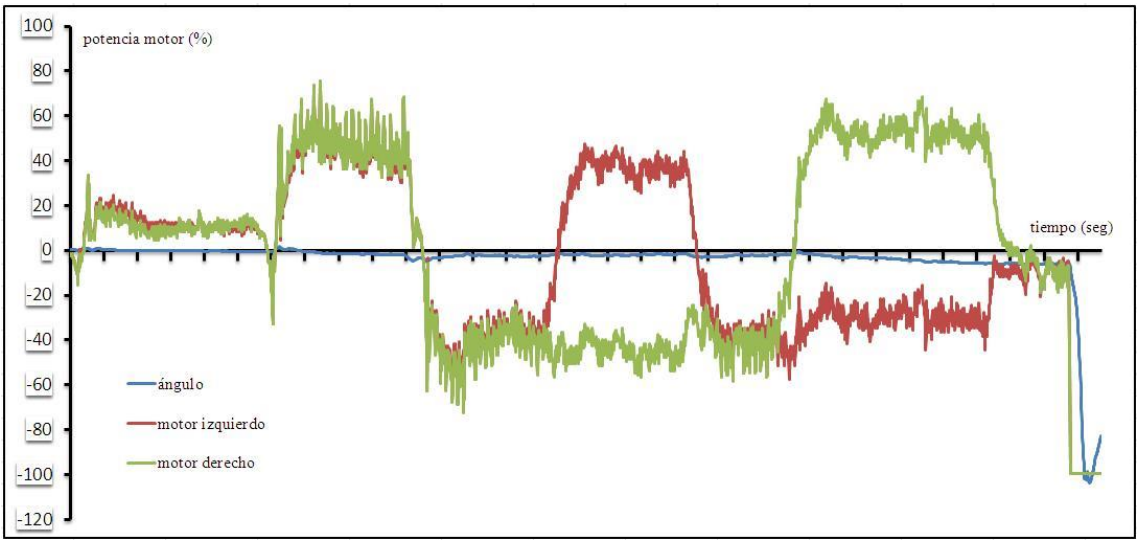


Efectivamente, el equilibrio ha mejorado muchísimo. La corrección del controlador es rapidísima y muy efectiva. Para llevarlo al límite se le han dado 4 golpes para alterar la estabilidad. Concretamente, a los 6, 10, 15 y 20 segundos del comienzo. Las tres primeras perturbaciones las aguanta sin problema, los motores corrigen perfectamente la posición y logra mantener el equilibrio. El cuarto golpe es más fuerte y hace que el robot pierda el equilibrio.

1.8.9.- Ensayo nº4. PID3 con movimiento

Las constantes del controlador ya están fijadas. Ahora se realiza una prueba para ver cómo responde al equilibrio en movimiento. A los seis segundos de comenzar el programa le mandamos una señal de avance hacia delante, a los 10 segundos le hacemos retroceder y a los 14 y a los 21 segundos le hacemos girar a derecha y a izquierda respectivamente. El resultado es el siguiente:

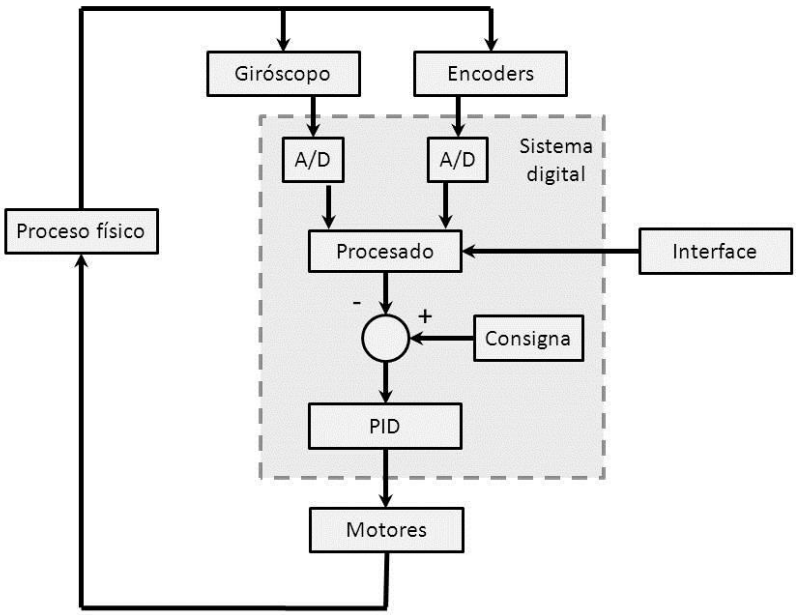
Ilustración 35 - PID 3 con movimiento



1.9.- Conclusiones y líneas futuras

1.9.1.- Diagrama funcional

Ilustración 36 - Diagrama funcional



1.9.3.- Asunciones y aproximaciones

Con el fin de simplificar el estudio del sistema físico podemos suponer y asumir las siguientes aproximaciones:

- LTI. Sistema linear en el tiempo.

Un sistema es lineal si satisface el *principio de superposición*, que engloba las propiedades de proporcionalidad o escalado y aditividad. Que sea proporcional significa que cuando la entrada de un sistema es multiplicada por un factor, la salida del sistema

también será multiplicada por el mismo factor. Por otro lado, que un sistema sea aditivo significa que si la entrada es el resultado de la suma de dos entradas, la salida será la resultante de la suma de las salidas que producirían cada una de esas entradas individualmente.

Para nuestro trabajo es necesario suponer linealidad así podremos resolver las ecuaciones diferenciales con facilidad. Existen métodos que se pueden aplicar a estas ecuaciones pero siempre con la condición de sistema lineal en el tiempo.

- SISO. Sistema de una sola entrada y una sola salida.

Para realizar el estudio del controlador que se va a usar es aconsejable trabajar con sistemas SISO (Single Input Single Output) ya que en consecuencia podremos recurrir a herramientas informáticas muy útiles para diseñar un controlador.

1.10.- Bibliografía y otras referencias

1.10.1.- Libros y apuntes

- LEGO MINDSTORMS NXT-G Programming guide. James Floyd Kelly. Apress. 2007.
- Uso conjunto de la plataforma LEGO MINDSTORMS NXT y metodologías PBL en Informática Industrial. Isidro Calvo, Gorka Perianez
- LEGO MINDSTORMS NXT. Owen Bishop. E-book gratuito. 2008.
- LEGO MINDSTORMS NXT POWER PROGRAMMING. John C. Hansen. Variant press.
- NXT-G Programming. Workshop for FLL Coaches. Developed by Tony Ayad Updated by LeRoy Nelson. California - Los Angeles Region FLL. 2012
- Manual del usuario NXT 2.0.

- INGENIERÍA DE CONTRL MODERNA. Katsuhiko Ogata. Editorial Pearson. 5ª edición. 2010.
- Norma UNE 157001:2014
- Fundamentos de control con MATLAB®. Enrique Pinto Bermúdez y Fernando Matía Espada. Editorial Pearson. 2010.
- Apuntes Regulación automática. Escuela universitaria politécnica. 2013.

1.10.2.- Otras referencias

- Diccionario de ingeniería: ***http://diccionario.raing.es/***
- Página de la escuela politécnica con guión para los proyectos: ***http://www.eps.us.es/docencia/docencia/tfgtfmpfc/pfc***
- Modelado de un péndulo invertido ***http://www.ib.cnea.gov.ar/~instyctl/Tutorial_Matlab_esp/invpen.html***
- Curvas características de los motores LEGO ***http://www.philohome.com/nxtmotor/nxtmotor.htm***
- Presentación ejemplo en Prezi ***https://prezi.com/-mk2qilvqsav/pendulo-invertido/***
- Comunicación Bluetooth® con LEGO ***http://crysol.org/es/node/779***
- Pagina de descargas de hitechnic ***http://www.hitechnic.com/file.php?f=786-HTWayC.nxc***
- Página oficial de hitechnic ***http://www.hitechnic.com/***
- Página oficial de sensores mindsensors ***http://www.mindsensors.com/***
- Sensores Vernier ***http://www.vernier.com/probes/***
- Envio mensajes desde PC a NXT por Bluetooth® ***http://www.mindsensors.com/forums/viewtopic.php?f=5&t=226***
- Página de descargas de LEGO

http://education.lego.com/en-gb/downloads/?q=%7bdc0ce993-6544-45a1-8680-b2a547d1eeb6%7d

- Interior de los motores. Caracteristicas.

  http://www.philohome.com/nxtmotor/nxtmotor.htm

- Descarga del programa de control "Motor Control"

  http://www.mindstorms.rwth-aachen.de/trac/wiki/MotorControl

- Bluetooth® con Visual basic

  http://iesromerovargas.es/recursos/elec/sol/basic4android24.htm

- Cálculo del momento de inercia

  http://hyperphysics.phy-astr.gsu.edu/hbasees/mi2.html

- Mindstorms Toolbox

  http://www.mindstorms.rwth-aachen.de/trac/wiki/MotorControl

- PID tuner en MATLAB®

  http://es.mathworks.com/videos/pid-control-design-with-control-system-toolbox-68748.html

- Segway con control a traves de mando PSP

  http://www.techbricks.nl/My-NXT-projects/nxt-self-balancing-segway-nxtway-robot.html

- Lego en Catia

  http://academy.3ds.com/news/play-lego-with-catia-v5/

- Otros ejemplos (1)

  https://sites.google.com/site/txemafraga/robotequilibrio

- Otros ejemplos (2):

  http://electronica-iespolitecnico-cartagena.blogspot.com.es/2014/01/robot-equilibrista-con-servos-y-arduino.html

- Otros ejemplos (3):

  http://www.tecnoneo.com/2013/12/el-robot-mip-de-wowwee-mantiene-el.html

- Lugar de las raíces, tutorial:

  http://www.ib.cnea.gov.ar/~instyctl/Tutorial_Matlab_esp/rlocus.html