Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FI DE CARRERA

TÍTOL:

Construcción y control de una maqueta de un ciclotrón electro-mecánico

AUTOR: Marc Camarasa Andrés i Humphrey Diéguez Fernández

TITULACIÓ: Enginyeria Automàtica i Electrònica industrial

DIRECTOR: Pau Martí Colom

DEPARTAMENT: 707 Enginyeria de Sistemes Automàt. i Informàtica Ind.

TÍTOL:

Contrucción y control de una maqueta de un ciclotrón electro-mecánico

COGNOMS: Camarsa Andrés **NOM: Marc**

COGNOMS: Diéguez Fernández **NOM: Humphrey**

TITULACIÓ: Ingeniería Automática y Electrónica Industrial

ESPECIALITAT: Automática y Electrónica Ind. PLA: 2003

DIRECTOR: Pau Martí Colom

DEPARTAMENT: Enginyeria de Sistemes Automàt. I Informàtica Ind.

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT **SECRETARI** **VOCAL**

DATA DE LECTURA:

Aquest Projecte té en compte aspectes mediambientals: □ Sí No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

¿Es posible acelerar una bola mediante la generación de campos magnéticos?
Este proyecto tiene como objetivo diseñar e implementar una maqueta que permita acelerar una bola metálica en un circuito cerrado, inspirado en un ciclotrón. La bola metálica debe acelerarse dentro de un circuito cerrado a partir de buscar disparos óptimos en una bobina. Los disparos se buscarán en función de la velocidad de la bola y de la distancia de esta respecto la bobina.
Para simular el sistema utilizamos el programa SIMULINK de MATLAB a partir de las ecuaciones diferencias del sistema y utilizando la función fminsearh para encontrar los disparos óptimos.
En las simulaciones comprobamos que era posible encontrar puntos de disparo y acelerar la bola mediante campos magnéticos producidos por un grupo de bobinas.
Para llevar esto a la práctica se ha realizado el montaje de una maqueta basándonos en el modelo de un ciclotrón.
Decidimos utilizar las bobinas más pequeñas que nos permitieran mover la bola y de esa manera maximizar la importancia de la obtención de los puntos de disparo óptimos.
Para implementar el control y probar las simulaciones realizadas decidimos utilizar una placa FLEX que el departamento disponía en ese momento. Esta placa incluye un microprocesador DSPIC con un Kernel en tiempo real.
Una vez construida la maqueta y realizada su puesta en marcha más programación del control en la placa FLEX se han determinado los puntos de disparo óptimos y se han obtenido resultados prácticos que confirman las simulaciones realizadas en el inicio del proyecto.

Paraules clau (màxim 10):

Ciclotrón	Maqueta	Simulink	Aceleración
Dspic	Kernel	Interrupción	Bobina
Sensor	fminsearch		

ÍNDICE:

1. INTRODUCCIÓN: OBJETIVOS Y JUSTIFICACIÓN PFC 6

1.2 ESTADO DEL ARTE 6

2. SIMULACIONES 7

2.1 CONCLUSIONES DE LAS SIMULACIONES 30

3. CONSTRUCCIÓN DE LA MAQUETA 31

3.1 DESCRIPCIÓN DE FUNCIONAMIENTO: 31
3.2 SELECCIÓN DE MATERIALES 32
3.2.1 Base de la maqueta 32
3.2.3 Sensores 32
3.2.3.1 Selección del sensor 32
3.2.3.2 Sensor montado en la maqueta 38
4.1 FLEX FULL BASE BOARD 48
4.2 FLEX DEMO DAUGTHER BOARD 50
4.3 DRIVER DE SEÑAL PARA CONTINUA 53
4.4 DRIVER DE SEÑAL PARA ALTERNA 56
5.2 RT DRUID 57
5.3 MPLAB IDE V 8.00 60
5.4 HYPERTERMINAL DE WINDOWS 61
5.5 MATLAB 62
6.1 CONTROL MEDIANTE PASO POR CENTRO DE LA BOBINA 64
6.2 CONTROL DE BÚSQUEDA DE PUNTOS ÓPTIMOS DE DISPARO 69
6.3 CONTROL MEDIANTE FUNCIONES HALLADAS A PARTIR DEL SEGUNDO TIPO DE CONTROL 73
6.3.1 Funciones de configuración de entradas de la Dspic 74
6.3.2 Funciones de configuración de salidas de la Dspic 74
6.3.3 Funciones de configuración de recursos de la Dspic 74
6.3.4 Funciones de configuración de salidas de la Dspic 75
6.3.5 Diagramas de funcionamiento 76

7. RESULTADOS 82

8. PRESUPUESTO 85

9. CONCLUSIONES 86

10. BIBLIOGRAFIA 87

AGRADECIMIENTOS

Este proyecto no se podría haber realizado sin la colaboración del departamento de Servicios Técnicos de Laboratorio por lo que queremos agradecerles todo su apoyo y ayuda a Kenneth, Oscar y Germán.

También agradecer a Pau Martí Colom por su implicación en este proyecto desde el inicio hasta el final.

Por último agradecer a David Pla por dejarnos herramientas para el montaje de la maqueta y a Joan Robert por proporcionarnos material y aconsejarnos en el montaje de la misma.

Marc Camarasa
Humphrey Diéguez

Febrero 2011

1. INTRODUCCIÓN: OBJETIVOS Y JUSTIFICACIÓN PFC

Este proyecto se realiza a petición del tutor del proyecto, a partir de la idea de construir una maqueta de un ciclotrón.

Para realizar este proyecto no disponíamos nada más que esa idea inicial. Se acordó empezar por una serie de simulaciones para comprobar la viabilidad del proyecto, seguido de la selección del material para la maqueta, la construcción de esta y la comprobación práctica del funcionamiento de la maqueta haciendo acelerar una bola metálica dentro de un tubo de plástico. Por lo tanto los objetivos del proyecto son:

- Comprobación de la viabilidad del proyecto mediante simulaciones del sistema con el programa Simulink (Matlab).
- Selección del material del proyecto.
- Construcción de la maqueta.
- Comprobación práctica de la aceleración de la bola.

Se pretende pues construir una maqueta programable donde se puedan realizar demostraciones de su funcionamiento básico a más de poder realizar diferentes prácticas a través de su programación.

1.2 Estado del arte

Desconocemos si existen o no otras maquetas parecidas a esta. Esta maqueta se ha desarrollado cogiendo de referencia el funcionamiento de un ciclotrón. Si bien se ha cogido como punto de partida las ecuaciones diferenciales de un levitador magnético como punto de partida aunque se les aplicado los cambios necesarios para obtener el modelo de nuestro sistema.

2. SIMULACIONES

Las simulaciones fueron el primer paso en la realización del proyecto. La intención original era comprobar si el proyecto era viable o no. En el momento de empezar con las simulaciones no teníamos ningún material seleccionado, por lo que el valor de los elementos fue elegido de manera totalmente arbitraria dentro de unos parámetros de valores posibles.

Las simulaciones se realizaron con el programa MATLAB. Basándonos principalmente en la aplicación SIMULINK.

Para realizar las simulaciones en Simulink originalmente se partió del siguiente sistema:

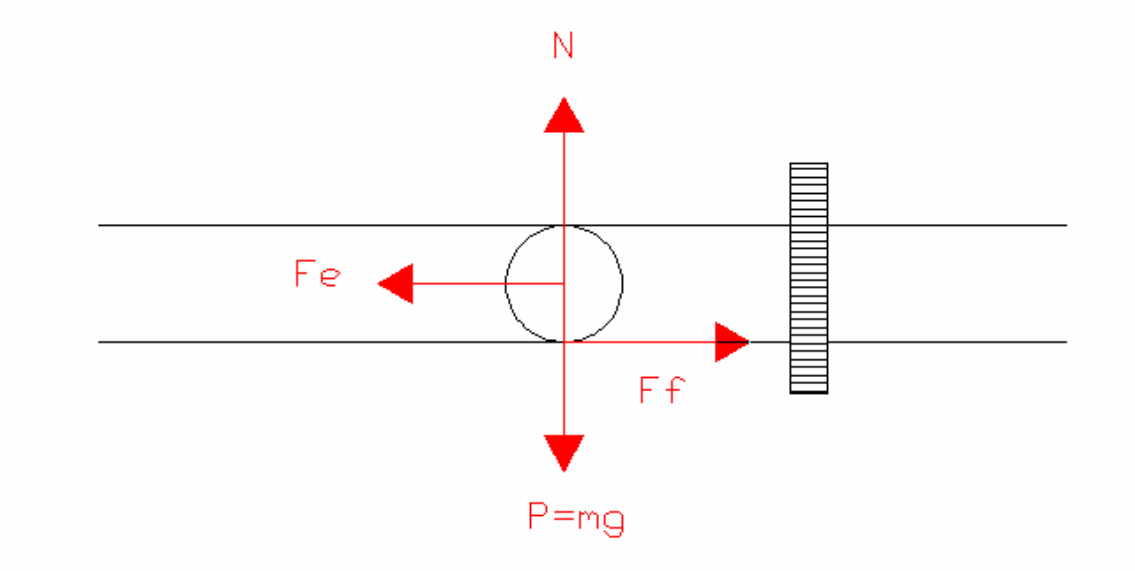


Figura 2.1. Sistema formado por bola dentro del tubo y bobina

Se han considerado las fuerzas que se presentan en el anterior esquema y utilizando la segunda ley de Newton obtenemos:

(1) $\sum F = m \cdot a \rightarrow F_e - F_F = m \cdot a$ donde:

(2) $F_e = L \cdot \frac{i^2(t)}{x^2}$, fuerza realizada por el campo magnético generado por la bobina

(3) $F_F = N \cdot \mu$, fuerza de fricción generada entre el tubo

Por lo que sustituyendo las ecuaciones 2 y 3 en la ecuación 1 obtenemos:

$$L \cdot \frac{i^2(t)}{x^2} - N \cdot \mu = m \cdot a$$

Donde como sabemos la aceleración es la derivada segunda de la posición por lo que

podemos expresar la ecuación anterior de la siguiente forma:

$$L \cdot \frac{i^2(t)}{x^2} - N \cdot \mu = m \cdot \frac{d^2}{dt} x(t)$$

Considerando el siguiente cambio de variables y aplicada en la ecuación anterior obtenemos la ecuación de estado con la que podremos simular el comportamiento de la bola frente la fuerza de la bobina:

$$\begin{aligned} X_1 &= x \\ \overset{*}{X}_1 &= v \\ \overset{**}{X}_1 &= a \end{aligned} \longrightarrow \quad L \cdot \frac{u^2}{X_1^{\,2}} - N \cdot \mu = m \cdot \overset{**}{X}_1 ,$$

Donde u es la entrada que apliquemos, en este caso u es la intensidad que consume el sistema.

Partiendo del anterior estudio se realizo el siguiente modelo de SIMULINK:

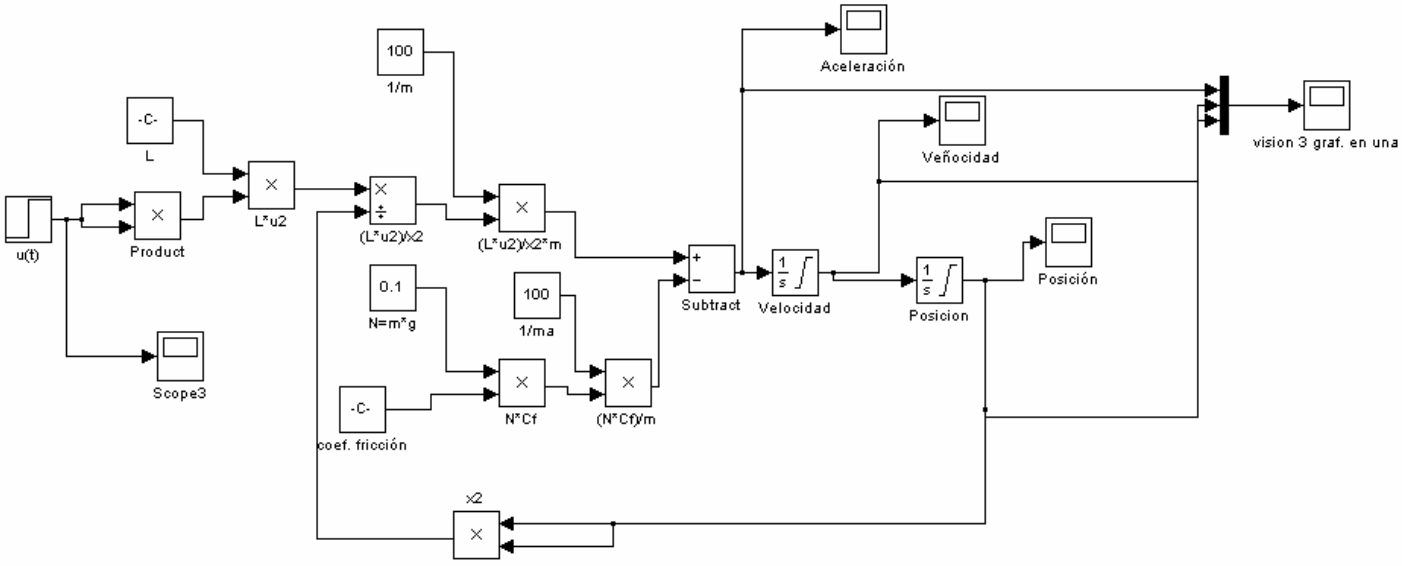


Figura 2.2. Modelo SIMULINK de la ecuación del sistema original

Podemos observar que la salida del sistema es la posición final de la bola sobre la cual ejercemos la fuerza magnética. Para ello aislamos la posición final:

$$\frac{L}{m} \cdot \frac{u^2}{X_1^{\,2}} - \frac{N \cdot \mu}{m} = \overset{**}{X}_1$$

Con este paso tenemos la aceleración de la bola, por lo tanto, hay que integrar dos veces para conseguir la posición:

$$X_1 = \iint \left(\frac{L}{m} \cdot \frac{u^2}{X_1^{\,2}} - \frac{N \cdot \mu}{m} \right) dt$$

Los valores de los elementos del circuito, escogidos de manera arbitraria, eran los siguientes:

L=5µH
m=0,1Kg
Fricción= 0.004N
Posición inicial=0.05

El valor 0 de la posición corresponde al centro de la bobina.

Este modelo nos permitió hacer una primera aproximación al sistema. Mirando la posición de salida comprobamos que era posible mover la bola mediante la fuerza magnética proporcionada por la bobina.

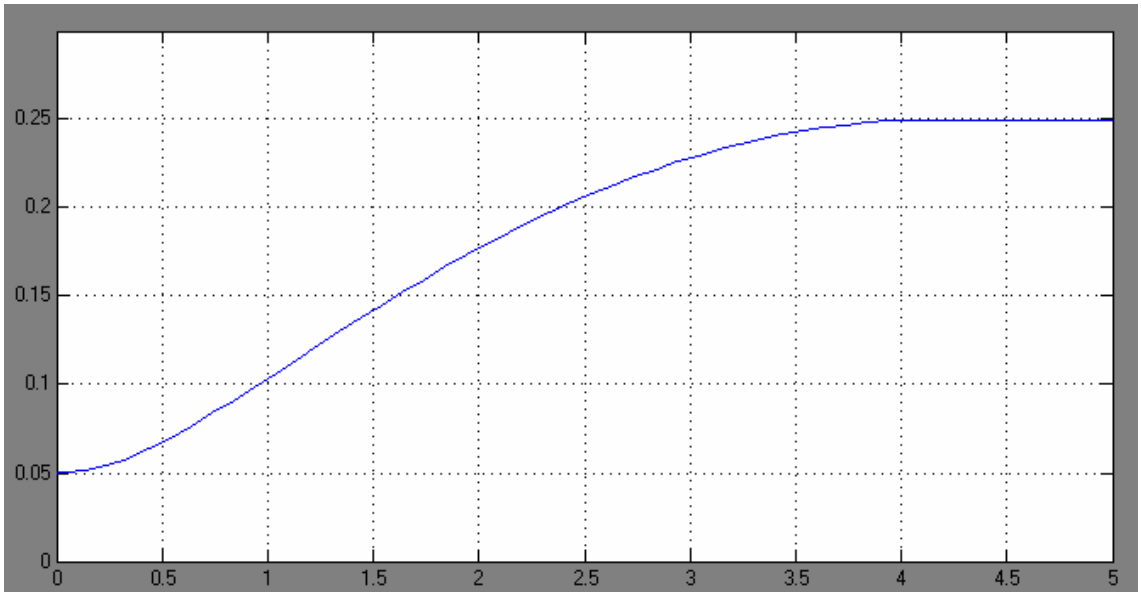


Figura 2.3. Gráfica posición Vs Tiempo

En este modelo el valor de entrada "*u*" es el valor de la corriente. Esto en la realidad no es así, el valor de corriente lo proporciona la bobina después de alimentarla a una fuente de tensión:

$$i = \int \left(\frac{v}{L} \right) dt$$

Pero como sabemos que la bobina tiene una resistencia intrínseca que limita su valor se ha tenido en cuenta esta resistencia en el siguiente modelo. Se ha calculado esta resistencia para que el valor de la intensidad no supere 0,5A pues se considera que este valor es factible de conseguir en el laboratorio, a demás es un valor lo suficientemente pequeño como para no falsear los resultados.
Otra mejora que se ha introducido en el nuevo modelo es la modelación de la fuerza de rozamiento. En el modelo anterior el valor de la fuerza de rozamiento era un valor

estático, una constante. En el nuevo modelo el valor de la fuerza de rozamiento depende de un valor estático, proporcional a su masa y a la gravedad, y a un valor dinámico proporcional a la velocidad.

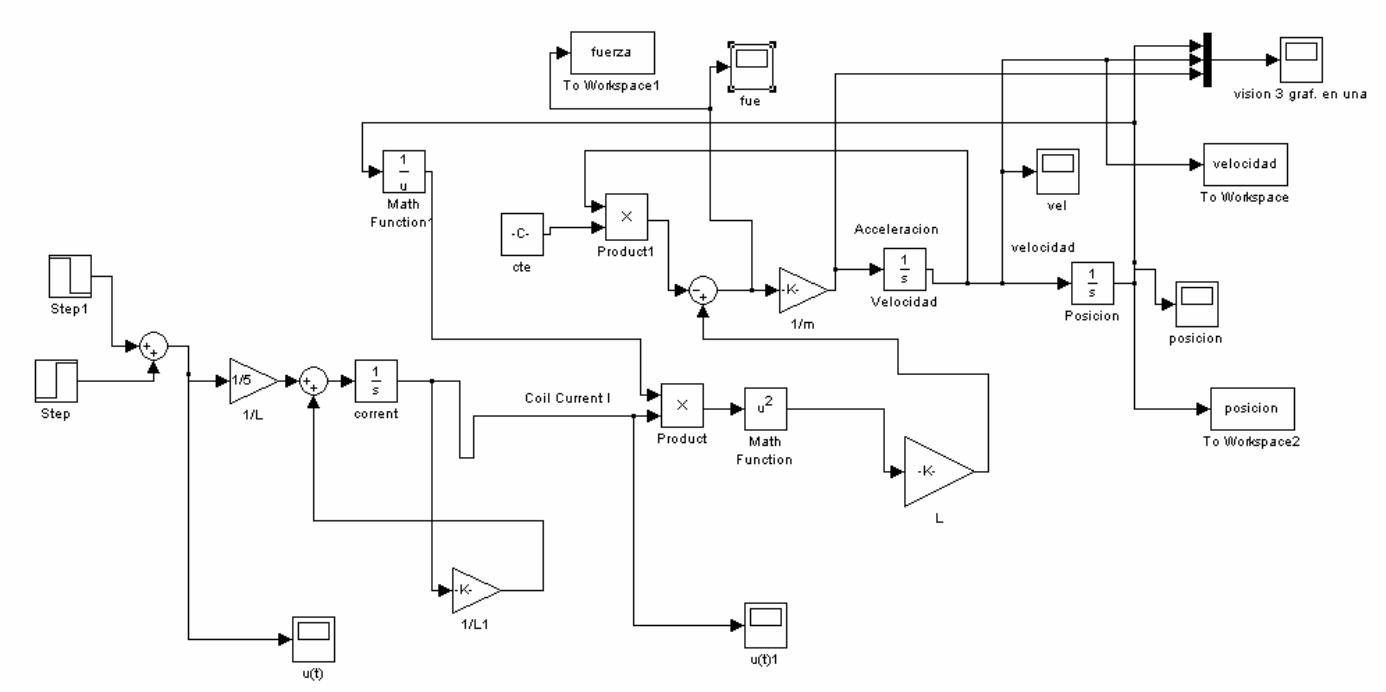


Figura 2.4. Modelo SIMULINK con bobina real y fricción dinámica

Como ya hemos visto antes el valor de la fuerza es proporcional al cuadrado de la fuerza:

$$F_e = L \cdot \frac{i^2(t)}{x^2}$$

Por lo tanto con pequeñas variaciones de la corriente se puede conseguir una gran variación en la fuerza magnética.

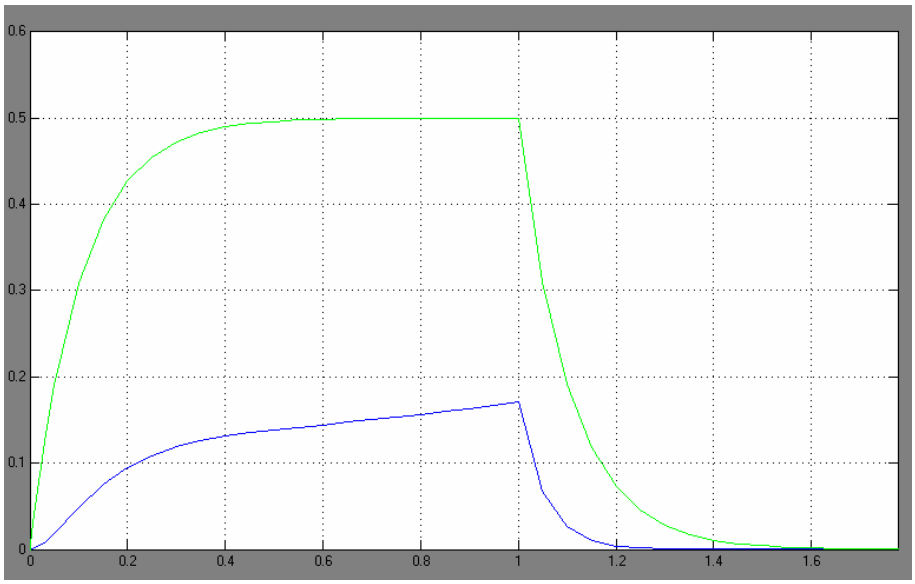


Figura 2.5. Gráfica corriente (fucsia) Vs Fuerza (amarillo)

En la gráfica anterior vemos los valores de corriente, saturada a 0,5A, y de la fuerza generada por esta. La fuerza ha sido escala x1000 para poder grafiarla junto con la corriente.

Observamos que el valor de la fuerza es de 0,17 10^{-3}N para una corriente de 0,5A. Observamos que pasa al doblar la corriente.

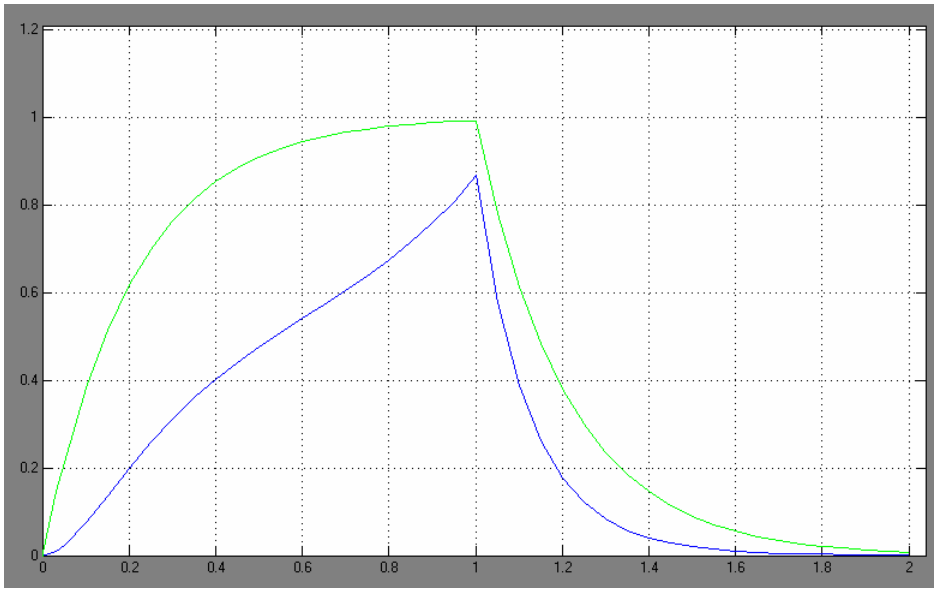


Figura 2.6. Gráfica corriente (verde) Vs Fuerza (azul)

Al doblar el valor de la corriente, de 0,5A a 1A, observamos que el valor de la fuerza se incrementa de manera notable y pasa de valer 0,17 10^{-3}N a valer 0,83 10^{-3}N.

De esta manera en el momento en que necesitemos incrementar la fuerza magnética, la mejor forma es incrementando el valor de la corriente por las bobinas. Esto lo conseguimos incrementando el valor de la tensión de alimentación.

A continuación veremos cómo evolucionan la posición (azul), la velocidad (verde) y la aceleración (rojo) para una corriente de 0,5A, y con el siguiente escalón de entrada

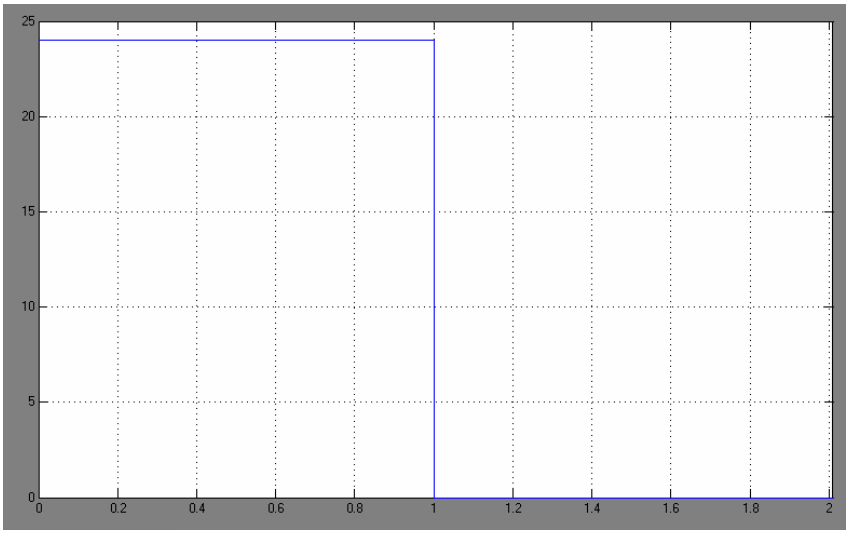


Figura 2.7. Gráfica escalón de entrada de 24V 1seg,

La respuesta para el anterior escalón de entrada es:

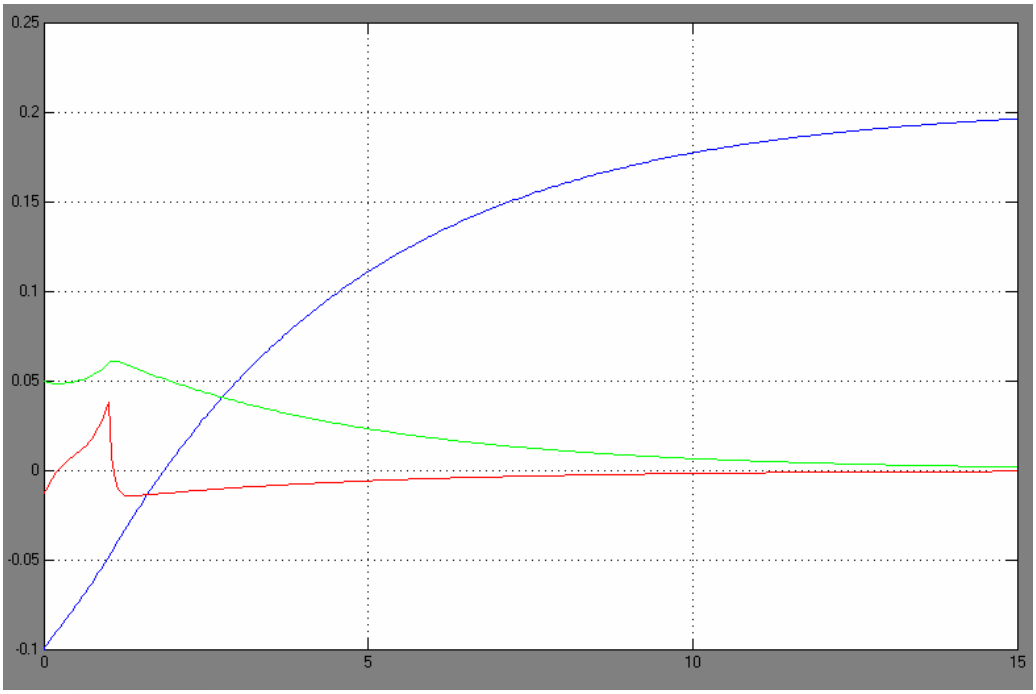


Figura 2.8. Gráfica posición, velocidad, aceleración para corriente 0,5A

Observamos como aceleramos la bola y por lo tanto aumentamos su velocidad inicial. La posición también se ve incrementada sobrepasando el cero, centro de bobina, hasta una posición a 20cm de esta.

Ahora veremos la respuesta si doblamos el valor de la corriente

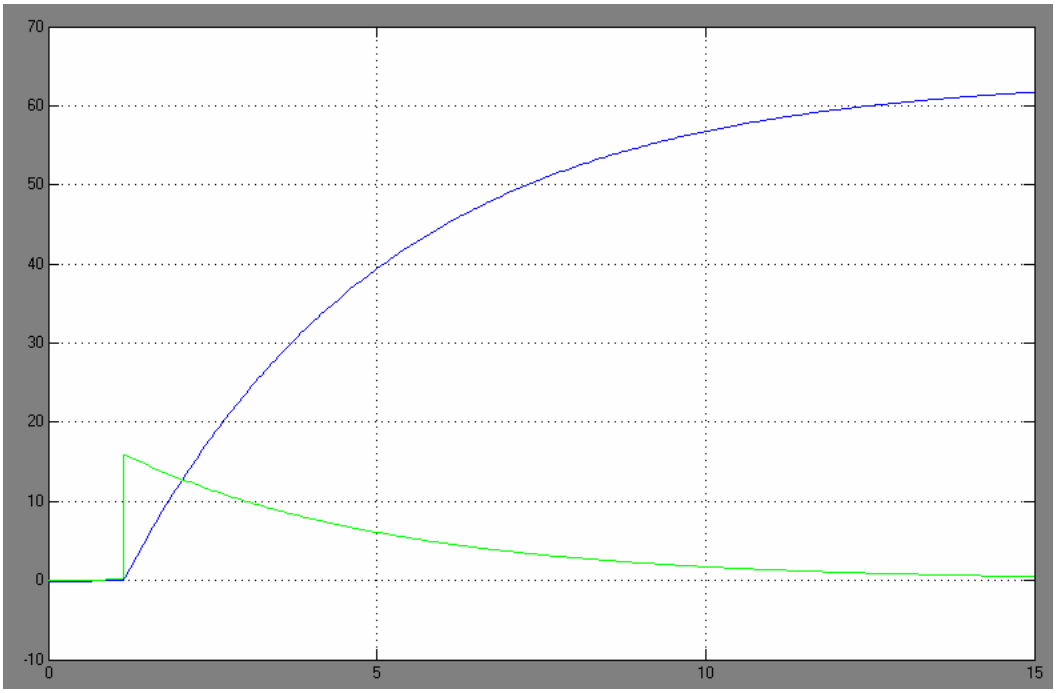


Figura 2.9. Gráfica posición, velocidad, para corriente 1A

Observamos que tanto la velocidad como la distancia recorrida se disparan hasta unos valores muy altos. Seguramente esto se debe a que la fuerza de rozamiento no está bien dimensionada, pero para las mismas condiciones de rozamiento observamos que los resultados simplemente doblando la corriente son mucho mejores de lo que podíamos imaginar.

Pero si nos fijamos en la grafica para corriente igual a 0,5A y un escalón de 2seg vemos que sucede algo que no corresponde con la realidad.

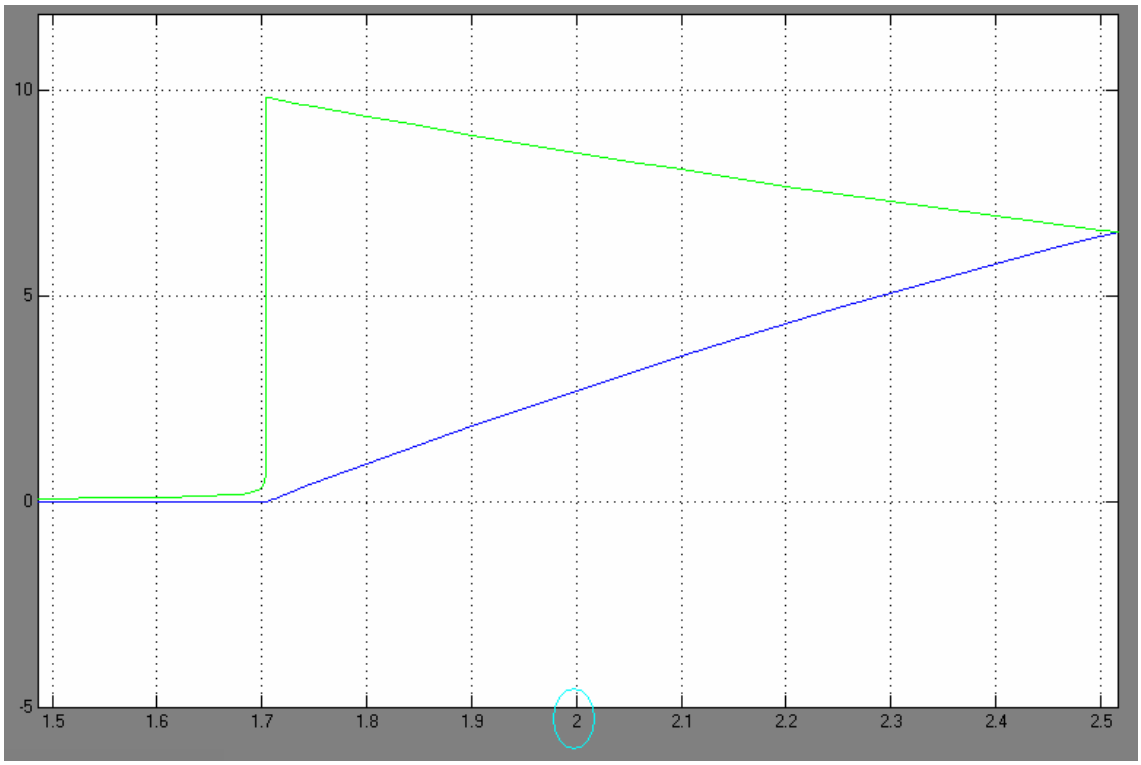


Figura 2.10. Gráfica posición, velocidad, con escalón 24V 2 segundos

En la gráfica anterior observamos que posición de la bola alcanza el centro de la bobina antes de que la bobina se desconecte a los 2 segundos. Nosotros tendremos en la maqueta bobinas toroidales que generan el campo magnético en el centro de su núcleo.

Si la bola atraviesa el centro de la bobina y esta aún no está desconectada, la fuerza de la bobina frenará o detendrá la bola. Dicho de otra manera la fuerza magnética ha de cambiar de signo cuando pasa por el centro de la bobina.

Esto nos forzó a buscar un nuevo modelo. Pero antes teníamos que determinar qué forma debía de tener la fuerza magnética de una bobina toroidal.

La fuerza magnética pues ha de seguir la forma de una función de este tipo:

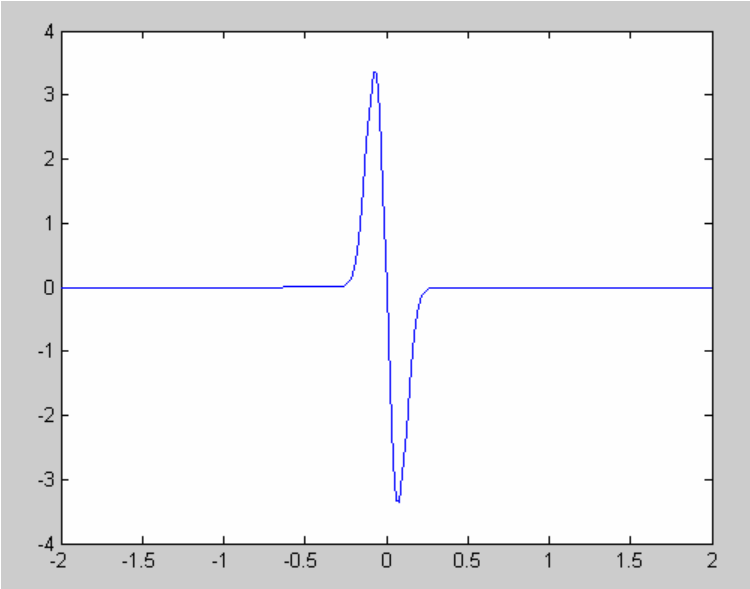


Figura 2.11. Función a seguir de fuerza magnética

Para obtener la función, primero tenemos que determinar qué función sigue la forma que queremos. Esta función tiene que ser del tipo:

$$Fun = -a \cdot \sin(x) \cdot \exp(x^2 \cdot b)$$

Después de hacer varias pruebas la función con la que trabajaremos para hacer las simulaciones es la siguiente:

$$Fun = -40 \cdot \sin(x) \cdot \exp(x^2 \cdot 100)$$

Esta ecuación nos proporciona la grafica anterior, y a partir de esta construimos un nuevo modelo de SIMULINK:

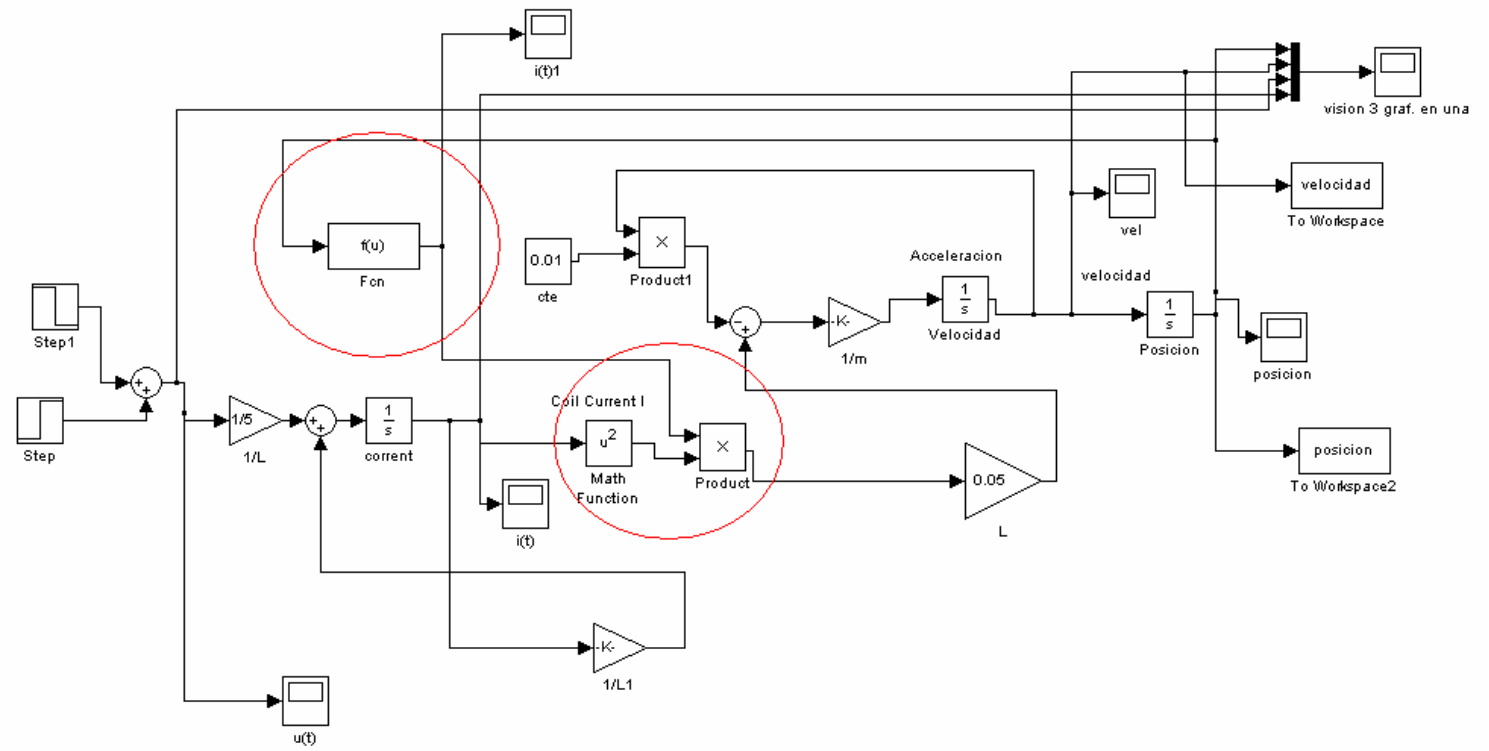


Figura 2.12. Modelo SIMULINK con función bobina toroidal

Se sustituye la función $Fun = \frac{1}{x}$ por la nueva función $Fun = -40 \cdot \sin(x) \cdot \exp(x^2 \cdot 100)$. De esta manera obtenemos en la salida delante de un escalón de 25 segundos de duración:

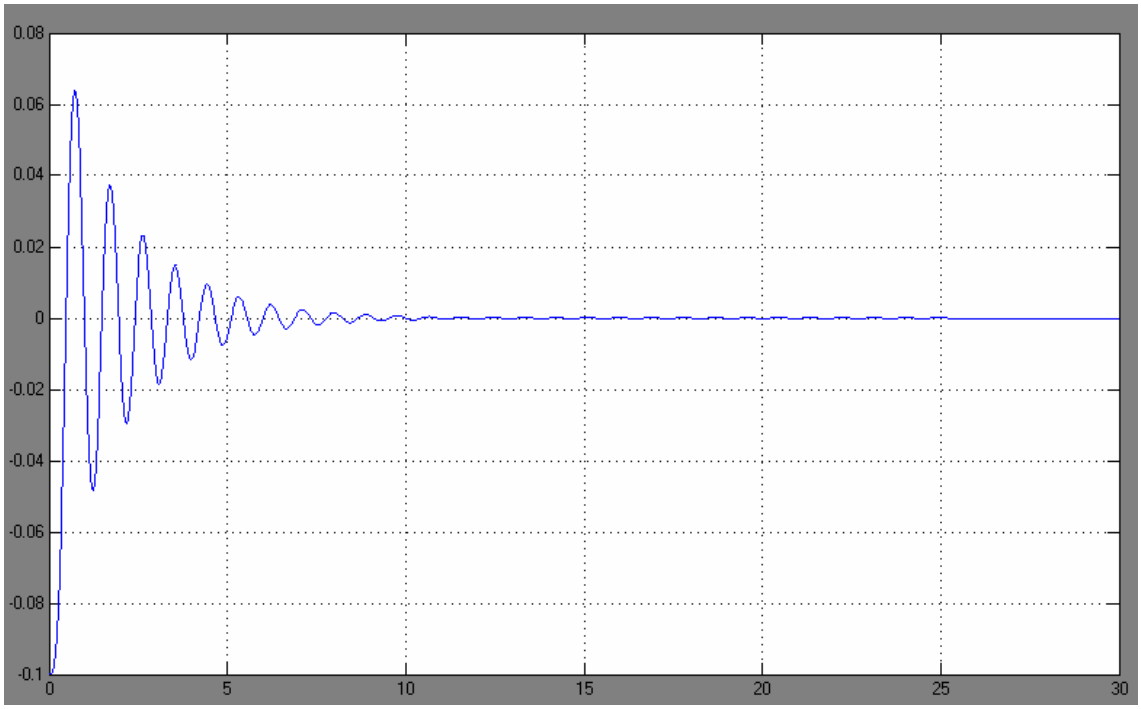


Figura 2.13. Gráfica posición Vs tiempo

Ahora cuando la bola pasa por el centro de la bobina, y esta no está desconectada la bola se frena, incluso si la fuerza es lo suficientemente grande la atrae hacia el centro provocando un efecto muelle hasta que la bola se detiene en el centro, tal y como vemos

en la gráfica anterior.

En la siguiente figura observaremos la respuesta delante de un escalón de 1 segundo:

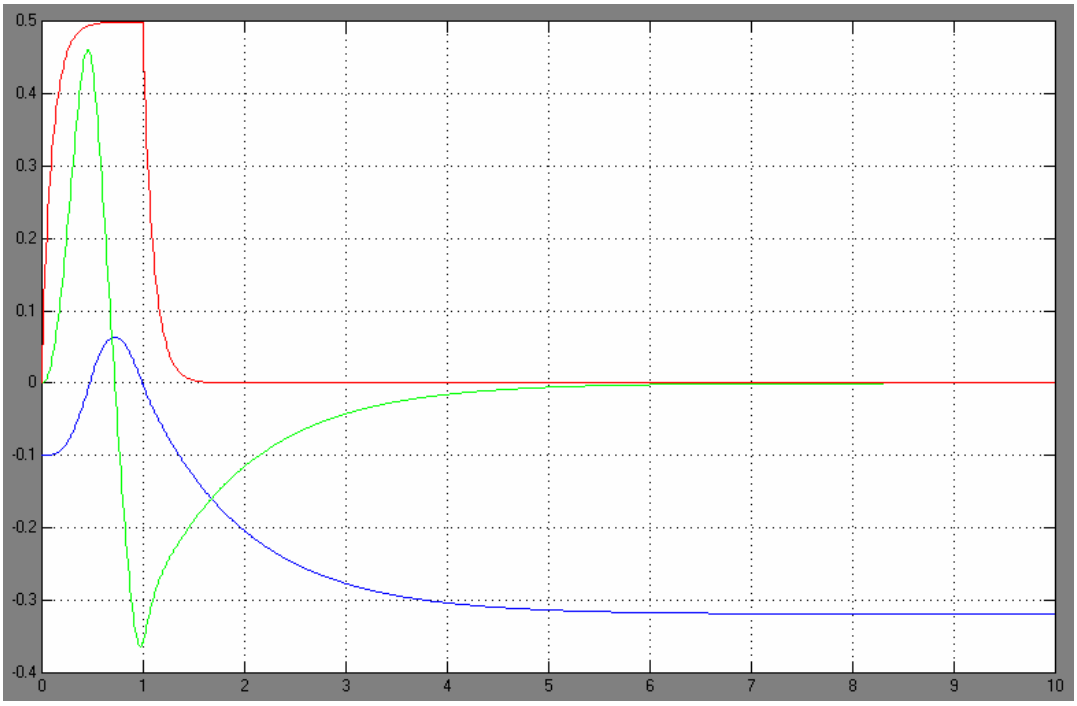


Figura 2.14. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escalón 1seg

Esta respuesta es mucho más real que las que obteníamos anteriormente. La bola aumenta de velocidad, se desplaza hacia el centro de la bobina y lo rebasa. Pero al llegar al centro la bobina aún está imantada y esto provoca que la bobina atraiga la bola a su centro, al desconectar la bobina, la bola se desplaza hacia atrás hasta que se detiene por la fricción. Hay que destacar que la bola parte del reposo, podemos ver en la gráfica como la velocidad toma valores positivos. Pero en el momento en que la bola pasa por el centro imantado de la bobina, la velocidad se empieza a decaer hasta hacerse negativa.

Probaremos de variar el escalón de entrada, simulando diferentes disparos a las bobinas. De esta manera intentaremos aproximarnos, si es posible, a los valores de un disparo optimo para unos valores de velocidad inicial y posición inicial determinados en este caso:

- Posición inicial : -0.1m
- Velocidad inicial: 0 m/s

Probemos con un escalón más corto de 0,7 segundos:

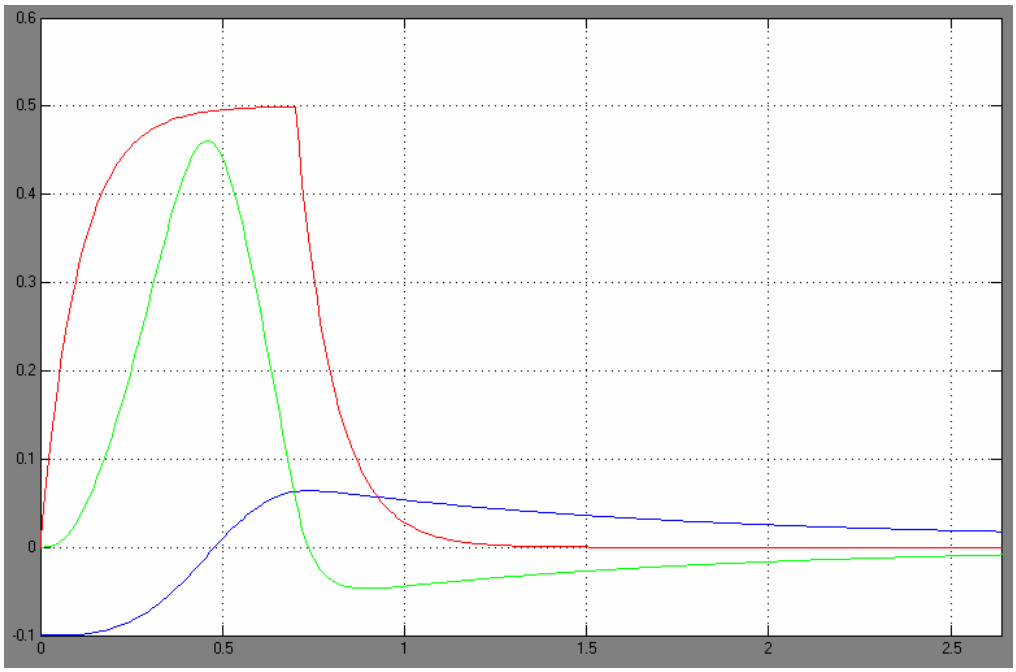


Figura 2.15. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escalón 0,7seg

Bajamos el tiempo de disparo pues claramente el problema es que la bola pase por el centro de la bobina con la bobina alimentada. Conseguimos una mejor respuesta, la bola sobrepasa la bobina y se detiene a unos 2,5cm de esta. Pero seguimos observando que la bobina se desconecta después de que la bola pase por el centro de la bobina y sigue frenando la bola. La velocidad decrece después del paso por el cero de la bola y sigue llegando a valores negativos, por lo tanto la bola retrocede. Este disparo es mejor que el anterior, pero dista mucho de ser un disparo óptimo
Según la gráfica la bola pasa por el cero un poco antes de 0,5 segundos. Sabemos que cuanto más cerca del 0 está la bola más grande se hace la fuerza magnética. Es una relación exponencial, de manera ideal podemos decir que si nos aproximamos al cero la fuerza tiende a infinito, pues la fuerza es inversamente proporcional a cuadrado de la distancia:

$$F_e = L \cdot \frac{i^2(t)}{x^2}$$

La duda que se nos plantea es si existe un punto en que aún cortando la alimentación de la bobina justo después del paso de la bola, esta gana más velocidad que apurando el corte justa.

Probemos con un escalón más corto de 0,49 segundos:

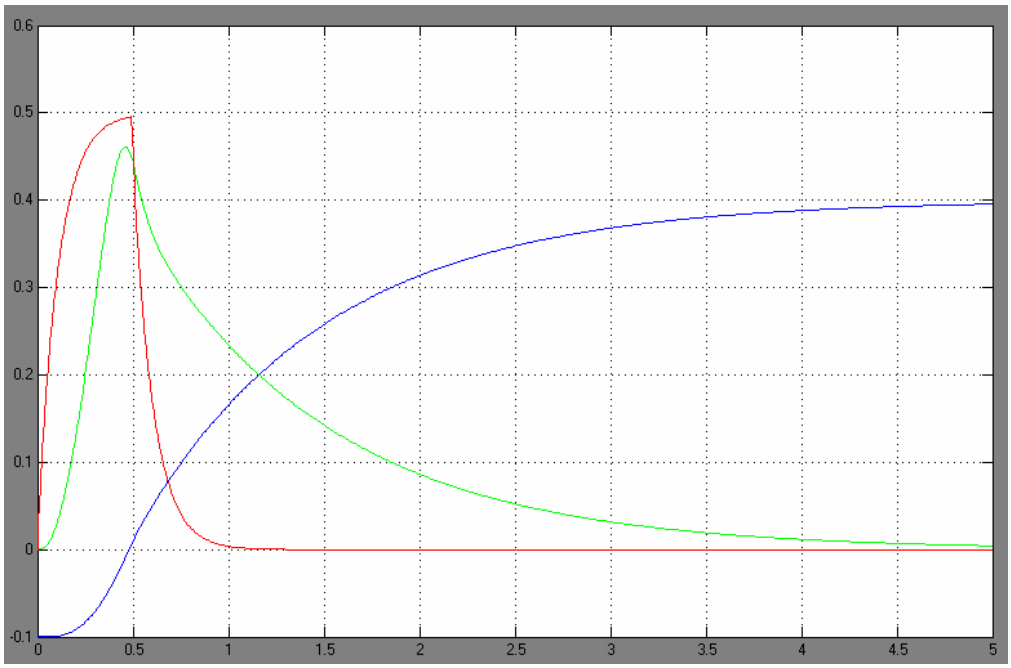


Figura 2.16. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escalón 0,49seg

En este caso desconectamos la bobina justo después del paso de la bola por cero. Claramente este es un disparo mucho mejor que los anteriores. La velocidad no se hace negativa en ningún momento, la bola se desplaza 0,39m.

Ahora bajaremos el disparo un poco más, lo suficiente para desconectar la bobina antes del paso de la bola por el centro. Así veremos si el disparo óptimo está antes del paso por 0 o antes.

Probemos con un escalón más corto de 0,43segundos:

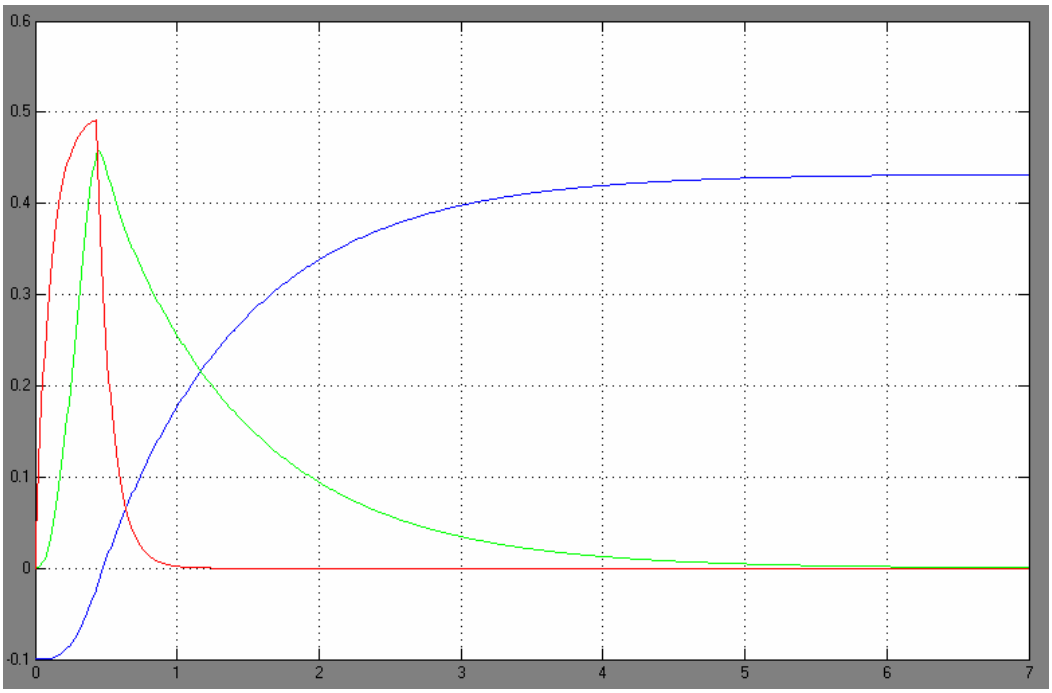


Figura 2.17. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escalón 0,43seg

Con este disparo la respuesta del sistema es mejor que con el anterior ya que la distancia recorrida por la bola es un poco mayor, con lo que nos aventuramos a decir que el disparo óptimo se encuentra desconectando la bobina antes del paso de la bola por el centro de esta.

Nos fijamos que aún que la bobina ya está desconectada, cuando la bola pasa por el centro, aún queda un copo de intensidad en la bobina dado que la bobina tiene un tiempo de descarga. Entonces la pregunta es si esta pequeña imantación afectara lo suficiente como para frenar la bola. Si buscamos hacer un disparo más corto para ver si esto mejorara la respuesta o no.

Una conclusión que podemos dar por cierta, y que más adelante en la fase de programación nos simplificará la tarea es la siguiente: Cuando empezamos con la idea de buscar disparos óptimos pensamos en qué momento se tenía que encender la bobina y en qué momento se tenía que desconectar para tener un disparo óptimo. Ahora podemos afirmar que el momento de conexión es superfluo, simplemente tiene que garantizar la máxima imantación de la bobina. Donde recae toda la acción de control es en la desconexión de la bobina y no en el disparo en sí. Por lo tanto la conexión de la bobina se tiene que realizar cuando la bola no esté pasando por el centro de una bobina y con la suficiente antelación para que la bobina esté completamente imantada, un buen punto seria en el paso de la bola por los sensores que calculan la velocidad de paso, pues se encuentran en el recorrido entre una bobina y la siguiente.

Probemos con un escalón más corto de 0,38 segundos:

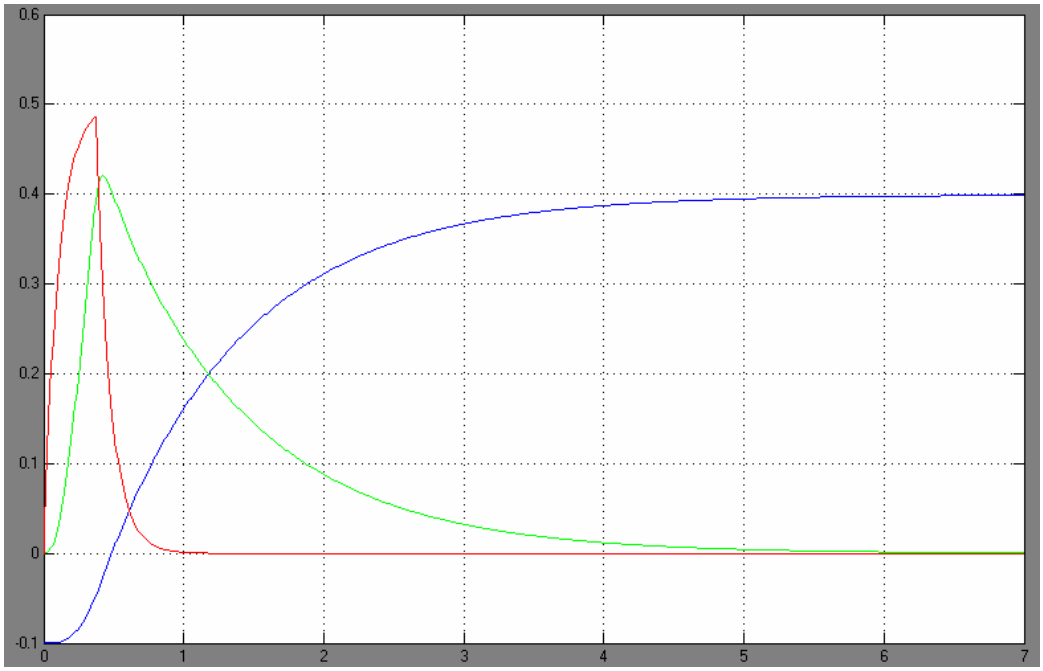


Figura 2.18. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escalón 0,38seg

En esta simulación podemos ver que aún qué no ha desaparecido del todo la imantación de la bobina, al alejarnos del centro para desconectar la bobina perdemos distancia y velocidad respecto el disparo anterior. Si nos fijamos la respuesta de la bola es muy parecida que en el disparo de 0,49 segundos, por lo tanto en estos 0,11 segundos que van entre los disparos de 0,49 y 0,38 tenemos un disparo óptimo muy próximo a 0,43 segundos. También nos fijamos en que hay un margen de 0,43 ±0,2 segundos donde la respuesta es muy similar y que es en esta zona donde tenemos un conjunto de disparos aceptables, por no decirles óptimos.

Todas las simulaciones anteriores se han realizado con una velocidad inicial nula, pero que pasa si la velocidad inicial es diferente, ¿seguiremos teniendo el mismo margen de disparos aceptables?

El siguiente paso, pues, es modificar la velocidad y buscar de nuevo un disparo optimo para comprobar si coincide con el anterior o no. Creemos que no tiene que coincidir pero vamos a buscar el disparo óptimo para estar seguros, pues si coincidieran se nos simplificaría mucho la programación de la futura maqueta.

Para las siguientes simulaciones vamos a fijar la velocidad inicial en 0,2 m/s

La distancia a centro de bobina no la modificamos pues tenga el valor que tenga en la realidad, en la futura maqueta, será una constante.

Probemos con un escalón más corto de 0,49 segundos:

Dado que este es el primer valor que nos daba una respuesta positiva en las simulaciones con velocidad inicial nula. Por lo tanto escogemos este disparo como punto de partida y así tendremos una primera valoración

Dependiendo del resultado de la simulación seguiremos buscando los mismos puntos de disparo que con velocidad inicial nula o iremos a buscar nuevos puntos de disparo óptimos.

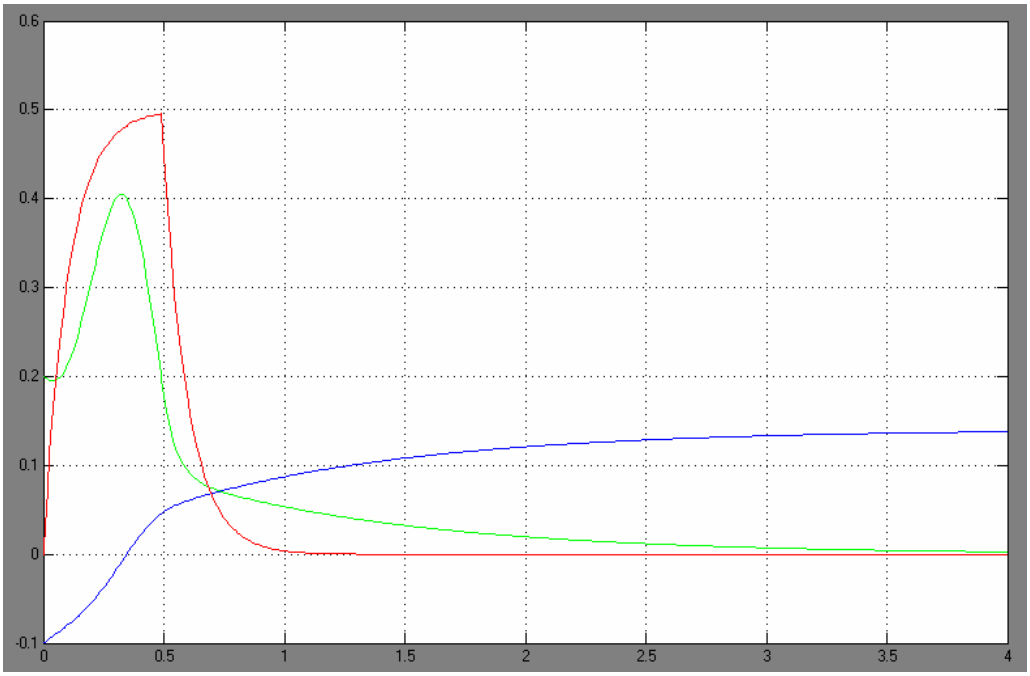


Figura 2.19. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escal. 0,49 s y Vel_0 0,2 m/s

La velocidad inicial hace que la bola pase antes por el centro de la bobina, esto provoca que cuando cortamos la bobina en 0,49 segundos frenamos la bola de manera considerable como podemos ver en la figura.
Podemos deducir que el disparo óptimo tendrá que ser un disparo más corto que antes. Por lo que los disparos óptimos variarán en función de la velocidad de la bola.

Para la siguiente simulación ya no buscaremos el mismo valor de disparo optimo que con velocidad inicial nula, si no que empezaremos buscando un valor próximo al paso por el centro de la bobina que puede estar aproximadamente en 0,3 segundos.

Para no repetir todo el proceso damos directamente el valor de del disparo optimo encontrado que es 0,30 ±0,2 segundos

Escalón de 0,3 segundos:

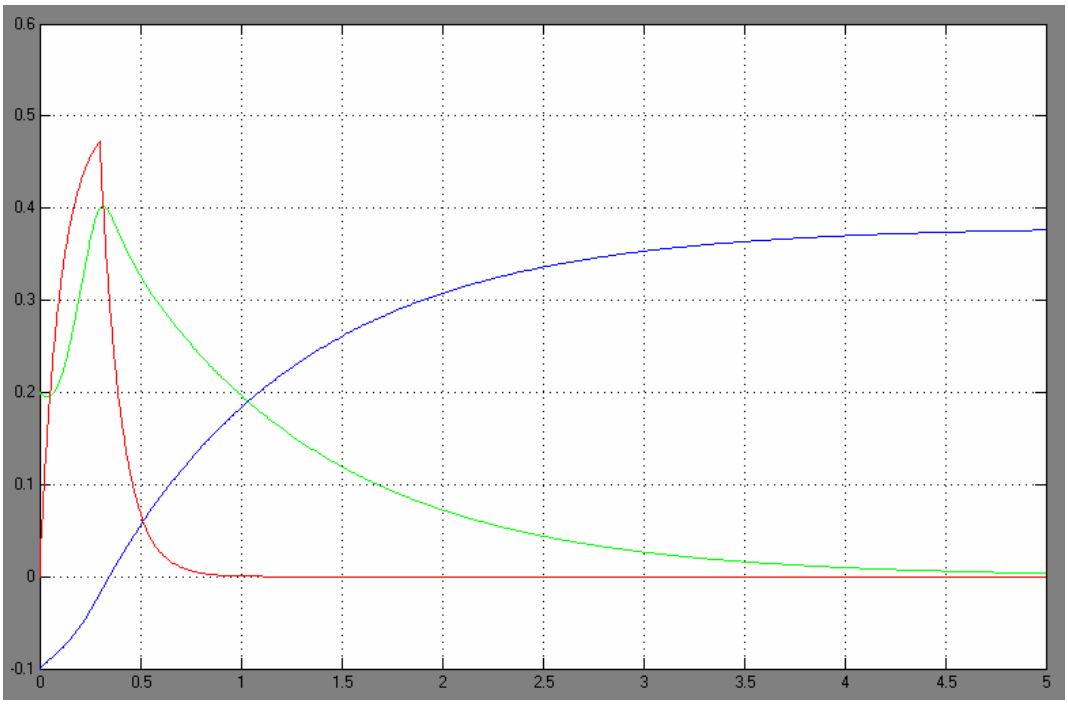


Figura 2.20. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escal. 0,3 s y Vel_0 0,2 m/s

En esta simulación ocurre algo con lo que no contábamos en principio. El disparo es óptimo para una velocidad inicial de 0,2 m/s, pero la respuesta es pero que la del disparo optimo para velocidad nula.

Se esperaba una mejor respuesta, mas velocidad máxima y más distancia recorrida, pero esto no sucede. La explicación la podemos encontrar en dos factores.

Por un lado con un disparo tan corto la bobina no alcanza su valor máximo que está establecido en 0,5A. Esto penaliza bastante al sistema pues recordemos que el valor de la fuerza magnética es directamente proporcional al cuadrado de la corriente.

Por otro lado al ser un escalón más corto y teniendo en cuenta que la duración del escalón de alimentación también es directamente proporcional al valor de la fuerza magnética tenemos otro factor que penaliza en el rendimiento del sistema.

Como se ha explicado anteriormente nosotros no tenemos problema en la conexión de la bobina, por lo tanto podemos establecer que la bobina ya está en funcionamiento cuando se acerca la bola. De esta manera eliminamos los dos factores que nos penalizan.
Anteriormente ya se estableció que las bobinas se activarían en algún punto del recorrido entre bobinas para simplificar la programación de la maqueta. Ahora además, se tiene que realizar de esta manera para eliminar posibles factores derivados de la velocidad de entrada a bobinas de la bola.

Por lo tanto hay que modificar la simulación. Mantendremos la velocidad inicial de la bola a 0,2 m/s pero activaremos la bobina un segundo antes. Para poder hacer esto tenemos

que modificar la posición inicial de la bola que pasara de valer -0,1m a valer -0,3m simulando que la bola viene de más atrás, de esta manera cuando la bola llege a -0,1m la bobina ha tenido tiempo de cargarse completamente.

La nueva simulación será por lo tanto 1 segundo más 0,3 segundos que era el disparo óptimo anteriormente encontrado

Escalón de 1,3 segundos:

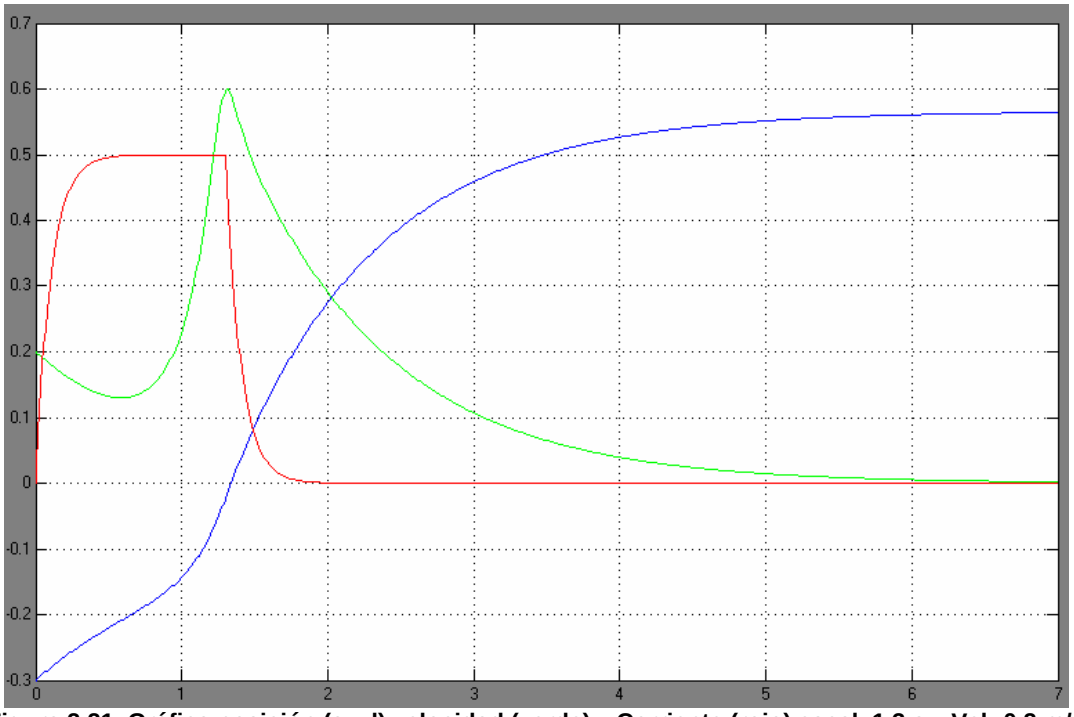


Figura 2.21. Gráfica posición (azul) velocidad (verde) y Corriente (rojo) escal. 1,3 s y Vel_0 0,2 m/s

De esta manera observamos que para la misma velocidad añadimos un offset de -1seg al disparo óptimo y vemos como mejora la respuesta del sistema. Con velocidad nula no podemos añadir este offset pues al estar la bola inmóvil y dentro de la zona de influencia de la bobina, si añadimos tiempo al pulso solo conseguiremos frenar la bola en su paso por el centro de la bobina o retenerla en el.

Ahora si que obtenemos una respuesta como la que imaginábamos antes de empezar a hacer la simulación con la bola con velocidad inicial mayor que cero. Tanto el parámetro de la velocidad como el de la distancia recorrida es superior que con la bola con velocidad inicial nula.

De todas maneras podemos darnos cuenta que en la maqueta hay un límite de distancia entre bobina y bobina y por lo tanto habrá un límite de velocidad a partir del cual ya no será posible cargar al máximo las bobinas y por lo tanto seguir acelerando la bola dentro del tubo.

Otra conclusión es que para cada velocidad existe un punto óptimo diferente. Esto nos complica mucho la programación de la maqueta, pues por definición existirán infinitas

velocidades e infinitos puntos óptimos.

No será objeto de este proyecto encontrar todos esos puntos óptimos, sino encontrar una serie de ellos y a partir de estos intentar hacer el mejor control posible.

En la siguiente simulación se conectan dos bobinas en cascada y se pasan los valores de velocidad y posición como salidas de la primera bobina y se pasan como valores iniciales de la segunda.

Se ha realizado esta simulación para demostrar como una segunda bobina es capaz de realizar un disparo aprovechando la velocidad de salida de la primera bobina.

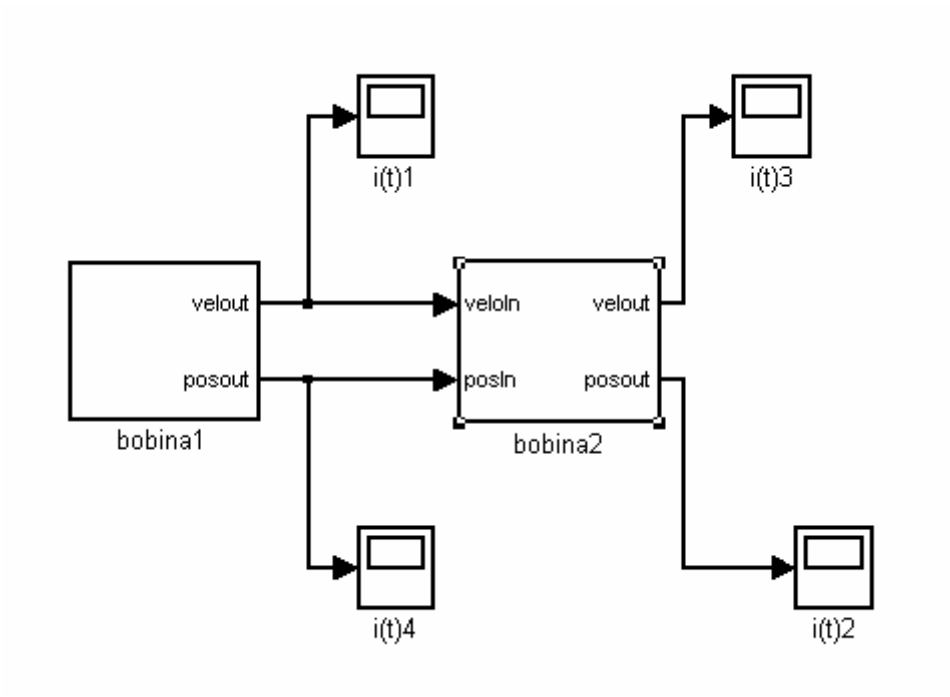


Figura 2.22. Modelo SIMULINK dos bobinas en cascada

Para realizar las conexiones entre bobinas se han realizado algunas modificaciones en los modelos existentes de las bobinas.

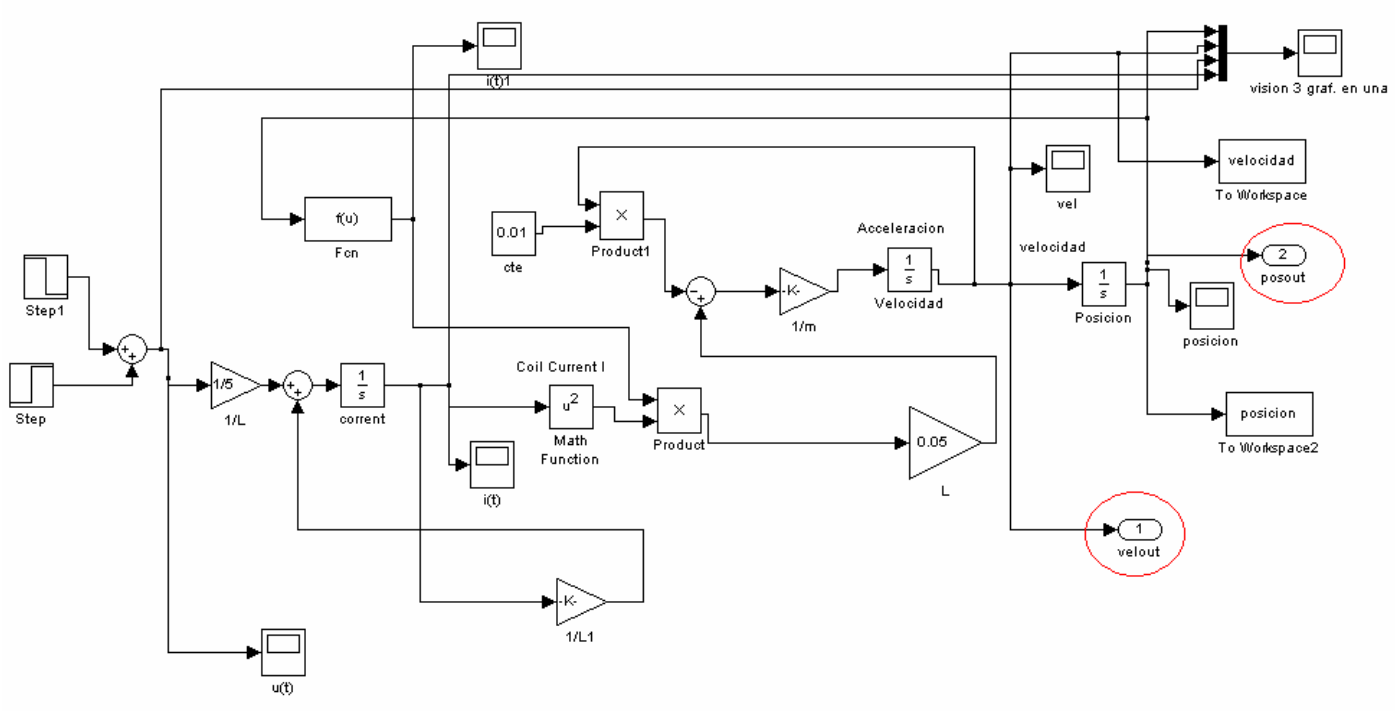


Figura 2.23. Cambios en modelo primera bobina

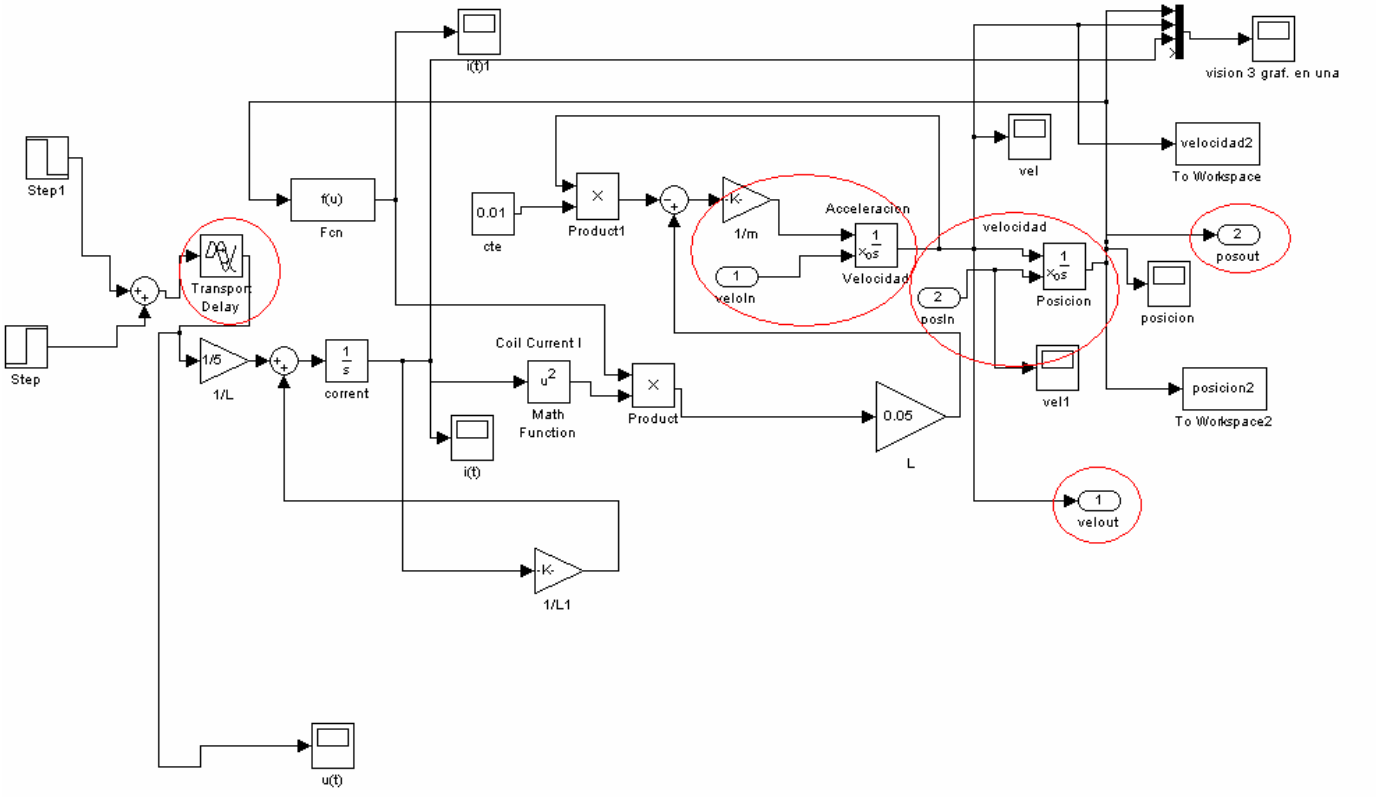


Figura 2.24. Cambios en modelo segunda bobina bobina

Se han utilizado dos puertos de salida para pasar los valores de salida de velocidad y posición para poder utilizarlos como condiciones iniciales de la segunda bobina, en la cual se han realizado las siguientes modificaciones respecto el modelo original.

Se ha añadido un nuevo terminal en los integradores. Este terminal es el de condiciones

iniciales externas, de esta manera conseguimos que los valores de velocidad y posición de salida de la primera bobina sean las condiciones iniciales de la segunda bobina,

También se ha añadido un 'Delay' en el pulso de entrada para poder realizar los disparos en tiempos diferentes y así simular diferentes tiempos de disparo de cada bobina.

Cogemos un disparo cualquiera en la primera bobina, en este caso con un escalón de 0,4 segundos con velocidad inicial nula.

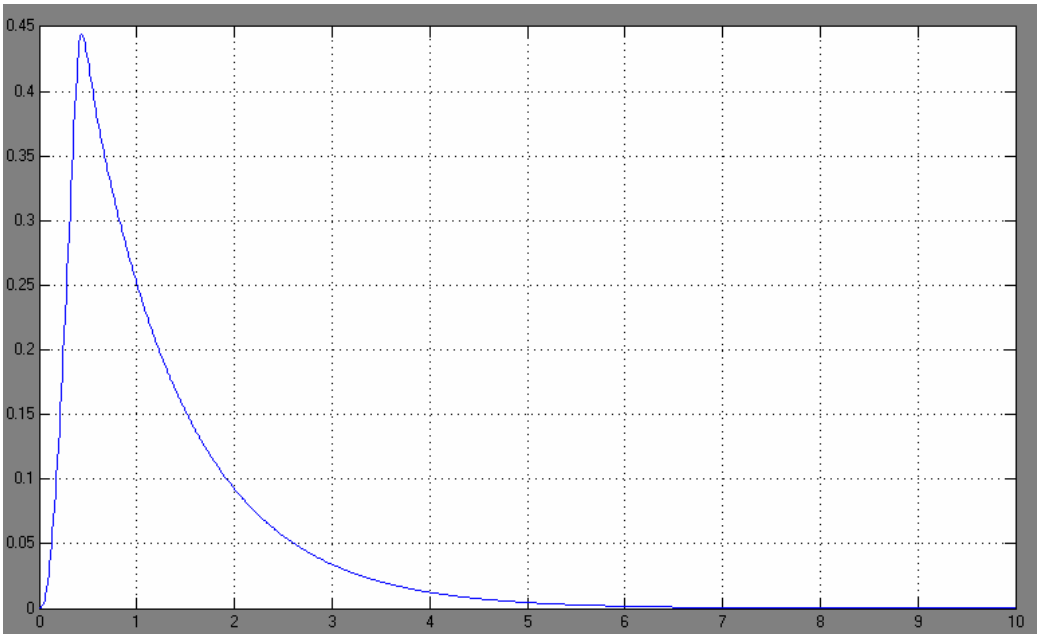


Figura 2.25. Gráfica velocidad de salida primera bobina

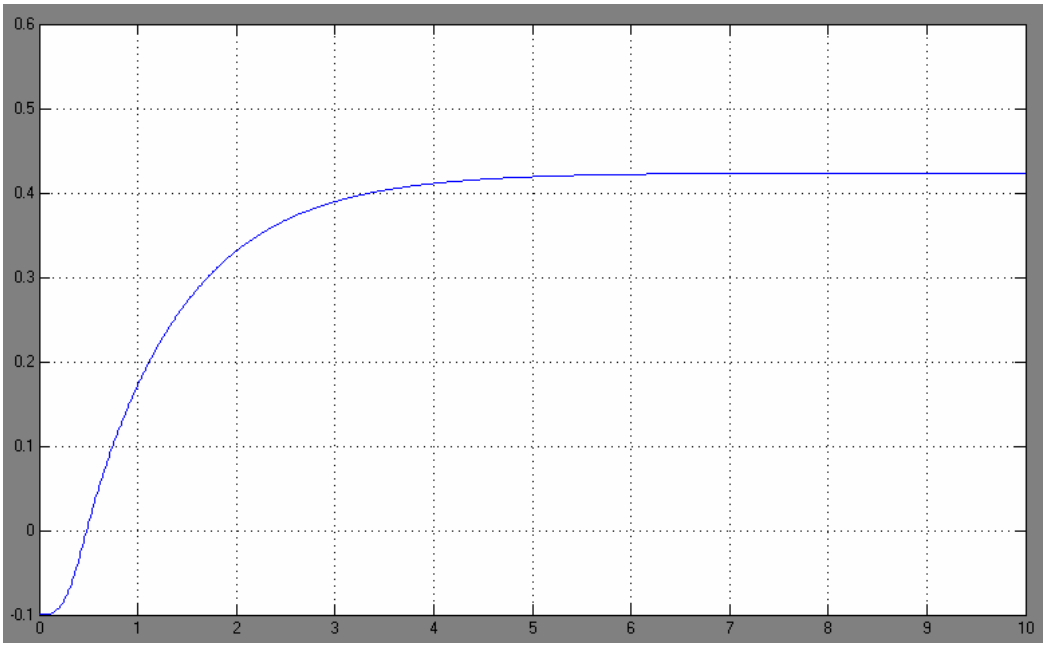


Figura 2.26. Gráfica posición de salida primera bobina

En la segunda bobina tenemos los siguientes resultados con un 'delay' entre disparaos de un segundo.

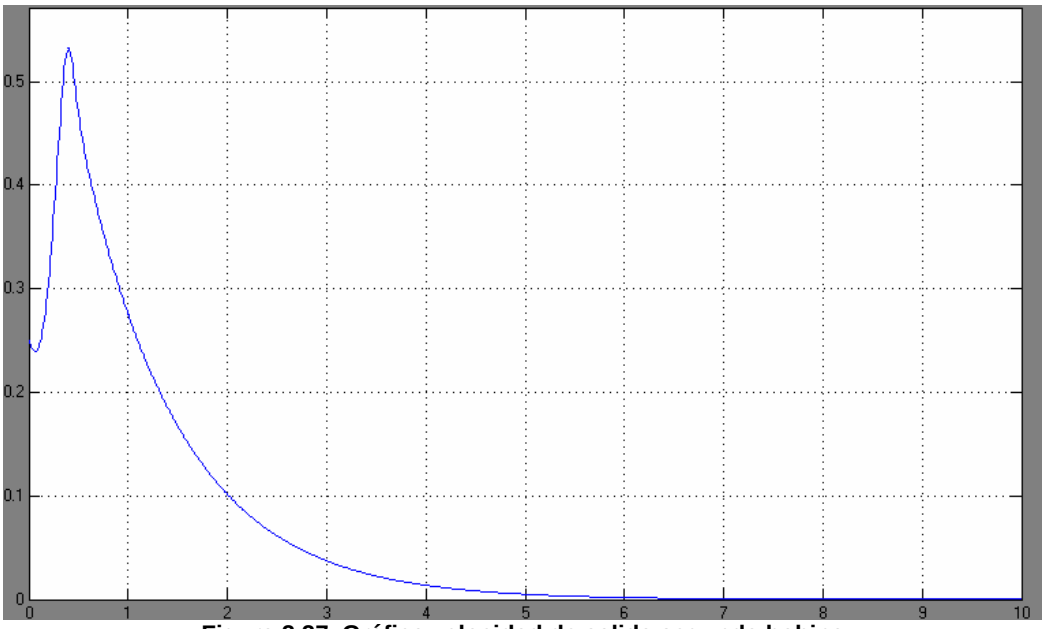


Figura 2.27. Gráfica velocidad de salida segunda bobina

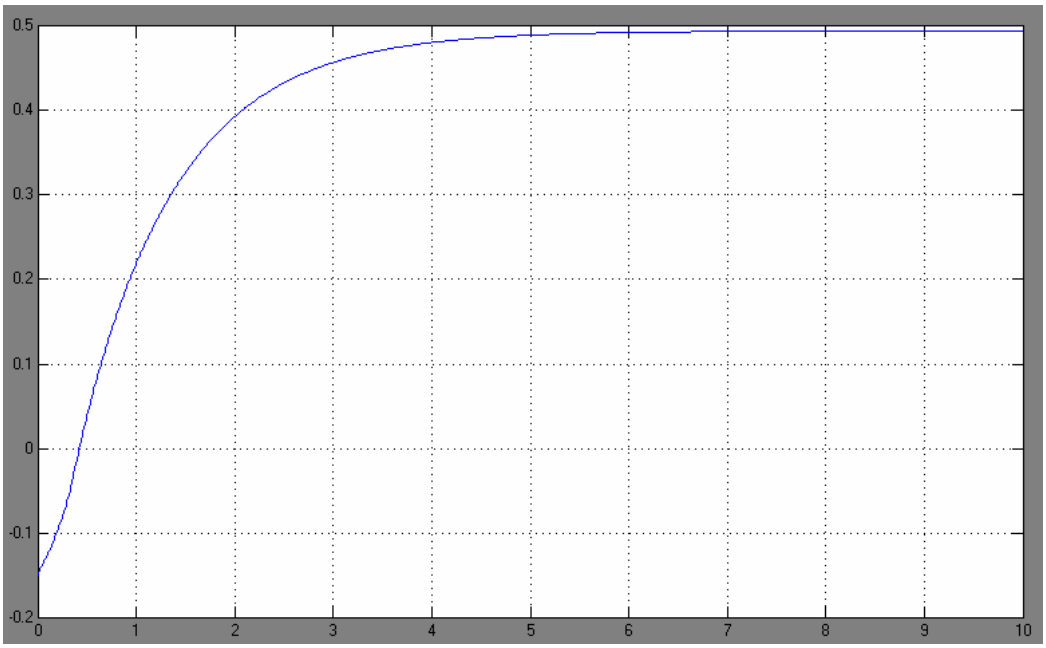
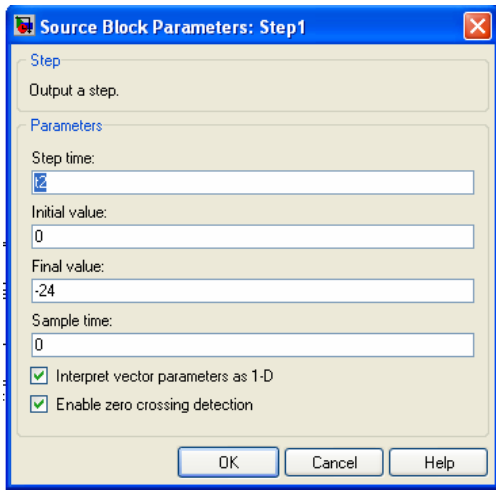


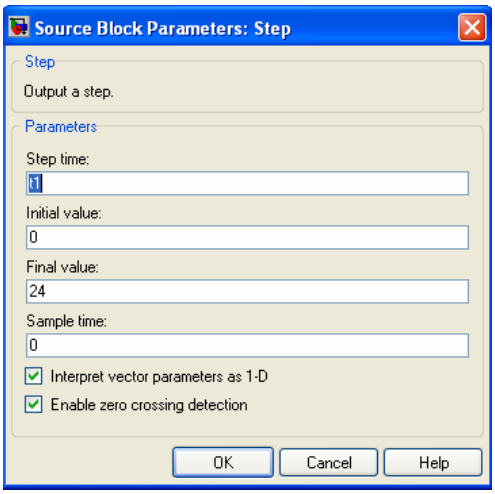
Figura 2.28. Gráfica posición de salida segunda bobina

Vemos como partiendo de los valores de salida de la primera bobina como valores de entrada de la segunda, podemos simular como una bobina desplaza la bola hasta la siguiente y esta vuelve a disparar la bola aumentando su velocidad máxima.

Para obtener puntos con las simulaciones utilizaremos la función de MATLAB **fminsearch**, la cual dada una función de varias variables, y un intervalo, determina el valor mínimo de una función. Esta función es muy utilizada para la optimización de sistemas no lineales, como se presuponemos que será el nuestro.
Esta función recorrerá un intervalo de disparos para una velocidad y nos devolverá el valor mínimo de la distancia. Para que nos devuelva el valor máximo de la distancia pasaremos este parámetro en negativo.

Para poder hacer una búsqueda los parámetros de tiempo del pulso de entrada del modelo de SIMULINK pasaran a ser dos variables (t1 y t2)





Estas dos variables irán variando su valor y de esta manera para un velocidad fijada podremos encontrar su disparo óptimo.

Como fijamos la velocidad el modelo que utilizaremos en el de una sola bobina, y sobre esta iremos variando el valor de la velocidad de entrada.

Para conseguir esto creamos los estos dos programas en MATLAB

Este programa lanza la simulación de SIMULINK 'prova8' y le pasa los valores de t1 y t2 que le pasamos con el vector 'tiempos'.

```
function v = fun2manel( tiempos )
 global t1;
 global t2;
 t2
if t2<t1
    v=10000;
else
t1=tiempos(1);
t2=tiempos(2);
sim('prova8');
v=-posicion^2
end
end
```

Este programa recorre los diferentes valores de pulso de entrada buscando el disparo optimo para cada uno de ellos. Abre un archivo salida.txt donde nos guardara los resultados de la búsqueda.

```
function j=pruebav2()
```

```
global t1;
global t2;
global velocidad;
global tiempos;
global x;
velocidad=1;
tiempos=[0,0];

for t1=0:2
   fi=fopen('salida.txt','a');
   for t2=0:3
      if t2>t1
      tiempos=[t1,t2];
      j=fminsearch(@fun2manel,tiempos);
      fprintf(fi,'%f ',j);
      x=fun2manel(j);
      fprintf(fi,'%f',x);
      fprintf(fi,'\n');
      end
    end
   fclose(fi);
end
```

En los anejos del proyecto se incluyen tablas de los resultados de estas simulaciones.

2.1 Conclusiones de las simulaciones

El objetivo de las simulaciones era comprobar que se podía realizar el proyecto. Sabíamos que al empezar a hacer las simulaciones sin saber de qué materiales íbamos a disponer, representaría que los puntos de disparo obtenidos no nos iban a servir en la maqueta real. Pero si que nos ayudo a saber cómo teníamos que proceder para encontrar esos puntos. Por supuesto no sirvió para comprobar que el proyecto era viable y se podría realizar.

Todas las simulaciones se realizaron en corriente continua, aunque nuestra intención era montar dos controles, uno para continua y otro para alterna. Sabíamos que en alterna podíamos tener algún problema con la fase de la alimentación de entrada, pero ganábamos en fuerza magnética pues podíamos alimentar las bobinas a 230V y esto nos generaba una mayor intensidad. Dado que las simulaciones partían de datos no reales no doblamos el trabajo de simulación haciendo las simulaciones en alterna pues se considero que los resultados eran lo suficientemente validos para poder empezar a realizar la construcción de la maqueta y la selección de los materiales de esta.
Por supuesto otra conclusión que sacamos, es que tendríamos que sacar los puntos óptimos uno a uno con la maqueta real una vez montada y en funcionamiento. Esto será una tarea larga pues tendremos que realizar lo mismo que realiza el fminsearch de MATLAB pero sobre la maqueta real, disparo a disparo y velocidad a velocidad.

3. CONSTRUCCIÓN DE LA MAQUETA

3.1 Descripción de funcionamiento:

El funcionamiento de la maqueta se basa en la aceleración de una bola metálica a partir del campo magnético generado por una serie de bobinas situadas en puntos determinados de la zona de rodadura.

A continuación se muestra un esquema simple de la maqueta con la disposición de los diferentes elementos:

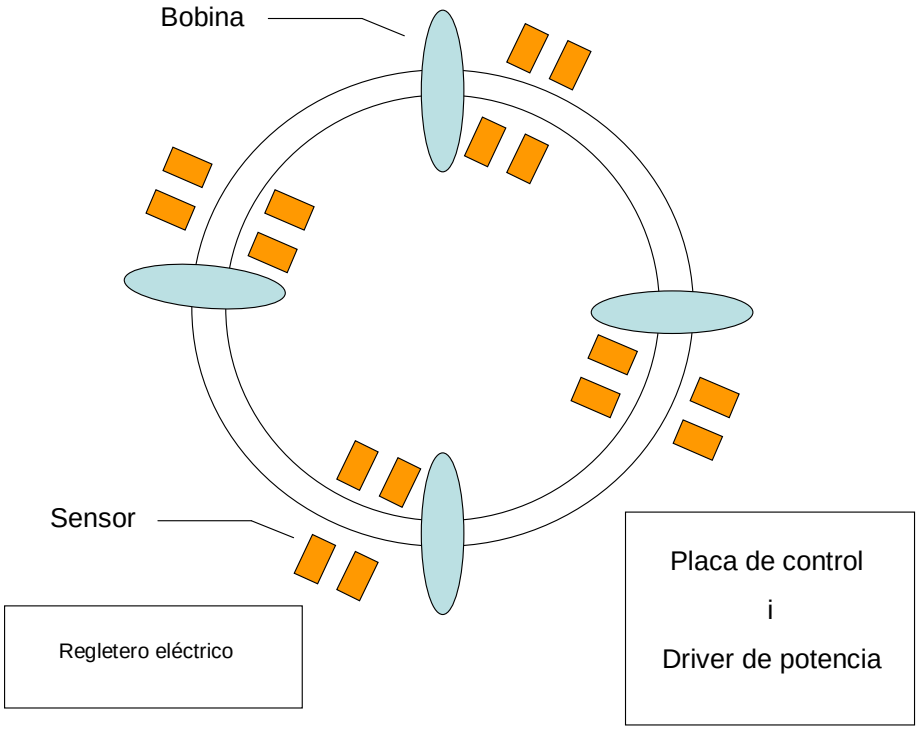


Figura 3.1.1. Esquema de la maqueta

La bola parte con velocidad inicial y al cortar una pareja de sensores se calcula la velocidad con la que pasa la bola. Con esta velocidad se calcula el corte óptimo de la alimentación a la bobina para dejar de generar el campo magnético que acelera la bobina y así evitar una parada en la velocidad de la bola.
Con esto se pretende una aceleración en la bola y con ello aumentar su velocidad o mantenerla.
En los puntos siguientes se indicará el proceso seguido para escoger el material adecuado de nuestra aplicación así como el material definitivo montado en la maqueta.

3.2 Selección de materiales

3.2.1 Base de la maqueta

La base seleccionada para soportar la maqueta es una plancha de madera conglomerada de 1 centímetro de grosor y de 90 centímetros de ancho por 90 centímetros de largo.
La elección de este material se debe a que no es conductor, no genera interferencias en los sensores ni en el campo magnético de las bobinas y nos proporciona la suficiente rigidez metálica para soportar el resto de elementos de la maqueta.

3.2.2 Zona de rodadura

La zona de rodadura seleccionada es un tubo PVC de 25 milímetros de diámetro.
La elección de este material se debe a que es un material transparente que permite el paso del haz de luz de los sensores permitiendo la detección de la bola y no realizando interferencias en el campo magnético de las bobinas.
Por otro parte este material es lo suficientemente resistente para aguantar las altas temperaturas generadas por las bobinas sin deformarse.

3.2.3 Sensores

3.2.3.1 Selección del sensor

Se ha optado por la instalación de sensores de tipo barrera ópticos debido a que detectan el paso de la bola por el tubo de PVC sin que este genere interferencias en el sensor, ni en el campo generado por las bobinas
.
Pero hasta llegar a los sensores que hay montados actualmente en la maqueta hemos realizado muchas pruebas para intentar seleccionar el mejor sensor calidad precio. A continuación se muestran todos los sensores que se han probado y cuáles son los problemas que han presentado por lo que han sido descartados.

Relación de sensores utilizados:

1. Honeywell HOA1180:

    - Tipo: Interruptor reflector
    - Distancia de detección: 12,7 mm
    - Tipo salida: NPN open collector
    - Tiempo de ascenso/descenso: 75 microsegundos
    - Tensión de alimentación: 1,6 V
    - Corriente consumo: 20 mA

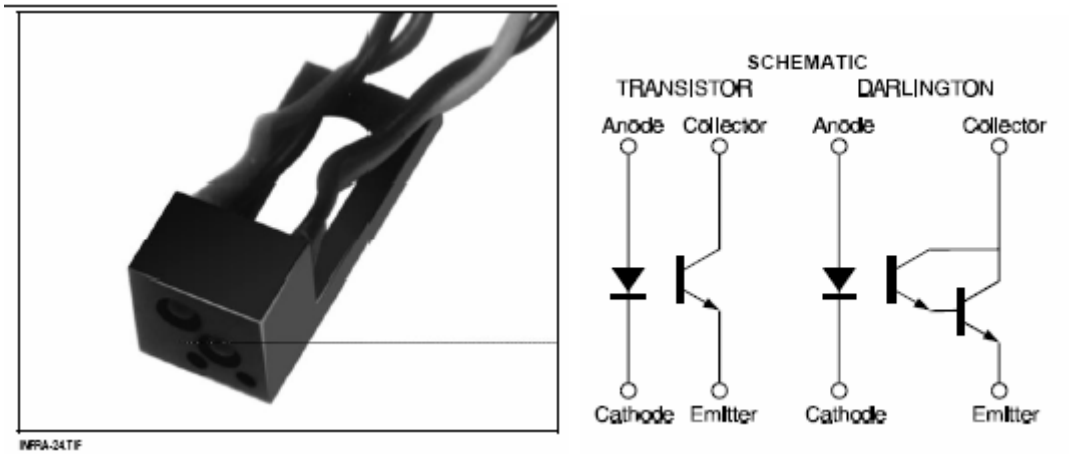


Figura 3.2.3.1.1. Sensor Honeywell HOA1180

Para limitar el consumo de este tipo de sensores es necesario colocar una resistencia en serie. El cálculo de la resistencia a colocar en serie se muestra a continuación:

Tensión de alimentación: 5V (fijada por nosotros)

$$R = \frac{Vcc - Vsensor}{Isensor} = \frac{5 - 1{,}6}{0{,}02} = 170\Omega$$

Esquema de conexión:

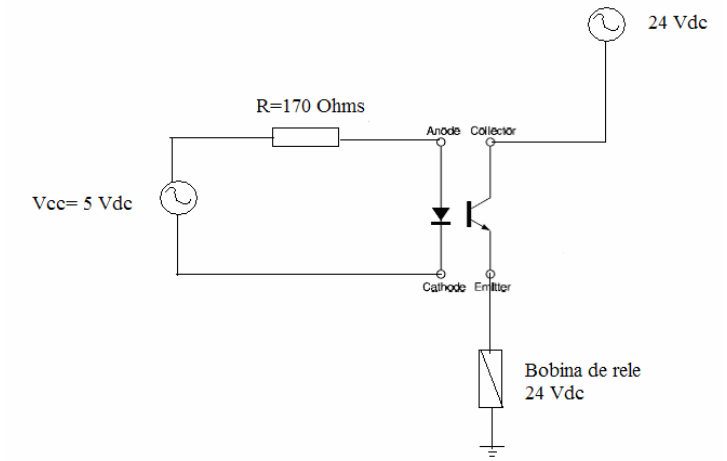


Figura 3.2.3.1.2. Esquema de conexión sensor Honeywell HOA1180

Problemática:

Las fluctuaciones que tenemos en la salida de la fuente de alimentación quema el sensor rápidamente ya que la tolerancia de este es inexistente.

Los dos sensores que se han probado a 5,1 Vdc se han quemado.

Coste del sensor: 14,23 Euros

2. Optek 815W:

- Tipo: Optointerruptor ranurado
- Anchura de la ranura: 9,5 mm
- Profundidad de la ranura: 12,6 mm
- Tiempo de ascenso/descenso: 75 microsegundos
- Tensión de alimentación: 1,6 V
- Corriente consumo: 20 mA

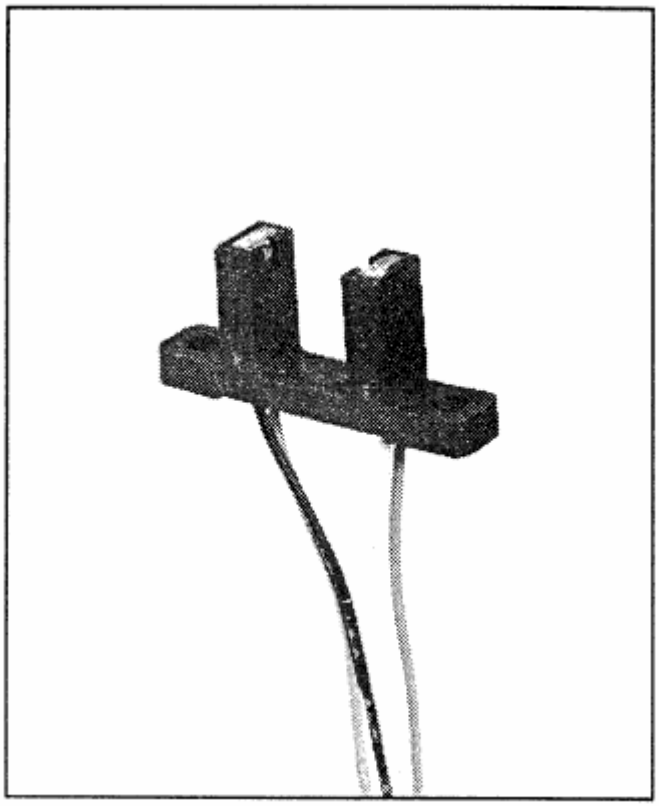


Figura 3.2.3.1.3. Sensor Optek 815W

Para limitar el consumo de corriente de este tipo de sensores es necesaria la instalación de una resistencia en serie. El cálculo de la resistencia a colocar en serie se muestra a continuación:

Tensión de alimentación: 5V (fijada por nosotros)

$$R = \frac{Vcc - Vsensor}{Isensor} = \frac{5 - 1{,}7}{0{,}02} = 165\Omega$$

Esquema de conexión:

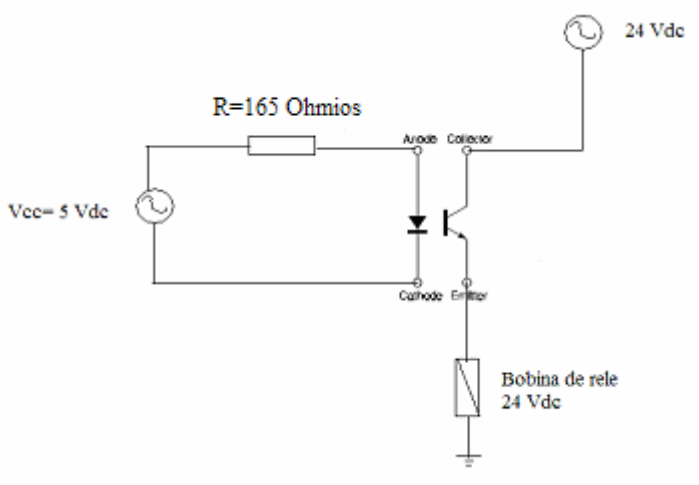


Figura 3.2.3.1.4. Esquema de conexión del Optek 815W

Problemática:

Las fluctuaciones que tenemos en la salida de la fuente de alimentación queman el sensor rápidamente ya que la tolerancia de este es inexistente.
Los dos sensores que se han probado a 5,1 Vdc se han quemado.
Por otro lado las dimensiones de la ranura no nos permite adaptar el sensor al tubo de plástico elegido para hacer circular la bola.

Coste del sensor: 6,72 Euros

3. Honeywell HOAD0150-2:

- Tipo: Optointerruptor ranurado (infrarrojos)
- Anchura de la ranura: 22 mm
- Profundidad de la ranura: 23,5 mm
- Tensión de alimentación: 1,6 V
- Corriente consumo: 20 mA

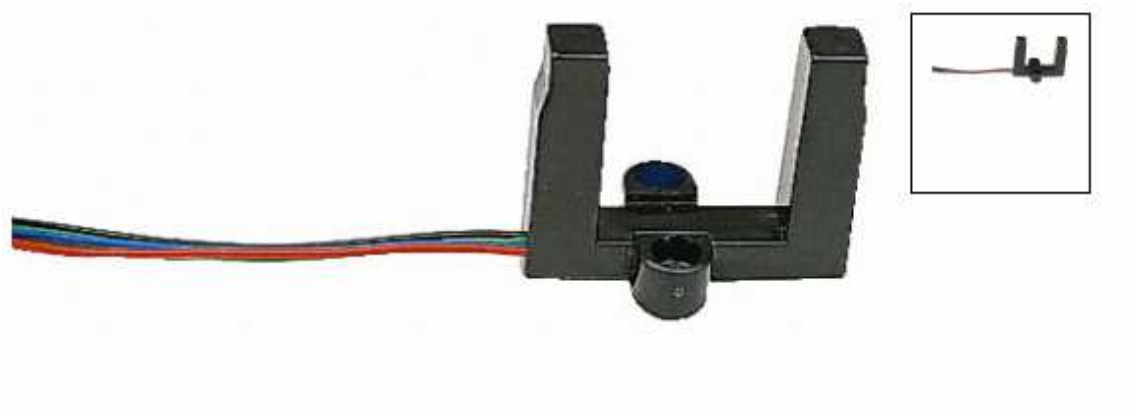


Figura 3.2.3.1.5. Sensor Honeywell H0 AD0150-2

Para limitar el consumo de corriente de este tipo de sensores es necesaria la instalación de una resistencia en serie. El cálculo de la resistencia a colocar en serie se muestra a continuación:

Tensión de alimentación: 5V (fijada por nosotros)

$$R = \frac{Vcc - Vsensor}{Isensor} = \frac{5 - 1{,}6}{0{,}02} = 170\Omega$$

Esquema de conexión:

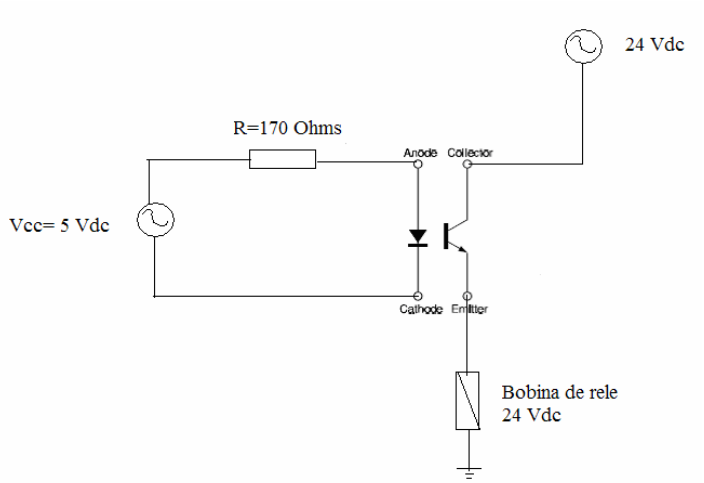


Figura 3.2.3.1.6. Esquema de connexion sensor Honeywell H0 AD0150-2

Problemática:

Este sensor nos permite colocar el tubo por el interior de la ranura, no detecta el tubo y si la bola, pero el gran problema que tiene es que la frecuencia de detección es muy baja y que la bola ha de pasar a una altura determinada por el interior de la ranura, es decir, a grandes velocidades el detector no detecta el paso de la bola.

Por otro lado el sensor no es capaz de entregar el amperio que necesita la bobina para producir el campo de inducción, este tipo de sensor es ideal para la conexión a PLC.

Coste del sensor: 30,56 Euros

4. Optek OPB732:

- Tipo: Interruptor reflector
- Distancia de detección: 25 mm
- Tipo salida: NPN open collector
- Tensión de alimentación: 1,8 V
- Corriente consumo: 20 mA

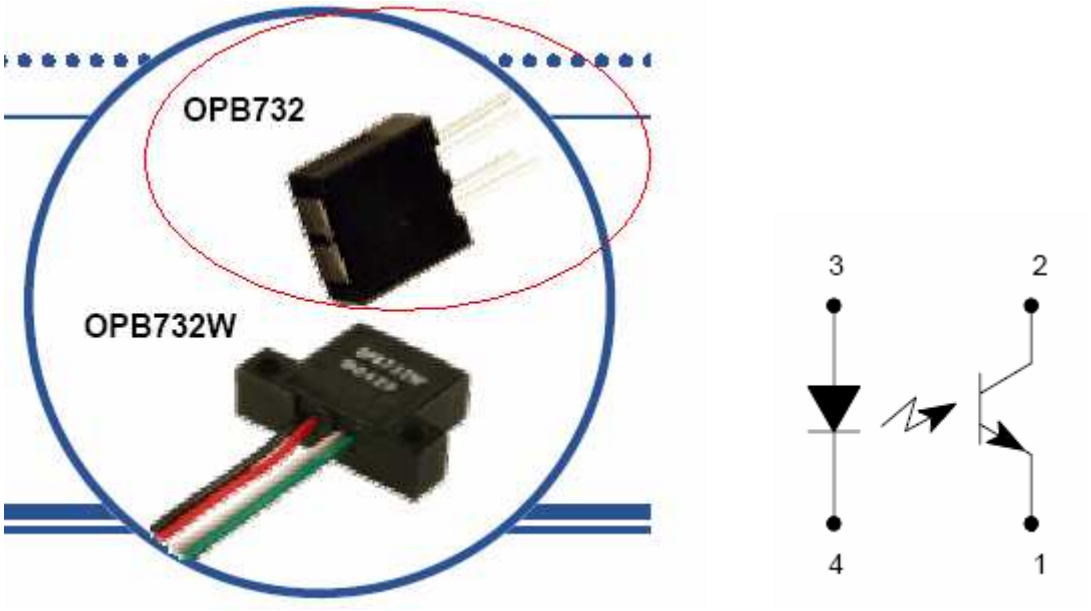


Figura 3.2.3.1.7. Sensor Optek OPB732

Para limitar el consumo de corriente de este tipo de sensores es necesaria la instalación de una resistencia en serie. El cálculo de la resistencia a colocar en serie se muestra a continuación:

Tensión de alimentación: 5V (fijada por nosotros)

$$R = \frac{Vcc - Vsensor}{Isensor} = \frac{5 - 1{,}8}{0{,}02} = 160\Omega$$

Esquema de conexión:

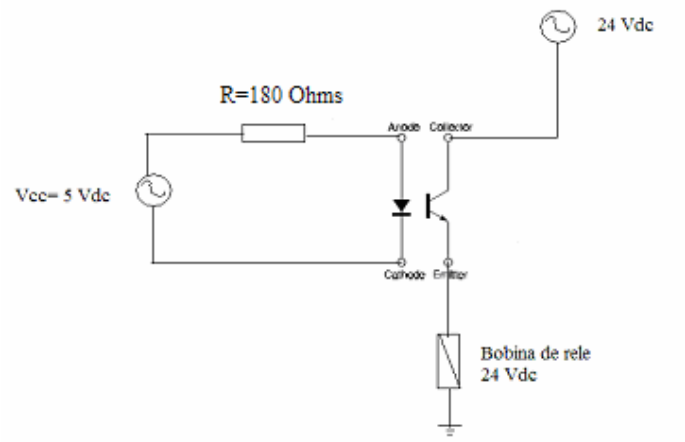


Figura 3.2.3.1.8. Eaquema de connexion sensor Optek OPB732

Problemática:

Este sensor la problemática que tiene es la incidencia de la luz exterior ya que influye en al detección del sensor y por otro lado el tiempo entre detecciones es muy bajo por lo que a grandes frecuencias no detecta el paso de la bola, por estos motivos este sensor fue descartado.

Coste del sensor: 4,00 Euros

3.2.3.2 Sensor montado en la maqueta

Finalmente después de realizar todas las pruebas con los sensores anteriormente citados, nos vimos obligados a escoger un tipo de sensor más robusto con mejores prestaciones para nuestra aplicación.

El sensor escogido es de tipo industrial con lo cual el coste del sensor es más elevado pero obtenemos las prestaciones mínimas necesarias para nuestra aplicación.

A continuación se describe el sensor seleccionado:

Pepperl+Fuchs GD18/GV18/73/120:

- Tipo: Interruptor reflector
- Distancia de detección: 0… 20 m
- Tipo salida: 2 PNP
- Tensión de alimentación: 10….30 Vdc
- Corriente consumo: < 20 mA

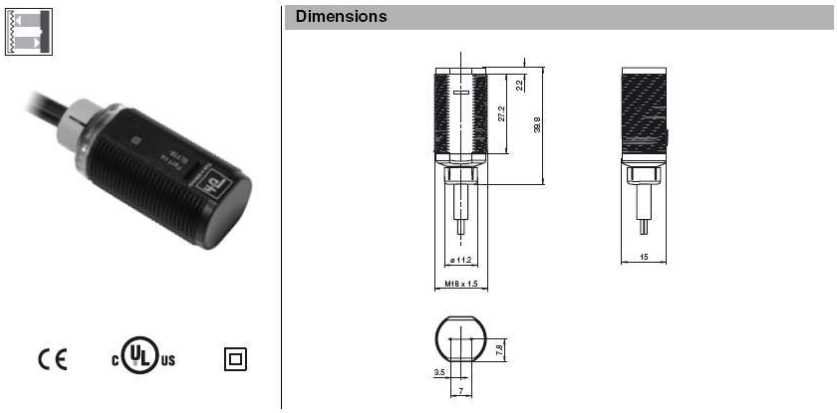


Figura 3.2.3.2.1. Sensor Pepperl+Fluchs GD18

Esquema de conexión:

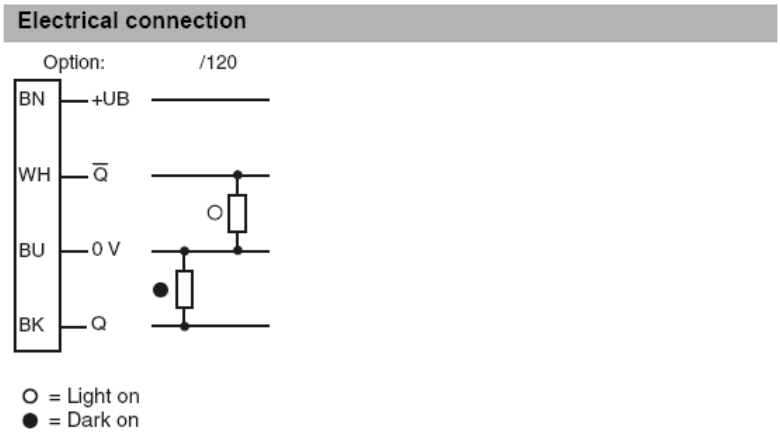


Figura 3.2.3.2.2. Esquema de connexion sensor Pepperl+Fluchs GD18

Estos sensores son muy robustos frente a la alimentación de entrada. La frecuencia de conmutación del interruptor que lleva integrado es muy alta por lo que se convierte en el mejor sensor que hemos probado.

Este sensor presento una serie de dificultades en la detección debido al grueso del haz de luz del sensor, por lo que se tubo que limitar el haz de luz que emitían para que fuese viable su utilización en nuestra aplicación.

Otro factor a tener en cuenta es la frecuencia de conmutación del sensor. Este sensor nos ofrece una frecuencia de conmutación de 500 Hz como vemos en la siguiente tabla:

GLV18 Series Specifications		
LOAD CURRENT		100 mA max
VOLTAGE DROP		≤ 1.5 VDC
SHORT CIRCUIT AND OVERLOAD PROTECTION		Yes
REVERSE POLARITY PROTECTION		Yes
SUPPLY VOLTAGE		10-30 VDC
VOLTAGE RIPPLE		10%
CURRENT CONSUMPTION		< 20 mA
RESPONSE TIME		≤ 1 ms
SWITCHING FREQUENCY		500 Hz
LIGHT SOURCE		Visible red LED 640 nm
STANDARDS		EN 60947-5-2
PROTECTION (IEC)		IP67
AMBIENT LIGHT RESISTANCE		≤ 30,000 lux
TEMPERATURE RANGE	*WORKING*	-13 °F to +140 °F
	STORAGE	-40 °F to +158 °F
HOUSING MATERIAL		Polycarbonate
	LENS	PMMA
APPROVALS		CE c UL us

Esto limita que el tiempo mínimo entre paso de bola por el mismo sensor sea de 2 mseg, lo cual se considera mas que suficiente para nuestra aplicación.

3.2.4 Bobinas

3.2.4.1 Selección de bobina

La selección de la bobina se ha realizado en función de la forma geométrica de esta. Se decidió por diseño utilizar una bobina toroidal ya que esta forma el campo magnético en su centro y además permite hacer pasar el tubo, por el que circula la bola metálica, por dentro de la bobina De esta manera podemos atraer la bola hacia el centro de la bobina y al desconectar esta antes de que la bola pase por el centro conseguimos acelerar la bola sin frenarla.

Realizamos varias pruebas para seleccionar el tipo de bobina. Estas pruebas están documentadas en el DVD que se adjunta en este proyecto.

Las primeras pruebas se realizaron con bobinas de contactores, tanto en continua como en alterna. Durante estas pruebas ya observamos de forma empírica que en alterna las bobinas ejercían más fuerza sobre las bolas metálicas que si las alimentábamos en continua.

Para poner un límite en el tamaño de la bobina se decidió seleccionar una que ejerciera la fuerza justa para desplazar la bola de bobina a bobina en corriente continua. De esta manera al minimizar la bobina nos obligábamos a afinar en el control de disparo.

Otro factor que se tuvo en cuenta fue el económico de esta manera también limitamos el valor de la bobina, de manera que los dos factores para determinar la bobina nos condujeron a escoger la bobina más pequeña posible.

La bobina seleccionada es la siguiente;

-Espiras 1350,
-Hilo 0,450 mm diametro
-resistencia: 28 ohm.
-Consumo a 24 V: 800 mA
-Inductancia: 100 mH

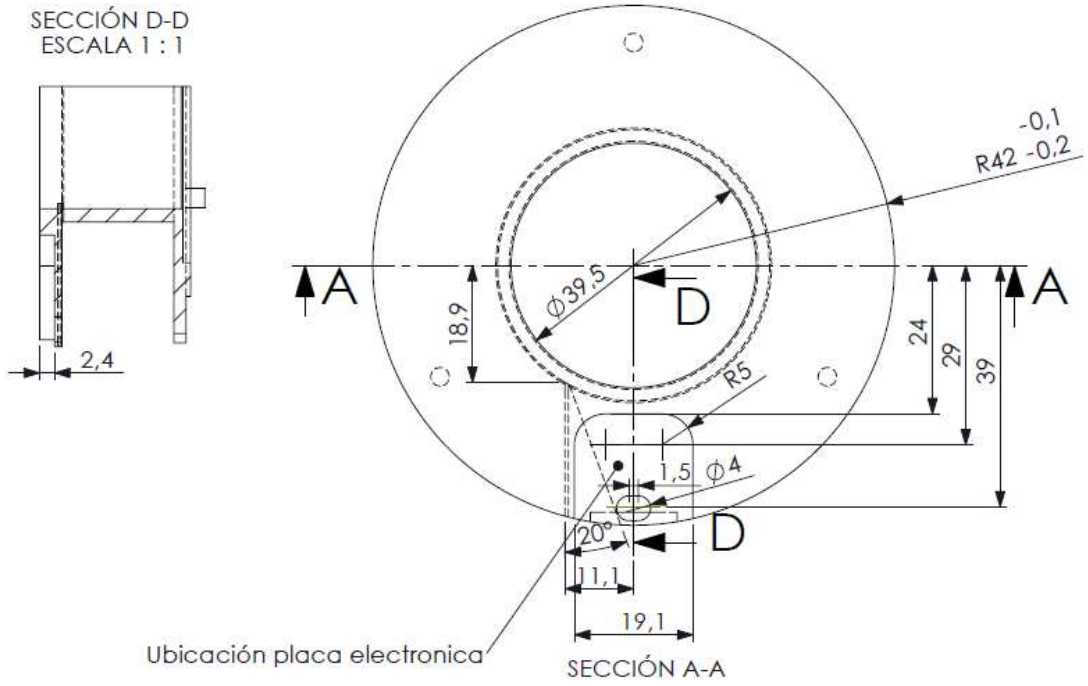


Figura 3.2.4.1.1. Bobina de 100mH seleccionada

Esta bobina nos permite hacer pasar el tubo de diámetro interior 25mm por el que circula la bola por dentro de ella, pues se puede observar que el diámetro de la bobina toroidal es de 39,5mm.

3.2.5 Grapas de fijación

Para sujetar el tubo en la maqueta se han utilizado unas grapas o abrazaderas de plástico sujetas sobre un taco de madera para proporcionar la altura necesaria para hacer pasar el tubo por dentro de las bobinas.

Figura 3.2.5.1. Grapa de fijación

3.3 Montaje de la maqueta

El montaje de la maqueta se ha realizado a mano sin más ayuda que nuestros propios medios y con herramientas simples, de las que podemos encontrar en cualquier casa. Nos referimos a este dato por que seguramente muchos de los problemas que hemos tenido después vienen derivados de este hecho. No lo decimos por que si, si no por que en el proceso de puesta en marcha del sistema nos hemos dado cuenta que pequeñas variaciones en los diferentes parámetros físicos nos comportaban grandes diferencias de funcionamiento.

Por supuesto todo se ha hecho con el máximo rigor posible pero dado los medios de que hemos dispuesto se han derivado errores en las distancias entre sensores o con las distancias de sensores a centro de bobina etc...

Pero también cabe destacar que debido a la necesidad de poder mover o cambiar elementos de la maqueta, tanto para solucionar averías como para implementar pruebas o fases del proceso de obtención de puntos de corte, la robustez de la maqueta se ha visto afectada, de manera que cada vez que se mueve la maqueta o se manipula para realizar cualquier acción, la maqueta queda desnivelada y se tiene que volver a nivelar. Estos cambios en el hardware dificultan el correcto funcionamiento del sistema.

Se comentara a continuación las diferentes soluciones que se han implementado en el montaje de la maqueta, tanto para salvar los problemas derivados de la selección de material, como para simplificar en lo posible tanto el soporte físico como la futura programación.

3.3.1 Sensores

Los sensores tienen que estar en parejas a una distancia conocida entre ellos y a una distancia conocida al centro de la bobina.

Figura 3.3.1.1. Detalle de la construcción de la maqueta, pareja de sensores

A una distancia conocida entre ellos para poder calcular la velocidad de paso de la bola. Para conocer la velocidad utilizamos el tiempo que trascurre entre el corte de la bola por el primer sensor y el corte de la bola por el segundo sensor. Conociendo la distancia entre los sensores solo nos queda aplicar:

$$V = \frac{e}{t}$$

A una distancia conocida al centro de la bobina para determinar el tiempo de disparo, pues sabiendo la velocidad de paso y sabiendo la distancia hasta el centro de la bobina se puede aproximar el tiempo que tardara la bola en llegar al centro de la bobina, este punto será el tiempo límite para efectuar la desconexión de las bobinas.

Para situar los sensores tanto a una distancia conocida entre ellos como a la altura necesaria para detectar el paso de la bola por la zona de rodadura, nos construimos y mecanizamos unos soportes metálicos a fin de roscar en ellos los sensores y así fijar tanto la altura de detección como la distancia entre sensores.

Figura 3.3.1.2. Soporte de sensores

Respecto la conexión eléctrica hay que tener en cuenta que los ocho sensores están dispuestos en parejas. Dentro de cada pareja tenemos:

- Primer sensor: Sensor que detecta el primer corte de la bola.
- Segundo sensor: Sensor que detecta el segundo corte de la bola.

Así pues tenemos un grupo de ocho primeros sensores y un grupo de ocho segundos sensores. Cada grupo de sensores esta conectado eléctricamente a una entrada del la placa de control a través de su placa de adaptación de señales.

Por otro lado la placa FLEX (explicada más adelante en este mismo proyecto) funciona con lógica negativa a sus entradas. Por lo tanto el sensor a de enviar un cero al detectar la el corte de la bola.

Al utilizar una entrada para cada grupo de sensores la solución a sido seriar todos los contactos de salida de los sensores para de esta manera al conmutar a cero un solo sensor todos los demás pasan a valer cero creando así una función AND lógica en la entrada de la placa FLEX de control.

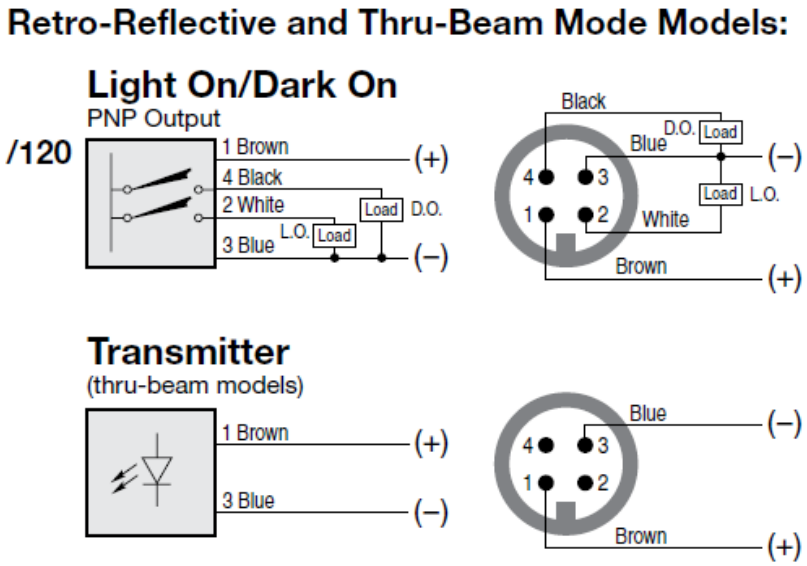


Figura 3.3.1.3. Conexión sensores (transmisor y receptor)

En la figura anterior observamos como se tienen que conectar los sensores de la maqueta, para obtener la lógica negativa, así como que contactos hay que seriar para obtener la AND lógica antes comentada.

3.3.2 Bobinas

Las bobinas están dispuestas en la maqueta repartidas cada 90 grados

Figura 3.3.2.1. Detalle de la construcción de la maqueta, ubicación bobinas

Debido a que las bobinas tienen una corriente nominal de 0,8A para 24V en continua y que para aumentar su fuerza magnética se conectan a 48V para obtener una corriente de 1,6A y que las fuentes de alimentación de que disponemos en el laboratorio sólo son capaces de alimentar un par de bobinas simultáneamente, se han alimentado las bobinas en grupos de dos. Si numeramos las bobinas del 1 al 4 de manera consecutiva, los grupos de conexión serán los siguientes:

- Grupo 1 bobinas 1 y 3
- Grupo 2 bobinas 2 y 4

De esta manera conseguimos que el tiempo que tienen las fuentes de alimentación para conectarse y desconectarse sea el doble que conectando todas las bobinas con una sola fuente de tensión.

Dado que el control de disparo de las bobinas de hace desde la placa de control, solo se han habilitado dos salidas para el disparo, pues cada salida dispara dos bobinas.

Para la conexión en alterna no se presenta este problema dado que se conectan las bobinas directamente a la red de compañía eléctrica y no tenemos el problema de que se amorren las fuentes. Pero por simplicidad de diseño hardware y de programación se ha respecta esta disposición de dos grupos de dos bobinas y dos salidas para las cuatro salidas.

3.3.3 Cableado

El cableado de potencia se ha realizado con cable de 1,5 mm^2, este cable puede aguantar hasta 13A por lo que se considera más que suficiente para alimentar los grupos de dos bobinas.

El cableado de control o señales se ha realizado o con el propio cable de los dispositivos o con cable especial para este tipo de comunicaciones.

Los cables se han conducido a través de dos canales de plástico de 26x16mm situadas en los laterales de la maqueta.

3.3.4 Regletero eléctrico

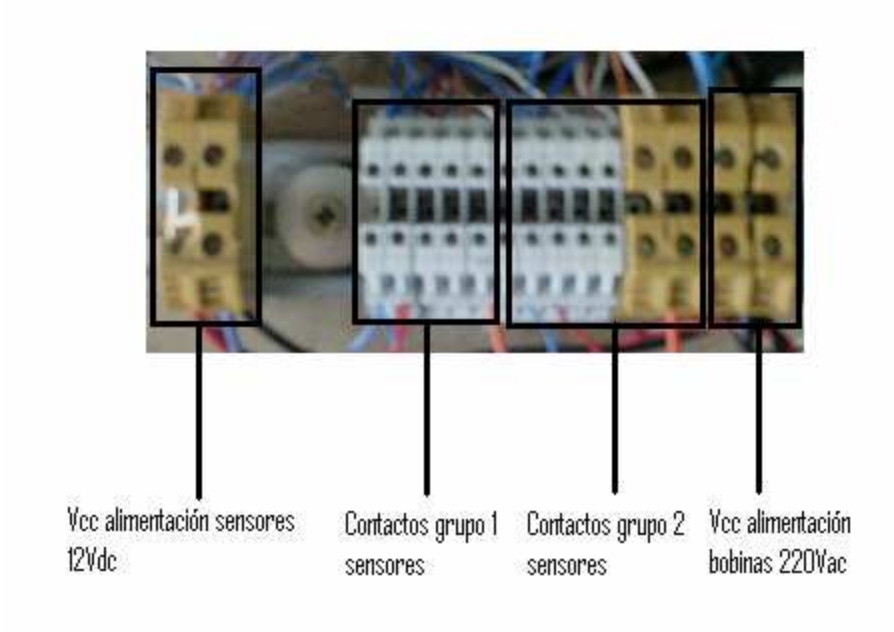


Figura 3.3.4.1. Detalle del regletero de conexiones eléctricas

3.3.5 Montaje final maqueta

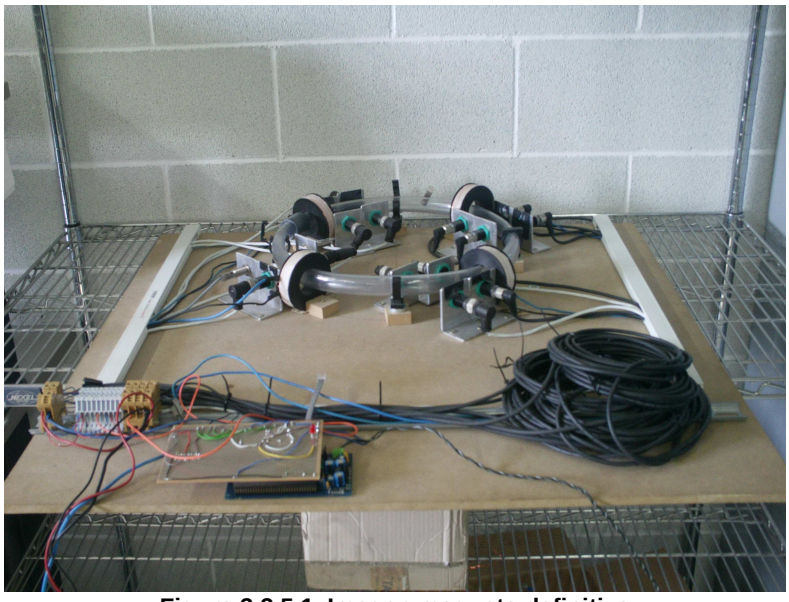


Figura 3.3.5.1. Imagen maqueta definitiva

4. HARDWARE, PLACA DE CONTROL Y DRIVERS DE SEÑALES

Las diferentes placas usadas para poder ejecutar los diferentes controles implementados sobre la maqueta son:

Las placas utilizadas son:

- Placa Flex full Base Borrad
- Flex Demo Daugther Borrad
- Driver de señal para continua
- Driver de señal para alterna

4.1 Flex Full Base Board

La placa Flex Full Board es una placa de la casa Flex que gracias a la DSPIC de la marca Microchip que lleva instalada permite realizar aplicaciones en tiempo real. Esta placa es la utilizada para la implementación de nuestro control, el resto de placas son para la adaptación de señales entre esta placa y nuestra maqueta.

Esta placa esta formada por los siguientes elementos:

- Microchip DSPIC DSC microcontroller dsPIC33F256MC710
- Zócalo de 100 pines Plug-In Module (PIM) válido para Microchip
- Conector de programación ICD2
- Conector USB para transmisión de datos
- Conectores de alimentación eléctrica
- LED's para la monitorización del estado de las funciones de la placa
- Microchip PIC18, PIC18F2550 micro controlador para la programación
- Conectores para el acoplamiento de placas de prueba

Figura 4.1.1. Placa Flex Full Base Board

A continuación se indican los recursos utilizados en nuestra aplicación así como la

configuración utilizada mediante los jumpers que tiene la placa para poder ejecutar la aplicación.

En primer lugar para poder utilizar la placa es necesario configurar los pines de la misma. A continuación se muestra la ubicación de los pines en la placa y la posición de los mismos para la configuración de la placa en nuestra aplicación (la posición en nuestra aplicación son los marcados de color gris).

Jumper	pos. 1-2	pos. 2-3
JP3	GND	-
JP4	GND	-
JP5	EARTH	-
JP6	+3.3V	+AV DD_{ext}
JP7	GND	AV SS_{ext}
JP8	+5V	+3.3V
JP9	+USB	+5V
JP10	DSP_MCLR	PIC18_MCLR
JP11	DSP_PDATA	PIC18_PDATA
JP12	DSP_PCLK	PIC18_PCLK
JP13	VDD pull up	-
JP15	SOSCI	CRYSTAL
JP16	SOSCO	CRYSTAL
JP17	GND	EARTH

Figura 4.1.2. Configuración de los Jumpers de la placa

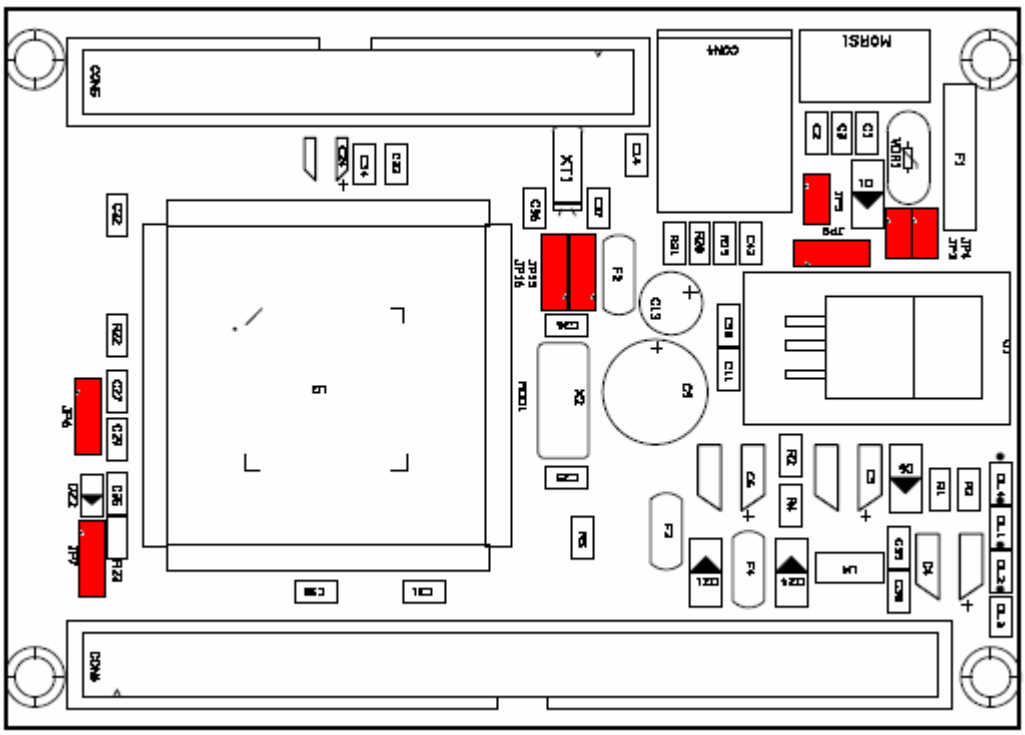


Figura 4.1.3. Ubicación de los Jumpers en la placa

Los registros que hemos utilizado de esta placa son:

- 3 entradas que pertenecen a los registros RD4, RD5 y RF2

- 3 salidas que pertenecen a los registros RF0, RF1 y RF3

Las entradas las utilizaremos para:

- RF2: Recibir los datos enviados por el PC a la Dspic
- RD4: Recibir las señales del segundo grupo de sensores ópticos
- RD5: Recibir las señales del primer grupo de sensores ópticos

Las salidas las utilizaremos para:

- RF3: Enviar los datos de la Dspic al PC
- RF0: Encendido y apagado del primer grupo de bobinas
- RF1: Encendido y apagado del segundo grupo de bobinas

Por último se muestra cada registro a que PIN de la placa pertenece.

Registro	Pin	Conector
RF0	21	Con 5
RF1	24	Con 5
RF2	42	Con 6
RF3	39	Con 6
RD4	17	Con 5
RD5	20	Con5

4.2 Flex Demo Daugther Board

Placa de pruebas que utilizamos para empezar a programar en el programa RT-Druid de Evidence y que nos permitió aprender el lenguaje de programación básico de la placa FLEX.

A continuación indicaremos las características de la placa, ver fig. 4.2.1, para las diferentes opciones que tiene y por último explicaremos las pequeñas aplicaciones realizadas sobre esta placa que luego nos ayudarán a implementar nuestra aplicación final.

Esta placa esta formada por:

- 2 salidas DAC con resolución de 12 bits
- 1 Acelerómetro
- Soporte para encoger
- 4 Pulsadores
- 8 Leds
- 1 LCD de 16 caracteres x 2 líneas
- 1 Zumbador
- 1 Potenciómetro
- 1 Sensor de temperatura
- 1 Sensor de luz

- 1 Transmisor-Receptor de infrarrojos
- 1 Zócalo para comunicación con otros módulos

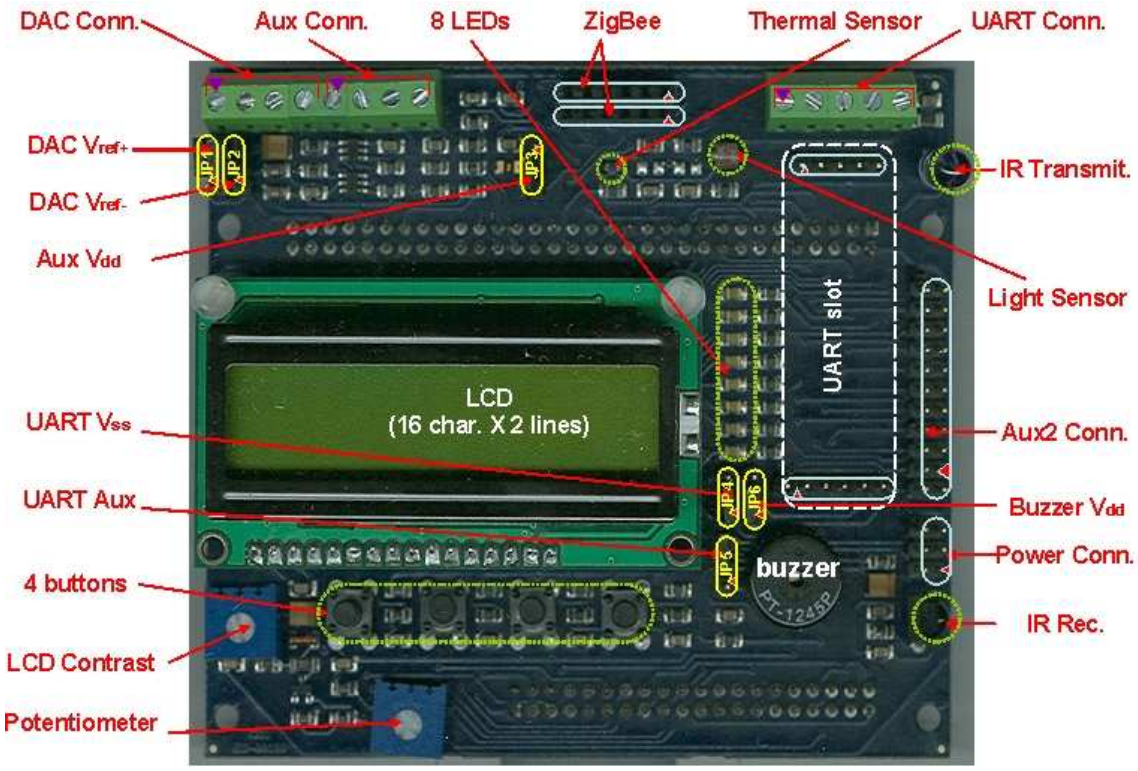


Figura 4.2.1. Flex Demo Daugther Board

Figura 4.2.2. Montaje placa Flex Full Board y placa Flex Demo Daugther Board

Utilizando esta placa se realizaron dos programas de prueba que fueron claves para el desarrollo de la aplicación final.

A continuación se explican brevemente las dos aplicaciones realizadas sobre esta placa de pruebas.

- El primer programa de pruebas realizado fue el encendido de uno de los leds de la placa de pruebas al pulsar un interruptor de la misma placa. Esta pequeña

aplicación se realizó mediante la programación en tiempo real (utilizando tareas) y mediante la programación de interrupciones.

Esto nos sirvió para poder realizar la lectura de los sensores ópticos utilizados para detectar el paso de la bola en la maqueta (señal de entrada del pulsador en el programa de la placa de pruebas) y la desactivación de las bobinas de nuestra maqueta en el momento deseado (encendido-apagado del led en nuestro programa de la placa de pruebas).

- El segundo programa de pruebas realizado fue la comunicación de la placa de pruebas, mediante la UART que tiene integrada, con el PC utilizando el Hyperterminal para la recepción de un mensaje continuado.

  Esto nos sirvió para comunicar la maqueta con el PC y así poder utilizar MATLAB para ver si obteníamos tiempos de corte en los que se conseguía acelerar la bola.

Jumper	pos. 1-2	pos. 2-3
DAC Vref+ [JP1]	+5V	+3.3V
DAC Vref- [JP2]	GND	GND_{out}
Aux Vdd [JP3]	V_{out}	+5V
UART Vss [JP4]	GND	GND_{out}
UART Aux [JP5]	INT (Konnex/EIB)	RTS (232/422/TTL)
Buzzer Vdd [JP6]	V_{out}	+5V

Figura 4.2.3. Configuración de los Jumpers de la placa

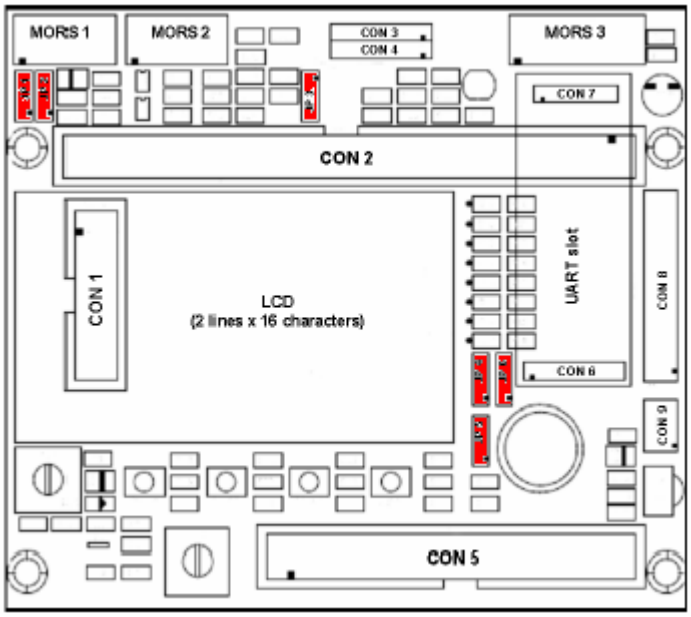


Figura 4.2.4. Ubicación de los Jumpers en la placa

4.3 Driver de señal para continua

Esta placa se ha realizado totalmente por nosotros, en ella se pone en práctica los conocimientos obtenidos con la placa de pruebas de la casa Flex.

A continuación indicaremos las características de la, ver fig. 4.3.1 y por último explicaremos la interconexión que se realiza y que nos permite comunicar la maqueta con el PC.

Esta placa esta formada por:

- 1 Conector DB9 para la comunicación RS232
- 1 UART formada por el chip MAXIM 233
- 2 Zócalos para realizar la interconexión entre las placas
- 2 Divisores de tensión para adaptar señales de los grupos de sensores
- 2 Relés de estado sólido para activar-desactivar los dos grupos de bobinas
- 3 Conectores

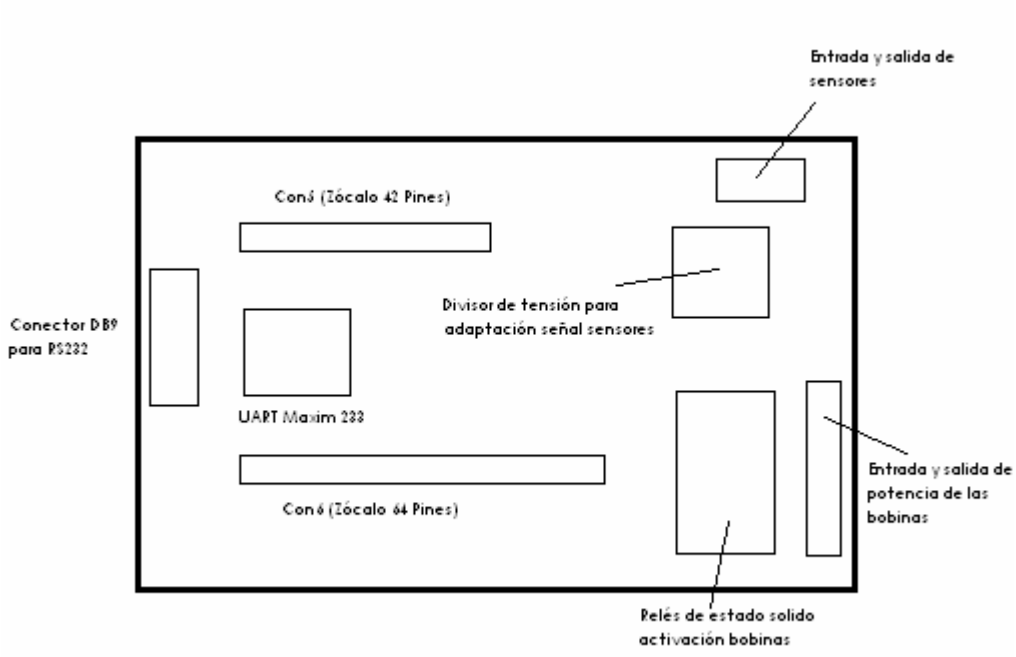


Figura 4.3.1. Diagrama de bloques driver continua

La conexión que se realiza en esta placa permite la comunicación entre la maqueta y la Flex full Board y entre la placa Flex y el PC mediante el modulo de comunicaciones RS233.

La conexión de las señales es la que se describe a continuación:

- Módulo de comunicaciones con MAXIM RS233:

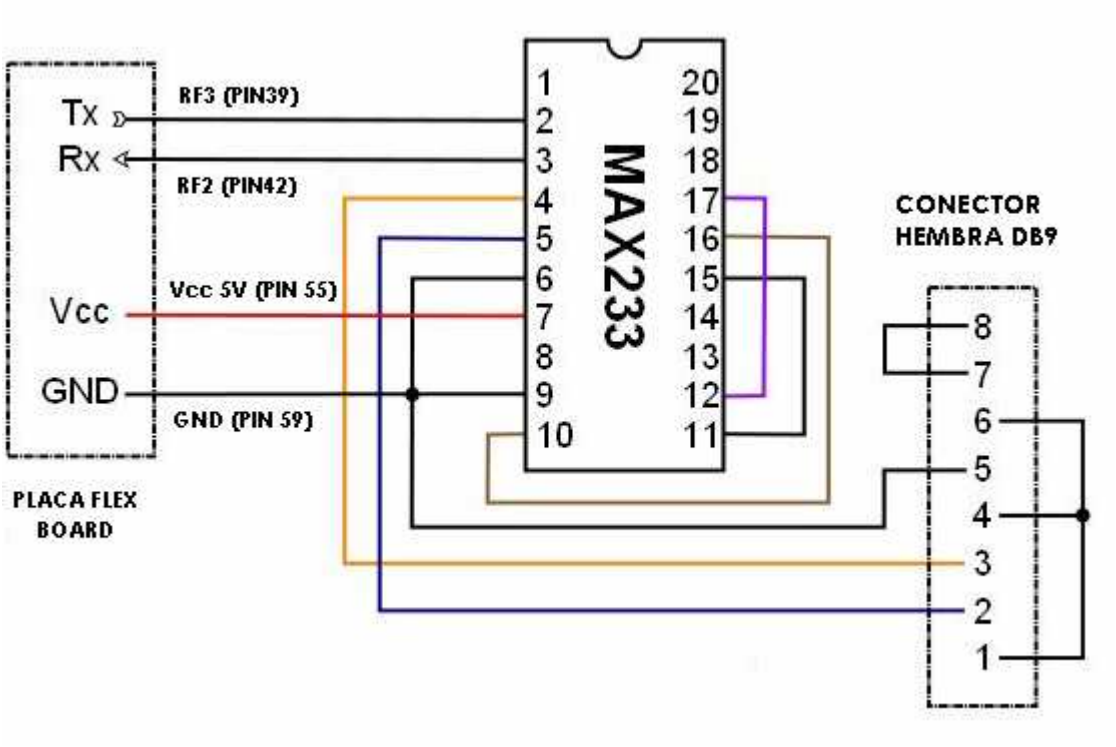


Figura 4.3.2. Esquema conexiones modulo MAXIM RS233

En el esquema anterior se indican las conexiones realizadas en la placa del Driver de Continua en el módulo de comunicaciones.

Todos los pines indicados en el esquema anterior pertenecen al CON 6 de la placa Flex Board, ver anejos adjuntados.

- Entradas Señal sensores:

Las entradas de señal de los sensores se conectan a la placa mediante los conectores indicados en la placa. Aquí se recibe la señal del sensor de 12 V, que es la tensión a la cual se alimentan los sensores, así que para poder adaptar esta señal hasta niveles lógicos de la placa Flex que equivalen a 3,3-5 V para 0 lógico y 0-2,2 V para 1 lógico, se instalan dos divisores de tensión que dividen por cuatro la tensión recibida por los sensores y así adaptamos la señal recibida a la placa de la entrada.

Por otro lado esta es la manera más fácil, rápida, sencilla y menos costosa de adaptar la señal y no producimos ningún tipo de retraso en la recepción de los cambios de estado en la entrada de la placa.

- Activación-Desactivación de las bobinas:

La activación-desactivación de las bobinas la realizamos utilizando la activación y desactivación de las salidas RF0 y RF1 de la placa Flex.

Estas salidas alimentan, en el caso de activar las bobinas de la maqueta, los terminales de control del relé de estado sólido permitiendo que el contacto que tiene este conmute y permita el paso de corriente a las bobinas de la maqueta en función de lo que

necesitemos.

Los relés de estado sólido son de la casa Crydom, la tensión de control oscila entre los 3 V y los 60Vdc y el consumo de este relé es de 20 micro amperios, lo cual lo hace perfecto para la activación directa desde la salida de la placa.

Por otro lado el contacto de este relé de estado sólido aguanta hasta 60 Vd y una corriente máxima de 3 A lo cual nos permite conectar un grupo de bobinas haciendo posible la activación y desactivación de estas.

Las características principales de estos relés son:

MODEL NO.	DO061A④	DO061B④	DMO063④
INPUT SPECIFICATIONS ①			
Control Voltage Range	3.0-9.0 Vdc	1.7-9.0 Vdc	3.0-10.0 Vdc
Nominal Input Impedance	270 Ohm	270 Ohm	200 Ohm
Typical Input Current @ 5 Vdc	15 mAdc	15 mAdc	20 mAdc
Must Turn On Voltage	3.0 Vdc	1.7 Vdc	3.0 Vdc
Must Turn Off Voltage	1.0 Vdc	0.8 Vdc	1.0 Vdc
OUTPUT SPECIFICATIONS ①			
Operating Voltage Range	3-60 Vdc	3-60 Vdc	0-60 Vdc
Load Current Range	.02-1.0 Adc		0-3.0 Adc
Max. Surge Current	5.0 Adc (1 Sec)		12.0 Adc(10 ms)
Max.Off-State Leakage @ Rated Voltage	200 µAdc		100 µAdc
Max. On-State Voltage Drop @ Rated Current	1.5 Vdc		0.4 Vdc ②
Max.Turn-On Time	50 µsec	50 µsec	50 µsec
Max. Turn-Off TIme	50 µsec	150 µsec	300 µsec

Figura 4.3.3. Características relé estado sólido DMO063

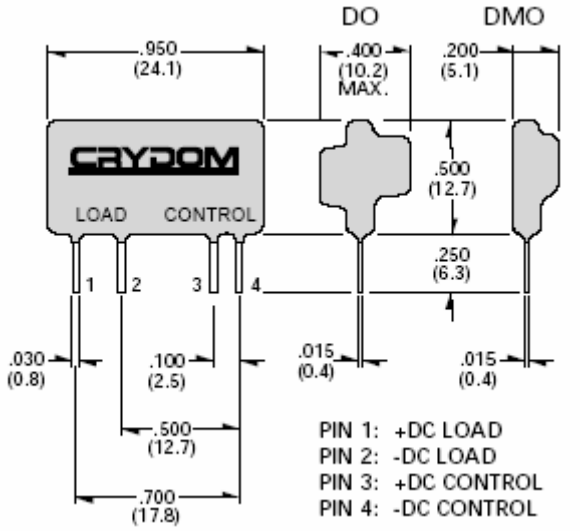


Figura 4.3.4. Esquema relé estado sólido DMO063

4.4 Driver de señal para alterna

La placa de driver para señal de alterna es exactamente igual a la descrita en el punto anterior.
La única diferencia entre las dos placas radica en los relés de estado sólido utilizados para la activación de los dos grupos de bobinas. A continuación se indican las características principales de los relés de estado sólido.

MODEL NUMBER	MP120D3	MP240D3	MP240D4
AC OUTPUT SPECIFICATIONS ①			
Operating Voltage Range (47-63 Hz) [Vrms]	12-140	24-280	24-280
Load Current Range [Arms]	.02-3	.02-3	.02-4
Transient Overvoltage [Vpk]	400	600	600
Max. Surge Current, (16.6ms) [Apk]	90	90	130
Max. On-State Voltage Drop @ Rated Current [Vpk]	1.6	1.6	1.6
Maximum I^2t for Fusing, (8.3 msec.) [A^2sec]	36	36	72
Max. Off-State Leakage Current @ Rated Voltage [mArms]	5.0	5.0	5.0
Min. Off-State dv/dt @ Max. Rated Voltage [V/µsec] ②	200	200	200
Max. Turn-On Time	1/2 Cycle	1/2 Cycle	1/2 Cycle
Max. Turn-Off Time	1/2 Cycle	1/2 Cycle	1/2 Cycle
Power Factor (Min.) with Max. Load	0.5	0.5	0.5

Figura 4.4.1. Características relé estado sólido MP240D3

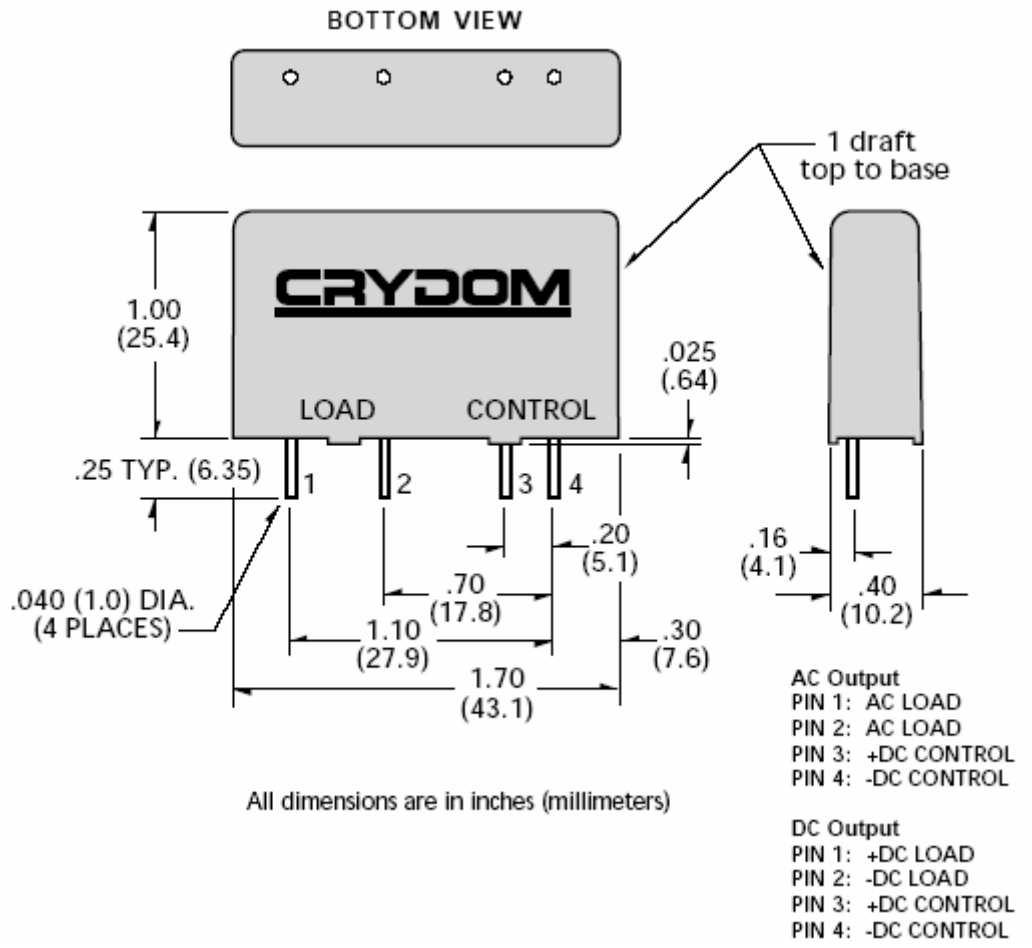


Figura 4.4.2. Esquema relé estado sólido MP240D3

5. ENTORNO DE PROGRAMACIÓN

En este capítulo veremos los diferentes apartados que integran la aplicación diseñada, desde que tipo de software utilizado para la programación, hasta la configuración e integración de cada módulo usado.

Para el desarrollo de la aplicación ha sido necesario el uso de los siguientes programas:

- RT Druid
- MPLAB IDE v 8.00
- Hyperterminal
- MATLAB

5.2 RT Druid

El programa RT Druid ha sido proporcionado por el Kit de evaluación de Flex / Evidence. Se compone de un conjunto de herramientas de desarrollo para crear y depurar aplicaciones integradas usando un código basado en C y C++ pero creado específicamente para las placas de la casa Flex.

RT Druid aporta un entorno de desarrollo sencillo, intuitivo e integrado que incluye un administrador de proyectos, un editor, herramientas de construcción, la opción de depurar la aplicación mediante simulación o hardware y una serie de instrucciones específicas para la programación de estas placas.

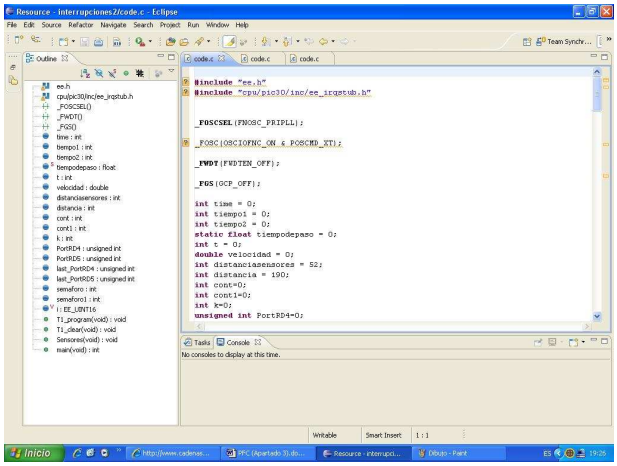


Figura 5.1.1. Ventana del RT Druid

Para realizar una aplicación con RTDRUID hay que realizar los siguientes pasos:

1. Seleccionar la pestaña “File” y clicar en “New”

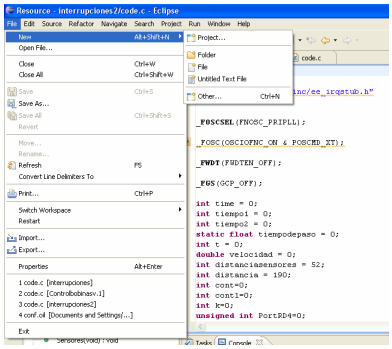


Figura 5.1.2. Paso 1 para crear un proyecto en RT Fruid

2. Seleccionar Evidence, seleccionar RT Druid Oil dentro del desplegable que aparecerá y clicar en “Siguiente”

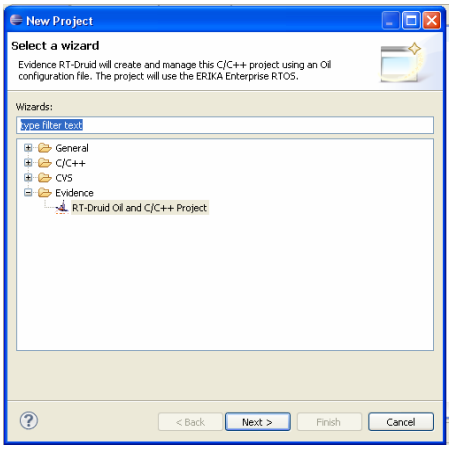


Figura 5.1.3. Paso 2 para crear un proyecto en RT Druid

3. Elegir el tipo de placa con el que vamos a trabajar que en nuestro caso es pic30, aparecerá un desplegable clicar en “Empty”, aparecerá un desplegable de nuevo, volver a clicar en “Empty” y por último clicar en “Next”.

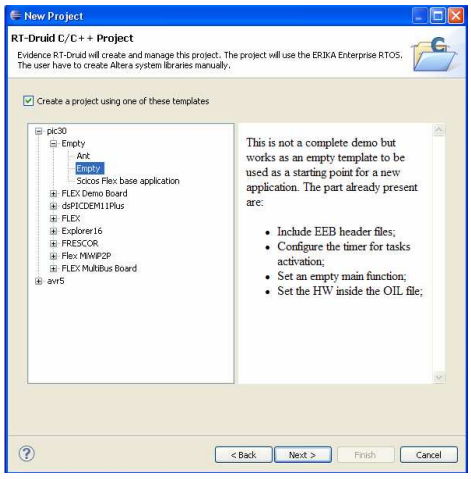


Figura 5.1.4. Paso 3 para crear un proyecto en RT Druid

4. Poner nombre de nuestro proyecto y clicar “Finish”

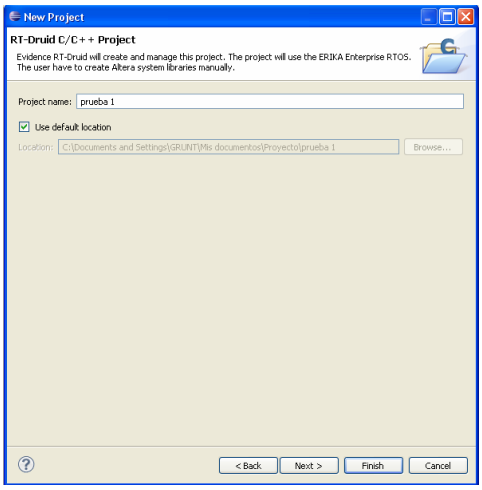


Figura 5.1.5. Paso 4 para crear un proyecto en RT Fruid

Una vez la aplicación está finalizada se debe crear un archivo con nombre “pic30.cof” para cargar el programa en la placa Flex a través del programa MPLAB. Para crear el archivo se han de seguir los siguientes pasos:

1. Seleccionar la pestaña de “Proyect” y clicar en “Build all”

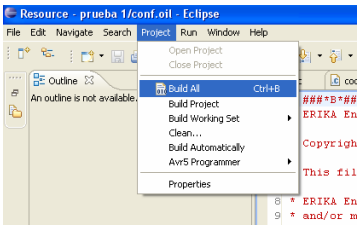


Figura 5.1.6. Paso 1 para crear el archivo de la aplicación “pic30.cof”

Este archivo se genera en la carpeta del proyecto definido anteriormente por lo que en el momento de querer volcarlo en la placa es necesario ir a buscar a esta carpeta.

5.3 MPLAB IDE v 8.00

El MPLAB IDE v 8.00 es una herramienta proporcionada por Microchip que permite cargar directamente la aplicación la DSPIC a través del archivo “pic30.cof” y del programador / debugador I2C de Microchip mediante el cable USB del ordenador conectado al I2C y con cable de teléfono del I2C a la placa flex donde está ubicado la DSPIC

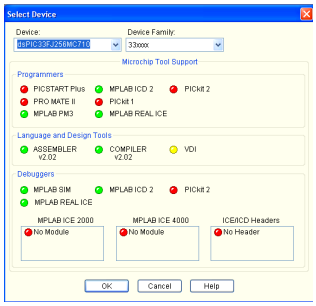


Figura 5.2.1. Configuración para cargar la aplicación en MPLAB

En primer lugar se debe configurar la DSPIC utilizada, en este caso DSPIC33FJ256MC710.

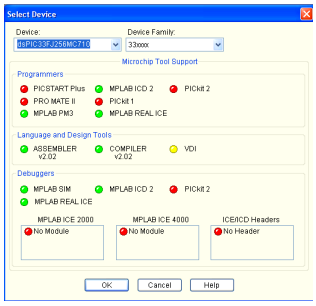


Figura 5.2.2. Selección de la dspic utilizada

Una vez escogida la DSPIC con la que vamos a trabajar hay seleccionar el programador o debuger que vamos a utilizar.

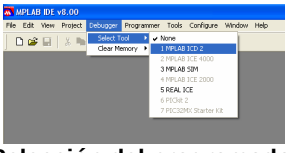


Figura 5.2.3. Selección del programador o debugador

Una vez realizados los pasos anteriores, se selecciona la ruta donde se encuentra el archivo “pic30.cof” creado anteriormente con el RT Druid, y se pulsa el botón “Program”. En unos instantes la aplicación es cargada en la DSPIC y ya puede ser utilizada.

5.4 HyperTerminal de Windows

La aplicación HyperTerminal de Windows ha sido empleada para realizar diferentes pruebas de visualización de datos adquiridos y enviados para la comunicación serie con el PC, a través del módulo UART de la placa realizada para el manejo de las entradas y salidas entre la maqueta y la placa Flex utilizada.

A continuación se detalla como se debe configurar para su correcto funcionamiento:

1. Ejecutar HyperTerminal y definir nombre a la nueva conexión

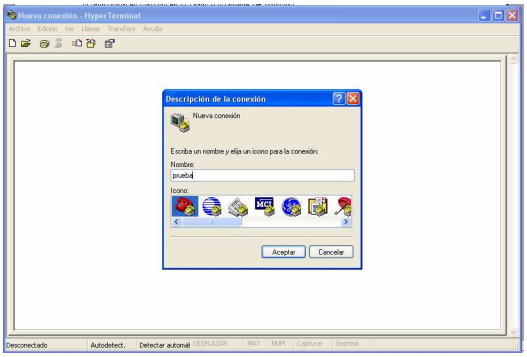


Figura 5.3.1. Crear nueva conexión

2. Seleccionar el puerto serie del PC al que está conectado la placa de manejo de señales entre la placa Flex donde está ubicada la Dspic y la maqueta:

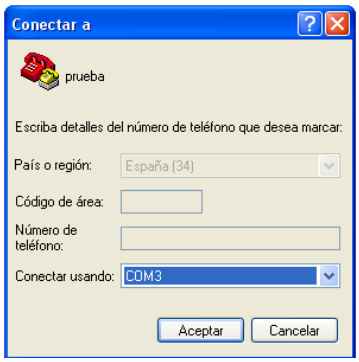


Figura 5.3.2. Conectar el puerto serie al PC

3. Configurar la conexión seleccionando la velocidad de transferencia, el número de bits de datos a enviar/recibir, la paridad y el control de flujo como se muestra en la siguiente figura:

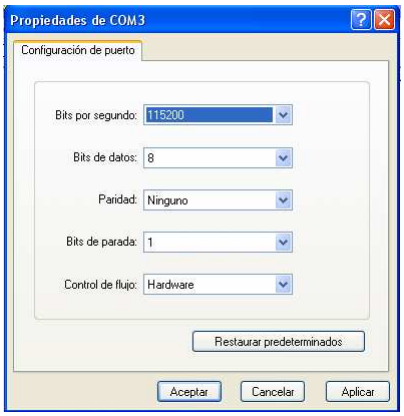


Figura 5.3.3. Configuración del puerto

4. Pulsar el icono de conectar la visualizar los datos enviados por la Dspic a través del puerto serie del Pc:

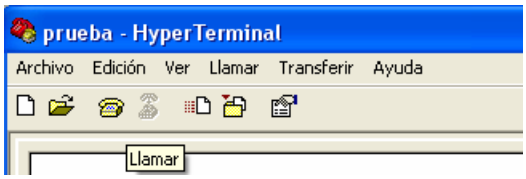


Figura 5.3.4. Visualización de datos en el HyperTerminal

5.5 MATLAB

El programa MATLAB se utiliza para obtener y poder trabajar con un mayor número de datos y así poder optimizar los tiempos de corte de corriente de nuestra aplicación.

MPLAB es una herramienta matemática para pruebas, control y diseño mediante programación.

El lenguaje utilizado es un seudo C.

A continuación se muestra el código realizado para obtener las aceleraciones experimentadas por la bola y el tiempo de corte de suministro de corriente a nuestras bobinas en cada una de las pruebas realizadas.

```
%m-file to adquire data from the serial port using RS232 at 115200bps
%data is stored in the matlab variable called t r x1 x2 u

clear all;                  %clear previous values
c='hola'
delete(instrfind)                   %clean all open ports
                            %configuring serial port
s=serial('COM1');                   %creates a matlab object from the serial port 'COM1'
s.BaudRate=115200;                      %baudrate=115200bps
s.Parity='none';                    %no parity
s.DataBits=8;                       %data sended in 8bits format
s.StopBits=1;                       %1 bit to stop
s.FlowControl='none';                   %no flowcontrol
s.Terminator='LF';                  %LineFeed character as terminator
s.Timeout=20;                       %maximum time in seconds since the data is readed
s.InputBufferSize=8192;
fopen(s)                        %open serial port object
for x=0:100
data=fread(s,2,'char');
v= (bitshift(data(1),8)+data(2));
v=v/1000

data=fread(s,2,'char');
t= (bitshift(data(1),8)+data(2));
t=t/1000
c
end

fclose(s)                       %close serial port object
delete(instrfind)                   %clean all open ports
```

6. CONTROL

Los controles y algoritmos que se han realizado y probado en la maqueta para comprobar empíricamente los resultados obtenidos en las simulaciones se pueden clasificar en tres tipos:

1. Control mediante el corte de alimentación de la bobina en el paso de la bola por el centro de la misma
2. Algoritmo de búsqueda de puntos óptimos de disparo
3. Control mediante funciones halladas a partir del segundo tipo de control

Estos controles se han implementado y probado tanto en funcionamiento en corriente continua de las bobinas como en funcionamiento con corriente alterna sobre las mismas.

Tal y como se demostrará en los apartados siguientes los resultados obtenidos en las simulaciones se asemejan bastante con los resultados hallados sobre la maqueta.

6.1 Control mediante paso por centro de la bobina

Este fue el primer control implementado sobre la maqueta y se partió de la hipótesis de intentar realizarlo aprovechando que la placa Flex Full Board dispone de un Kernel en tiempo real. Pero después de diversas pruebas se observó que la implementación de este control utilizando el Kernel de tiempo real no era factible debido a que el disparo de las tareas no coincidía con el funcionamiento de la maqueta.

Cuando se producía un evento en la maqueta, si la tarea que en ese momento havia disparado el Kernel no era la encargada de leer ese evento, el evento pasaba inadvertido por el control. Aunque el kernel es capaz de disparar las tareas de manera muy rápida, se perdían eventos y comportaba un funcionamiento incorrecto del control implementado. Debido a este hecho, se realizó este mismo control utilizando interrupciones y tal y como se puede apreciar en los videos entregados junto con la memoria se consiguió acelerar la bola significativamente.

El principio de funcionamiento de este control parte de calcular la velocidad de la bola y realizar el corte de alimentación de las bobinas justo en el momento en el que la bola pasa por el centro de estas. Observamos que el resultado obtenido no era el esperado por lo que realizamos ajustes en los tiempos de disparo.

Los ajustes consistieron en rebajar el tiempo de disparo para cortar la alimentación de la bobina antes que la bola llegase al centro de la misma, de esta manera se consiguió que la bola adquiriera una mayor aceleración.

Fue un buen comienzo, pero las variaciones de la velocidad de entrada influían mucho en los tiempos de disparo y esto provocaba que el control fuese poco fiable, por este motivo decidimos montar otro tipo de control.

En este apartado se explica el código utilizado tanto en tiempo real como en interrupciones.

Tiempo real:

El código en tiempo real se divide en tres tareas, el código completo está disponible en los anexos de la memoria.

- Tarea “Tiempo 1”

Esta Tarea es la encargada de ver si la bola ha cortado el primer grupo de sensores viendo si se produce un cambio en el registro RD5 y en caso de producirse este evento guardar el valor del Timer en ese preciso momento, encender los dos grupos de bobinas y realizar un pequeño retardo para intentar evitar que las siguientes tareas no vean los eventos producidos en la maqueta. A continuación se muestra el código de esta Tarea.

```
TASK(Tasktiempo1)
{
PortRD5 = PORTDbits.RD5;
        if (PortRD5 != last_PortRD5)
        {
                if (paro==0)
                {
                        tiempo1=time;
                        LATFbits.LATF0 = 1;
                        LATFbits.LATF1 = 1;
                        paro=1;
                        for (casa=0;casa<300;casa++)
                        {
                        }
                {
        }
last_PortRD5=PortRD5;
}
```

- Tarea “Tiempo 2”

Esta Tarea es la encargada de ver si la bola ha cortado el segundo grupo de sensores viendo si se produce un cambio en el registro RD4 y en caso de producirse este evento guardar el valor del Timer en ese preciso momento y realizar un pequeño delay para intentar evitar que las siguientes tareas no vieran los eventos producidos en la maqueta. A continuación se muestra el código de esta Tarea.

```
TASK(Tasktiempo2)
{
PortRD4 = PORTDbits.RD4;
        if (PortRD4 != last_PortRD4)
        {
                if (paro1==0)
                {
                        tiempo2=time;
                        inicio=1;
                        paro1==1;
```

```
                    for (casa=0;casa<300;casa++)
                    {
                    }
                }
        }
last_PortRD4=PortRD4;
}
```

- Tarea “Disparobobinas”

Esta función es la encargada de calcular la velocidad con la que iba la bola y ver el tiempo que tardaría en llegar al centro de la bobina cortando la corriente de la misma en ese preciso momento. A continuación se muestra el código de esta Tarea.

```
TASK(Taskdisparobobina)
{
        if (inicio==1)
        {
                tiempodepaso=(tiempo2-tiempo1)*1e-3;
                velocidad=distanciasensores/tiempodepaso;
        velocidad1=velocidad*1000;
                t=distanciabobina/velocidad;
                t=tiempo2+t;
                tiempo3=time;
                if (tiempo3>=t)
                {
                        LATFbits.LATF0 = 0;
                        LATFbits.LATF1 = 0;
                        inicio=0;
                        tiempo1=0;
                        tiempo2=0;
                        time=0;
                        tiempo3=0;
                        paro=0;
                        paro1=0;
                }
                else
                {
                        tiempo3=time;
                }
}
```

Como se pude apreciar era muy difícil ajustar los eventos de la maqueta con los disparos de estas tareas para hacer funcionar el programa de forma global, por lo que se optó por descartar el funcionamiento en tiempo real y dirigir el control hacia interrupciones debido a perdida de información anteriormente comentada.

Interrupciones:

Al cambiar el con el Kernel de tiempo real a control mediante interrupciones, se consiguió acelerar la bola cosa que hasta el momento no se había conseguido.

Al igual que en el apartado anterior se mostrará el código de las interrupciones,el resto del código se puede ver en los anejos de la memoria.

Este código está dividido en dos interrupciones principales.

- Interrupción del Timer

Esta interrupción es la que produce el microcontrolador cada vez que detecta un flanco de subida en el oscilador de la placa.

Pero además de las funciones propias del timer, en nuestra aplicación indicamos que en el momento en que el timer llegue al valor del tiempo que hemos calculado para que la bola llegue al centro de la bobina, se desactive la alimentación de las bobinas, para dejar de aplicar sobre la bola el campo magnético generado, y se habilite la interrupción de los sensores. A continuación se muestra el código de esta interrupción.

```
ISR2(_T1Interrupt)
{
	T1_clear();
	CounterTick(myCounter);
	time=time+1;
	if (m==1)
	{
		if (time>t)
		{
			LATFbits.LATF0 = 0;
			LATFbits.LATF1 = 0;
			semaforo1=0;
			m=0;
			paso=0;
			paso1=0;
		}
	}
}
```

- Interrupción de detección de sensores y cálculo de tiempo de corte alimentación bobinas

Esta interrupción se habilita cada vez que se produce una detección en los sensores de la maqueta.

Si detecta el primer grupo de sensores, se captura el tiempo justo en el que se produce el

evento y se habilita la detección del segundo grupo de sensores.

En el caso de ser el segundo grupo de sensores el que detecta el paso de la bola, se produce el cálculo de la velocidad y del tiempo en el que la bola tardará en llegar al centro de la bobina, que es el tiempo que tardaremos en desconectar los dos grupos de bobinas, y se activa la alimentación de los dos grupos de bobinas.

La activación de las bobinas se realiza con el segundo grupo de sensores para evitar posibles interferencias generadas por el campo magnético generado por la bobina que acabamos de pasar. A continuación se muestra el código de dicha interrupción.

```
ISR2 (_CNInterrupt)
{
PortRD4=PORTDbits.RD4;
PortRD5=PORTDbits.RD5;

/*Código al detector el Segundo grupo de sensores*/

if (PortRD4 != last_PortRD4)
{
        if (semaforo==0)
        {
                tiempo2=time;
                tiempodepaso=(tiempo2-tiempo1)*1e-3;
                velocidad=distanciasensores/tiempodepaso;
                t=distancia/velocidad;
                t=t*1000;
                m=1;
                cont++;
                semaforo=1;
                semaforo1=0;
                time=0;
                LATFbits.LATF2 = 1;
                LATFbits.LATF3 = 1;
        }
}
/*Código al detector el Primer grupo de sensors*/

if (PortRD5 != last_PortRD5)
{
        if (semaforo1==0)
        {
                tiempo1=time;
                cont1++;
                semaforo1=1;
                semaforo=0;

        }
}
last_PortRD4=PortRD4;
```

```
last_PortRD5=PortRD5;
IFS1bits.CNIF=0;
}
```

Los resultados obtenidos con este código tanto en funcionamiento en continua como en funcionamiento en alterna se pueden observar en los videos adjuntados junto a esta memoria.

6.2 Control de búsqueda de puntos óptimos de disparo

Al ver que somos capaces de acelerar la bola con el control anterior y para mejorar las aceleraciones que podemos conseguir, surge la necesidad de encontrar varios puntos de disparo óptimo. Con el control anterior no siempre acelerábamos la bola sino que a veces la frenábamos ya que no dábamos tiempo a anular por completo el campo magnético de la bobina.

Así que decidimos buscar una serie de tiempos de corte más óptimos a partir de una velocidad inicial conocida y así conseguir sacar una función que nos permitiese calcular este tiempo óptimo para todas las velocidades de paso de la bola.

Para realizar este control tuvimos que realizar una serie de cambios en el montaje de la maqueta tal y como se muestra a continuación.

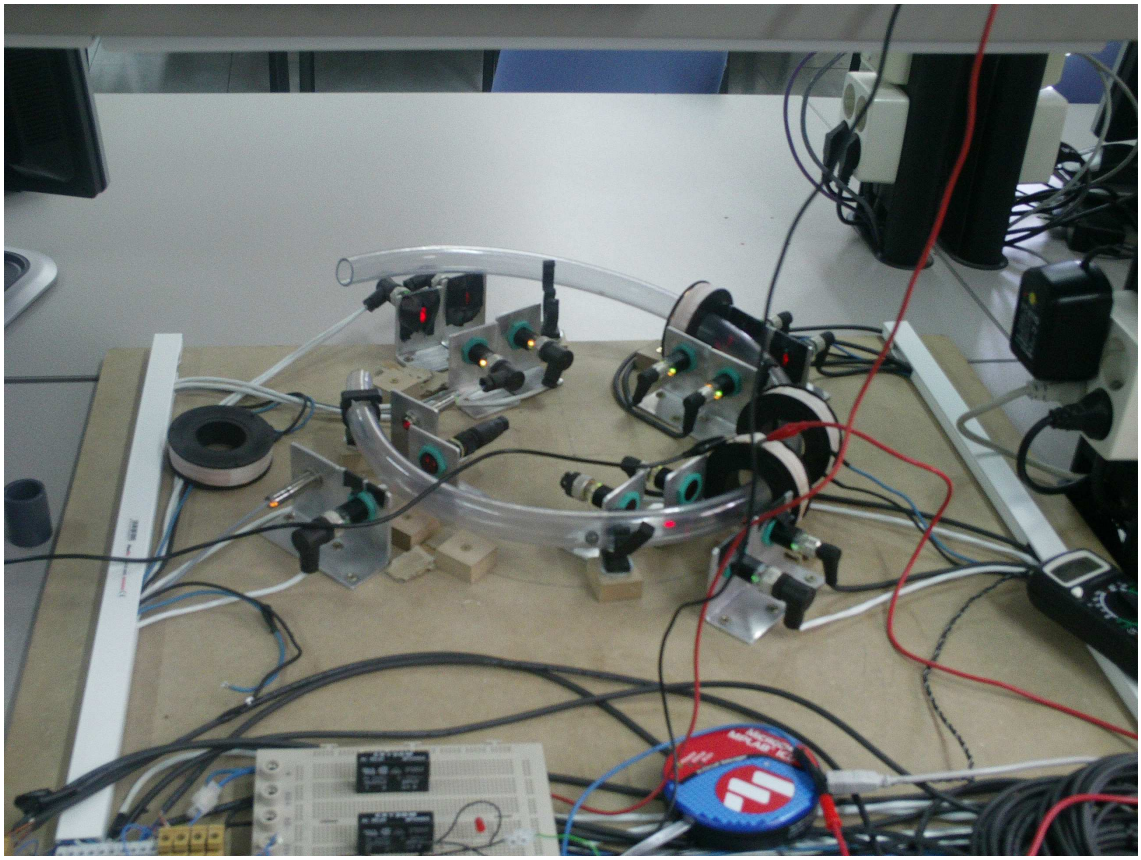


Figura 6.2.1. Modificaciones realizadas en la maqueta

Como se puede apreciar se procedió a abrir el tubo y lanzar la bola desde una altura conocida que nos permitía conocer la velocidad de inicio. Fijando la velocidad de entrada fuimos variando el tiempo de disparo hasta conseguir un disparo óptimo.

Mediante la pareja de sensores de entrada a la bobina y la pareja de sensores de salida podemos determinar si la velocidad de la bola ha aumentado después del disparo producido.

Una vez obtenidos un par de puntos de disparo, no muy lejanos entre ellos, sacamos la función de la recta de ese par de disparos y pasando los valores de velocidad de entrada, tiempo de disparo y velocidad de salida a través de la UART hacia MATLAB, implementamos un algoritmo entrenador para sacar diferentes puntos de disparo.

Este algoritmo calculaba el punto de disparo interpolando con los disparos obtenidos de forma manual. Por supuesto la velocidad de de inicio tenia que estar entre los dos valores encontrados anteriormente. De esta manera sacábamos más puntos de disparo entre los dos obtenidos de forma manual.

Como esto nos limitaba a encontrar disparos únicamente entre los dos valores encontrados de forma manual, fuimos buscando otros pares de valores de forma manual para poder incrementar el rango de búsqueda.

También realizamos algo parecido utilizando el control mediante paso por centro de bobina, recogiendo los datos y quedándonos con los mejores puntos de disparo encontrados.

De esta manera fuimos construyendo poco a poco una tabla de disparos para intentar implementar un control basado en estos.

El código generado para este control no difiere mucho del explicado en punto 6.1 así que se procederá a explicar interrupción de detección de sensores y cálculo de tiempo de corte alimentación bobinas que es donde se aplican los cambios para poder llevar a cabo este control.

El código entero se puede encontrar en los anexos de esta memoria.

- Interrupción de detección de sensores y cálculo de tiempo de corte alimentación bobinas

En esta función se introducen dos cambios sustanciales: el primero es que a partir del tiempo que tarda la bola en llegar al centro de la bobina se va acortando el tiempo en cada iteración que se produce al lanzar la bola para ver si conseguimos una aceleración más grande sobre la bola o no.

Por otro lado se configura el puerto RS232 de comunicaciones para transmitir los resultados obtenidos al PC y capturarlos a través del programa MATLAB y así poder analizar los resultados. A continuación se muestra el código realizado para esta aplicación.

```
ISR2 (_CNInterrupt)
{
PortRD4=PORTDbits.RD4;
PortRD5=PORTDbits.RD5;

/*Código al detector el Segundo grupo de sensores*/

if (PortRD4 != last_PortRD4)
{
        if (semaforo==0)
        {
                /*Calculamos la velocidad inicial*/
                tiempo2=time;
                tiempodepaso=(tiempo2-tiempo1)*1e-3;
                velocidad=distanciasensores/tiempodepaso;
                /*Enviamos la velocidad inicial al PC*/
                velocidad1=velocidad*1000;
                my_velocidad=velocidad1;
                low_byte=(EE_UINT8)my_velocidad;
                high_byte=(EE_UINT8)(my_velocidad>>8);
                EE_UART1_Send(high_byte);
                EE_UART1_Send(low_byte);
                /*Calculamos el tiempo de corte óptimo*/
                t=distancia/velocidad;
                t=(t*1000)-cont4;
                /*Enviamos el tiempo de corte al PC*/
                tiempo3=t;
                my_tiempo=tiempo3;
                low_byte=(EE_UINT8)my_tiempo;
                high_byte=(EE_UINT8)(my_tiempo>>8);
                EE_UART1_Send(high_byte);
                EE_UART1_Send(low_byte);
                /*Igual código anterior*/
                m=1;
                cont++;
                semaforo=1;
                semaforo1=0;
                time=0;
                LATFbits.LATF2 = 1;
                LATFbits.LATF3 = 1;
                /*Incrementamos cont4 para que el corte sea antes en el
                próximo lanzamiento*/
                cont4++;
        }
}

/*Código al detector el Primer grupo de sensors*/

if (PortRD5 != last_PortRD5)
{
```

```
        if (semaforo1==0)
        {
                tiempo1=time;
                cont1++;
                semaforo1=1;
                semaforo=0;
        }
}
last_PortRD4=PortRD4;
last_PortRD5=PortRD5;
IFS1bits.CNIF=0;
}
```

A partir de utilizar este código y de realizar muchos lanzamientos obtuvimos los siguientes resultados.

Velocidad inicio	Velocidad final	Tiempo de corte
0.019	0.426	1.607
0.068	1.925	1.315
0.073	1.677	1.283
0.086	1.625	1.207
0.118	0.658	1.02
0.126	0.171	0.967
0.13	0.565	0.942
0.147	0.192	0.843
0.169	0.504	0.82
0.172	0.658	0.654
0.177	0.838	0.666
0.187	0.327	0.61
0.189	0.945	0.593
0.191	1.13	0.58
0.233	0.712	0.335
0.246	1.083	0.256
0.339	0.962	0.428
0.39	0.896	0.433
0.426	1.061	0.416
0.429	1.019	0.386
0.481	0.634	0.39
0.485	0.776	0.388
0.49	1.019	0.415
0.509	0.65	0.377
0.509	0.881	0.

En la tabla anterior se indican los puntos en los que se ha obtenido aceleración de la bola con el control anteriormente explicado con la maqueta.

Como se puede apreciar se han agrupado de tal forma para que se pueda calcular la función necesaria que se aplicará en el control que explicaremos en el siguiente apartado.

Junto a esta memoria se pueden ver una serie de videos en los que se muestra como se obtienen dos tiempos de corte óptimos con este control.

6.3 Control mediante funciones halladas a partir del segundo tipo de control

En este control se aprovechan todos los datos conseguidos anteriormente y que confirman los resultados obtenidos mediante las simulaciones realizadas al inicio del proyecto.

Este es un control realizado a tramos a partir de los puntos encontrados en el apartado anterior. La idea de este control, a partir de las tres franjas bien diferenciadas encontradas anteriormente, es encontrar la función de cada tramo que nos permita calcular el tiempo de corte óptimo en función de la velocidad inicial para cada uno de los tramos, para así conseguir acelerar la bola de una forma más continua y no tan esporádica como se había conseguido hasta ahora.

Por supuesto realizar este control, limita la acción de control, pues solo actuamos si la velocidad de entrada está comprendida en uno de los tramos que se observan en la anterior tabla.

En este apartado procederemos a explicar todo el código en forma de diagrama de bloques de nuestra aplicación, como siempre el código de este control se puede encontrar en los anexos adjuntados en esta memoria.

El código de este control se puede dividir en cinco partes:

- Inicialización de variables
- Inicialización de entras / salidas
- Interrupción del Timer 1
- Interrupción Entradas de los Sensores y cálculo tiempo de corte óptimo
- Programa Principal

A su vez las funciones utilizadas las podemos dividir en distintos grupos:

1. Funciones de configuración de las entradas de la Dspic
2. Funciones de configuración de las salidas de la Dspic
3. Funciones de configuración de recursos internos de la Dspic

6.3.1 Funciones de configuración de entradas de la Dspic

Esta función es la encargada de configurar las entradas de la Dspic por las que recibiremos información de nuestra aplicación física (La maqueta).

Función Sensores:

Esta función es la encargada de definir los bits 4 y 5 del puerto TrisD como entradas esto lo hacemos colocando estos bits a 1, ya que esta es la forma de indicar que son entradas.

Por otro lado configuramos estas entradas como interrupciones utilizando los bits de configuración de interrupciones asociados a estas entradas, en este caso son los bits CN13IE y CN14IE que los ponemos a 1 para activar el funcionamiento como interrupciones de estas dos entradas y aprovechamos para inicializar los registros de interrupción asociados a estas entradas.

Estas interrupciones se activan cuando en el bit 4 y en bit 3 del TrisD tenemos un cambio de nivel alto a nivel bajo.

6.3.2 Funciones de configuración de salidas de la Dspic

Esta función es la encargada de configurar las salidas de la Dspic por las que enviaremos información de nuestra Dspic a la aplicación física (La maqueta).

Función Bobinas:

Esta función es la encargada de definir los bits 0 y 1 del puerto TrisF como salidas esto lo hacemos colocando estos bits a 0, ya que es la forma de indicar que son salidas.

Por otro lado aprovechamos para inicializar estas salidas a cero, es decir, las desactivamos de inicio para activarlas cuando nos sea conveniente.

6.3.3 Funciones de configuración de recursos de la Dspic

Esta función es la encargada de configurar el timer de la Dspic para poder trabajar a una frecuencia determinada.

Función T1_Program:

Esta función es la encargada de configurar el T1 de nuestra Dspic a modo de interrupciones solo cuando se produce el flanco de subida en nuestro oscilador y el periodo de cada pulso lo configuramos a 1 ms.

Función T1-clear:

Esta función es la encargada de poner el registro de interrupción a cero para al iniciar la ejecución del programa esperemos realmente a que se produce el primer flanco de subida para poder activar la interrupción y así evitar problemas al inicio del programa.

6.3.4 Funciones de configuración de salidas de la Dspic

Nuestras interrupciones están formadas por variables y registros específicos de la Dspic que estamos programando.

Las interrupciones las podemos dividir en distintos grupos:

1. Control del tiempo
2. Control de la velocidad

Control del tiempo

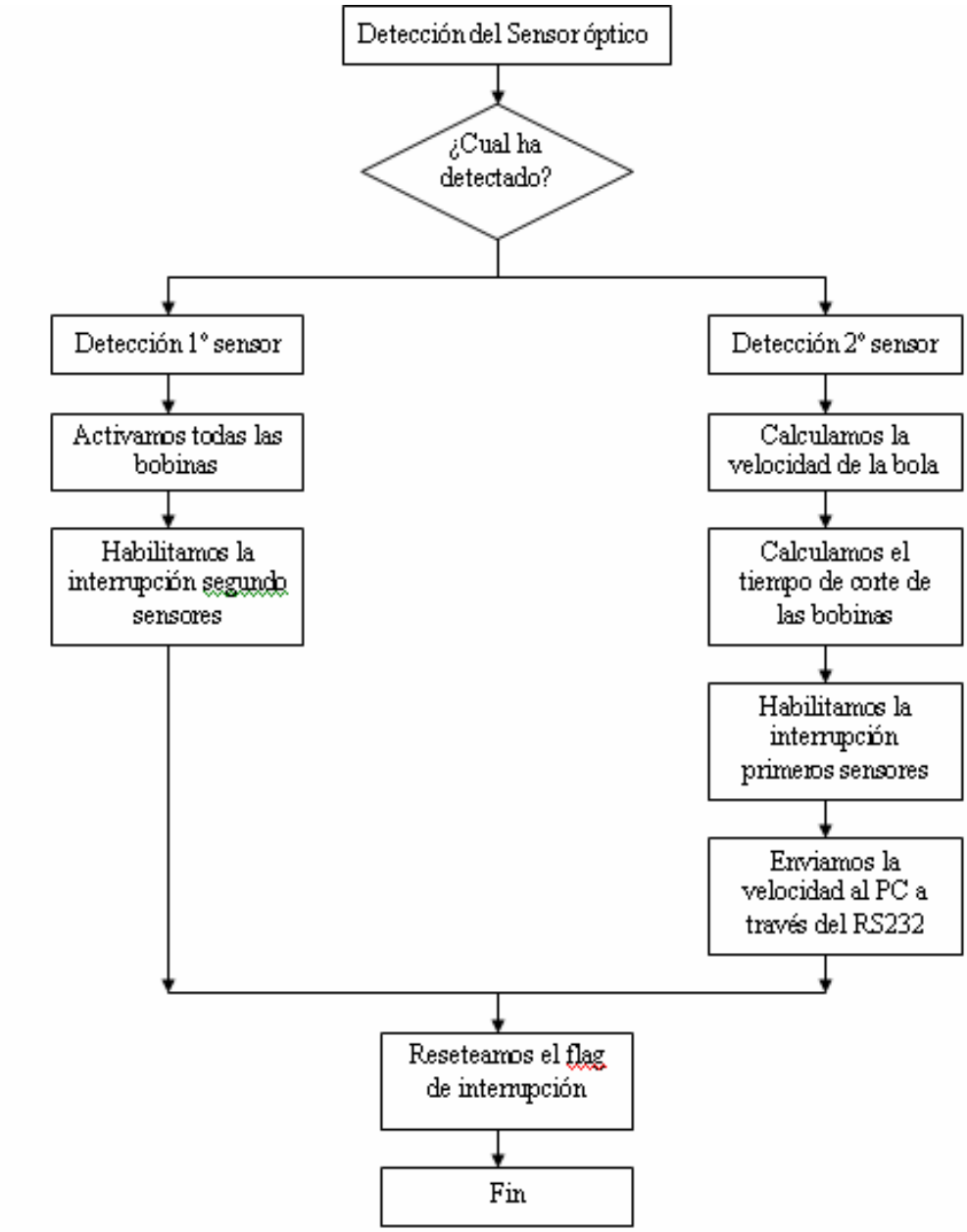
Esta interrupción es la encargada de llevar un control del tiempo en nuestra aplicación y cuando llegamos al tiempo en el que necesitamos desconectar la alimentación de las bobinas desactivamos las salidas RF0 y RF1 que son las que se encargan de alimentar las bobinas de nuestra aplicación.

Control de la velocidad

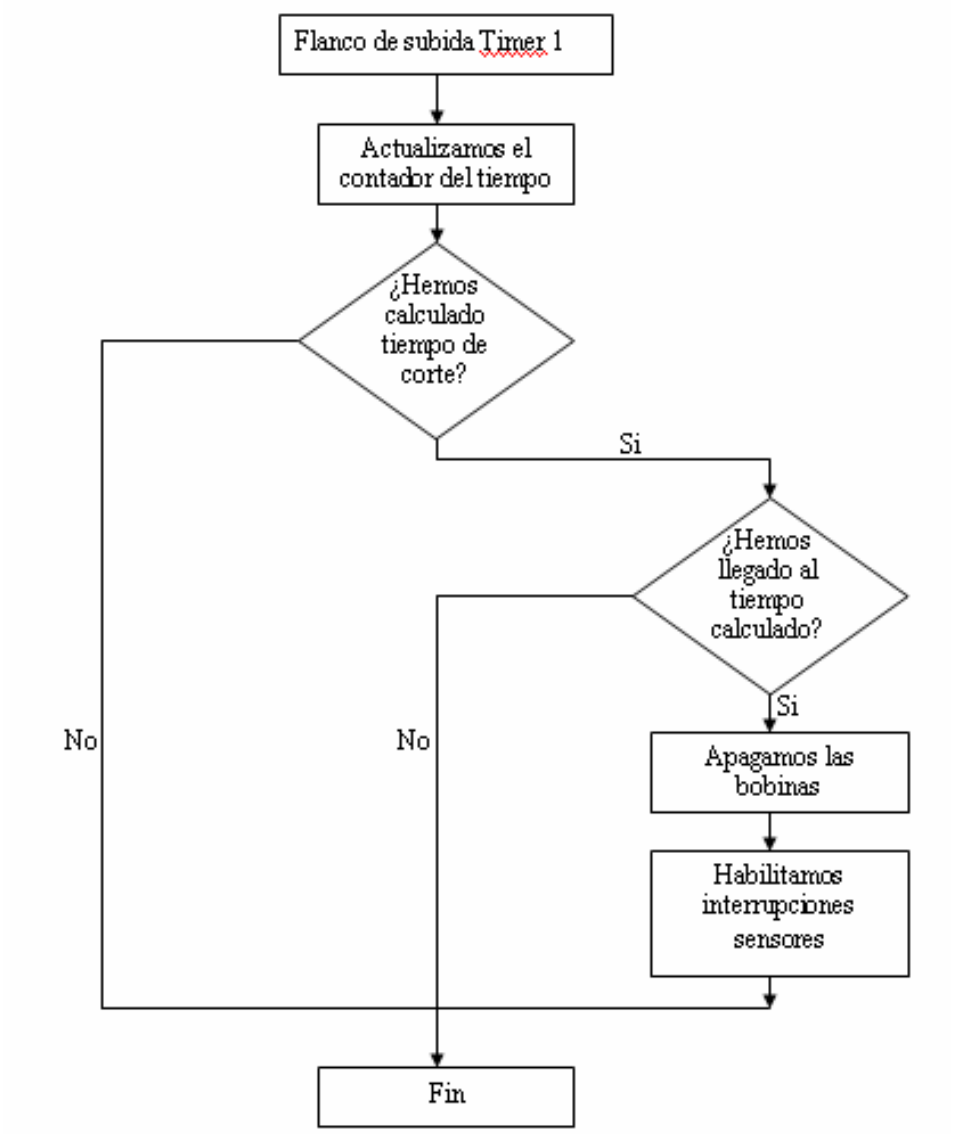
Esta interrupción se produce cada vez que hay cero en las entradas RD4 y RD5 de la Dspic que son las entradas encargadas de recibir la información de los sensores ópticos de la maqueta.
La función de esta interrupción es que cuando se produzca el corte del sensor óptico, si es el primero activa las bobinas y si es el segundo calcula la velocidad que lleva la bola y calcula el tiempo optimo para hacer la desconexión de las bobinas y este tiempo se pasa a la interrupción del timer 1 que es la que controla que realmente pase el tiempo calculado para desconectar las bobinas.

6.3.5 Diagramas de funcionamiento

Interrupción control de velocidad:



Interrupción control de tiempo:



Después de explicar todo el código para hacernos una idea de la configuración realizada en la placa e intentar disipar las dudas que se puedan producir al examinar los códigos adjuntados en los anexos, procedemos a explicar el último control realizado.

Este es un control realizado a tramos a partir de los puntos encontrados en el apartado anterior. La idea de este control era a partir de las tres franjas bien diferenciadas encontradas anteriormente donde se aceleraba la velocidad de la bola encontrar la función de cada tramo que nos permitiese calcular el tiempo de corte óptimo en función de la velocidad inicial de la bola cada vez que se encontrase dentro de uno de estos tramos y así conseguir acelerar la bola de una forma más continua y no tan esporádica como se había conseguido hasta ahora.

El resultado fue bueno tal y como se puede apreciar en los videos entregados junto esta memoria.

A continuación se muestra el código con las funciones halladas en cada uno de los tramos obtenidos en el apartado anterior.

```
ISR2 (_CNInterrupt)
{
PortRD4=PORTDbits.RD4;
PortRD5=PORTDbits.RD5;
if (PortRD4 != last_PortRD4)
{
        if (semaforo==0)
        {
            tiempo2=time;
            tiempodepaso=(tiempo2-tiempo1)*1e-3;
            velocidad=distanciasensores/tiempodepaso;
            velocidad1=velocidad*1000;
            my_velocidad=velocidad1;
            low_byte=(EE_UINT8)my_velocidad;
            high_byte=(EE_UINT8)(my_velocidad>>8);
            EE_UART1_Send(high_byte);
            EE_UART1_Send(low_byte);
            semaforo=1;
            cont=cont+1;
            tiempo=0;
            if (velocidad<0.17)
            {
                if ((velocidad>=0.068)&(velocidad<=0.073))
                {
                    t=((0.008751-(0.032*velocidad))/0.005);
                    t=t*1000;
                    tiempo=t;
                    m=1;
                    semaforo=1;
                    time=0;
                    paso=1;
                    LATFbits.LATF0 = 1;
```

```
        LATFbits.LATF1 = 1;
}
if ((velocidad>0.073)&(velocidad<=0.086))
{
        t=((0.022227-(0.076*velocidad))/0.013);
        t=t*1000;
        tiempo=t;
        m=1;
        semaforo=1;
        time=0;
        paso=1;
        LATFbits.LATF0 = 1;
        LATFbits.LATF1 = 1;
}
if ((velocidad>0.086)&(velocidad<=0.118))
{
        t=((0.054706-(0.187*velocidad))/0.032);
        t=t*1000;
        tiempo=t;
        m=1;
        semaforo=1;
        time=0;
        paso=1;
        LATFbits.LATF0 = 1;
        LATFbits.LATF1 = 1;
}
if ((velocidad>0.118)&(velocidad<=0.126))
{
        t=((0.014414-(0.53*velocidad))/0.008);
        t=t*1000;
        tiempo=t;
        m=1;
        semaforo=1;
        time=0;
        paso=1;
        LATFbits.LATF0 = 1;
        LATFbits.LATF1 = 1;
}
if ((velocidad>0.126)&(velocidad<=0.13))
{
        t=((0.007018-(0.025*velocidad))/0.004);
        t=t*1000;
        tiempo=t;
        m=1;
        semaforo=1;
        time=0;
        paso=1;
        LATFbits.LATF0 = 1;
        LATFbits.LATF1 = 1;
}
```

```
                    if ((velocidad>0.13)&(velocidad<=0.147))
                    {
                        t=((0.028884-(0.099*velocidad))/0.017);
                        t=t*1000;
                        tiempo=t;
                        m=1;
                        semaforo=1;
                        time=0;
                        paso=1;
                        LATFbits.LATF0 = 1;
                        LATFbits.LATF1 = 1;
                    }
                    if ((velocidad>0.147)&(velocidad<=0.169))
                    {
                        t=((0.021927-(0.023*velocidad))/0.022);
                        t=t*1000;
                        tiempo=t;
                        m=1;
                        semaforo=1;
                        time=0;
                        paso=1;
                        LATFbits.LATF0 = 1;
                        LATFbits.LATF1 = 1;
                    }
                    low_byte=(EE_UINT8)tiempo;
                    high_byte=(EE_UINT8)(tiempo>>8);
                    EE_UART1_Send(high_byte);
                    EE_UART1_Send(low_byte);

                }
                else
                {
                    t=((0.2128-(0.161*velocidad))/0.346);
                    t=t*1000;
                    tiempo=t;
                    m=1;
                    semaforo=1;
                    time=0;
                    paso=1;
                    LATFbits.LATF0 = 1;
                    LATFbits.LATF1 = 1;
                    low_byte=(EE_UINT8)tiempo;
                    high_byte=(EE_UINT8)(tiempo>>8);
                    EE_UART1_Send(high_byte);
                    EE_UART1_Send(low_byte);

                }
            }
}
if (PortRD5 != last_PortRD5)
```

```
{
        if (semaforo1==0)
        {
            tiempo1=time;
            cont1=cont1+1;
            semaforo1=1;
            semaforo=0;
        }
}
last_PortRD4=PortRD4;
last_PortRD5=PortRD5;
IFS1bits.CNIF=0;
}
```

7. RESULTADOS

A partir de los controles implementados se consiguieron encontrar una serie puntos óptimos de disparo, donde conseguíamos acelerar la bola dentro del tubo. Estos puntos óptimos de disparo se pueden ver en la siguiente tabla:

Velocidad inicio	Velocidad final	Tiempo de corte
0.019	0.426	1.607
0.068	1.925	1.315
0.073	1.677	1.283
0.086	1.625	1.207
0.118	0.658	1.02
0.126	0.171	0.967
0.13	0.565	0.942
0.147	0.192	0.843
0.169	0.504	0.82
0.172	0.658	0.654
0.177	0.838	0.666
0.187	0.327	0.61
0.189	0.945	0.593
0.191	1.13	0.58
0.233	0.712	0.335
0.246	1.083	0.256
0.339	0.962	0.428
0.39	0.896	0.433
0.426	1.061	0.416
0.429	1.019	0.386
0.481	0.634	0.39
0.485	0.776	0.388
0.49	1.019	0.415
0.509	0.65	0.377

Los colores de la tabla hacen referencia a los diferentes tramos que se implementaron en el control final.

A continuación podemos ver la gráfica de la función hallada.

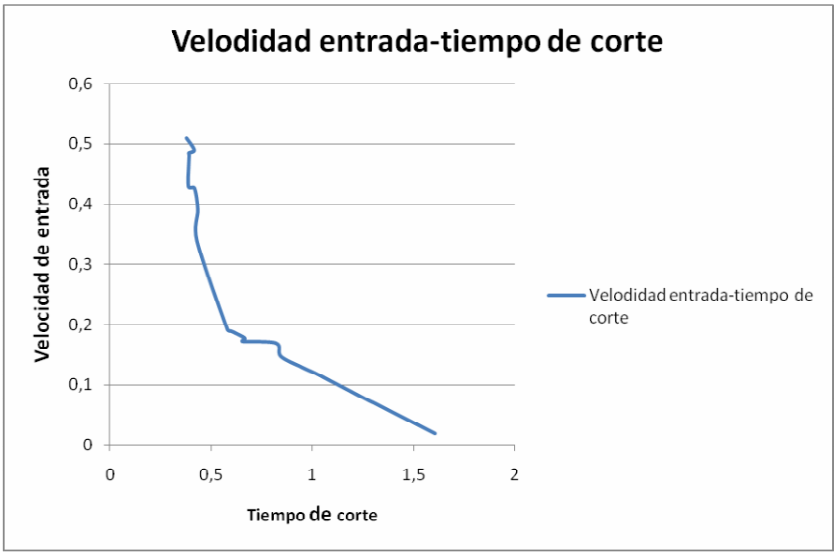


Figura 7.1. Función velocidad entrada-tiempo de corte

Como se puede apreciar en el gráfico anterior, la función tiende a parecerse a una exponencial. Para realizar el control cogimos los tramos más lineales de la función y realizamos el control linealizando tramo a tramo.

Los tramos están marcados en colores en la tabla anterior, si nos fijamos, hay tres valores sin color. Los valores de velocidad inicial 0,233 y 0,246 fueron descartados pues aunque son disparos donde aceleramos la bola, los puntos no siguen la tendencia del resto de la tabla. En el resto de puntos encontrados, a mayor velocidad de entrada menor tiempo de disparo. Los puntos descartados rompen esta tendencia y han sido eliminados. El valor 0,019 fue eliminado pues era una velocidad inesperadamente baja y no conseguimos ningún punto cercano a este. Por lo tanto no lo situamos dentro del primer tramo de control pues el siguiente punto encontrado distaba mucho de este.

En la siguiente figura observaremos el efecto que tiene en la función los dos puntos antes mencionados:

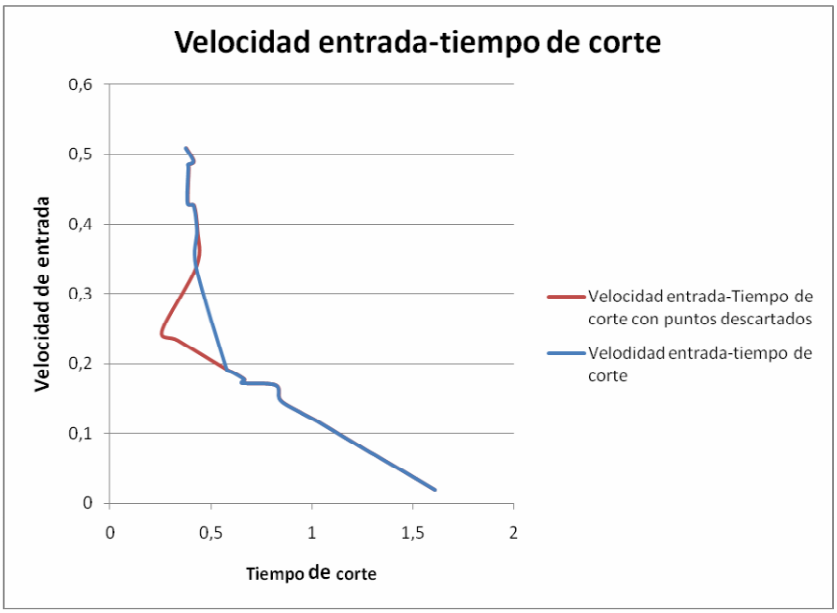


Figura 7.2. Función velocidad entrada-tiempo de corte con puntos descartados

El tiempo de corte disminuye al aumentar la velocidad, cosa totalmente normal pues a más velocidad la alimentación de la bobina se tiene que cortar antes para no frenar la bola. Pero con los puntos descartados la velocidad de entrada aumenta junto con el tiempo de corte, cosa en principio ilógica que nos ha hecho descartar estos puntos. Por otro lado, la función sin estos puntos tiende más a una función exponencial cosa mucho más lógica.

8. PRESUPUESTO

NÚM	UNID.	DESCRIPCIÓN	Cantidad	Precio €	Importe €
1.Material maqueta					
1.1	u	Base de madera, de 90x90cm	1,00	20,00	20,00
1.2	m	Tubo de PVC-reticulado, transparente de 25mm de diametro interior.	1,00	1,57	1,57
1.3	u	Sensor tipo reflector Pepperl+Fuchs GD18/GV18/73/120:	8,00	65,00	520,00
1.4	u	Bobinas de 1350 espiras de inductancia 100mH con resistencia de 24 ohms	4,00	35,00	140,00
1.5	u	Grapas de fijacion de tubo. De plastico negro	4,00	0,74	2,96
1.6	u	Grapas de fijacion de tubo. De plastico negro	4,00	0,74	2,96
1.7	u	Soporte para sensores. Trocolado para porder roscar los sensores.	8,00	5,87	46,96
1.8	m	Canaleta de plastico blanca de 20mm para pasar cableado de bobinas y sensores.	2,00	2,16	4,32
1.9	m	Cable de 1,5 mm per cablejar bobines i alimentacions.	15,00	0,40	6,00
1.10	u	Regletas para conexión electrica.	15,00	0,87	13,05
1.11	u	Material y construcción driver continua	1,00	70,00	70,00
1.12	u	Material y construcción driver alterna	1,00	70,00	70,00
1.13	u	Cable de comunicación serie	1,00	3,00	3,00
1.14	u	Placa flex de control basada en DSPIC	1,00	50,00	50,00
			TOTAL		**897,82 €**
2.Desarroyo					
2.1	u	Horas de construcción de la maqueta	100,00	30,00	3.000,00
2.1	u	Horas de ingenieria	250,00	60,00	15.000,00
			TOTAL		**18.000,00 €**
		1.Material maqueta			**897,82 €**
		2.Desarroyo			**18.000,00 €**
		TOTAL			**18.897,82 €**

9. CONCLUSIONES

Los objetivos marcados en el inicio del proyecto se han conseguido. Estos objetivos eran los siguientes:

- Comprobación de la viabilidad del proyecto mediante simulaciones del sistema con el programa Simulink (Matlab).
- Selección del material del proyecto.
- Construcción de la maqueta.
- Comprobación práctica de la aceleración de la bola.

En los diferentes apartados de esta memoria se puede apreciar como todos los puntos de los objetivos se han conseguido realizar de manera satisfactoria.

Se ha realizado la construcción de una maqueta partiendo de una idea sin ninguna referencia ni modelo a seguir, pues desconocemos si se ha construido anteriormente una maqueta de estas características. Se han realizado una serie de simulaciones para establecer la viabilidad del proyecto. Estas simulaciones han determinado que el proyecto era viable permitiéndonos emprender el montaje de la maqueta. Para construir la maqueta se han hecho diferentes pruebas hasta encontrar cada uno de los elementos más idóneos para su construcción y por último se han implementado una serie de controles con los cuales se ha conseguido llevar a la práctica la idea inicial de este proyecto.

Aprovechando este punto hacemos notar que en el montaje de la maqueta en numerosas ocasiones se ha tenido deshacer el trabajo hecho y volver hacia atrás para rectificar hipótesis que en un principio parecían inamovibles y que al final han resultado erróneas, como por ejemplo la utilización del Kernel de tiempo real de la DSPIC. Esto ha ralentizado mucho la realización de este proyecto hasta conseguir montar una maqueta para poder realizar futuras aplicaciones.

Las líneas de futuro son muchas, ya que lo que se ha pretendido con este proyecto no es la realización de una aplicación en concreto, sino la construcción de una maqueta y una estructura o estándar de programación para poder realizar diferentes aplicaciones al gusto del futuro usuario o programador. Por ejemplo se podría afinar la función de los puntos óptimos utilizando la Interpolación polinómica de Lagrange, restando un porcentaje de seguridad en cada punto conseguido para así poder implementar un control que mantenga la bola girando indefinidamente. También se podría implementar un control de velocidad fijando que la velocidad de entrada sea igual a la de salida, buscando los puntos de disparo para este caso en concreto y encontrando la función de control correspondiente.

10. BIBLIOGRAFIA

Programación placa FLEX

www.evidence.eu.com
www.microchip.com