Máster Universitario en Ingeniería Industrial

Escuela Politécnica Superior de Jaén

UNIVERSIDAD DE JAÉN
ESCUELA POLITÉCNICA SUPERIOR (JAÉN)

DISEÑO, CONSTRUCCIÓN Y CONTROL DE UNA PLANTA BALL AND PLATE

Alumno/a: **García Martín, Carlos**

Tutor/a: Prof. D. Elisabeth Estévez Estévez
Dpto.: Electrónica y Automática Industrial

Julio, 2018

Índice

1. INTRODUCCIÓN

La teoría moderna de control está basada en el conocimiento del comportamiento interno de los sistemas, reflejado en las variables que influyen en su dinámica. Estas variables constituyen el concepto de estado del sistema. El conocimiento de la evolución de todas las variables que influyen en la dinámica del sistema permite efectuar un control más potente de ésta y abordar el control de sistemas más complejos.

Dentro de la teoría moderna, los sistemas de equilibrio son uno de los problemas más desafiantes en el campo de control. El presente estudio se realiza con fines académicos al construir una plataforma mecatrónica a bajo costo que permita experimentar controladores que se encuentran actualmente en el campo industrial. Experimentar y ver la respuesta de los controladores que actualmente se implementan en procesos industriales reúne modelado, análisis y control en tiempo real obteniendo una validación de los controladores y hace más didáctico el aprendizaje. La idea de realizar una plataforma mecatrónica nace de la necesidad de visualizar físicamente la respuesta de los controladores diseñados en otro tipo de herramientas de laboratorio usadas para control diferente a las que comúnmente se implementan. El aprendizaje basado en proyectos es un método de enseñanza que implica a los estudiantes en la resolución de problemas de ingeniería reales. Si además, el proyecto está apoyado con equipos físicos, frente a los modelos de simulación, se proporciona al alumno una versión completa del problema a resolver, en lugar de limitar su trabajo a un modelo simplificado, a la vez que aumenta su motivación

El objetivo principal es disponer de una herramienta de alto nivel para el diseño de sistemas de control y realizar su modelado para una posterior simulación y generación de código que pueda ser evaluado sobre un robot físico de bajo coste. En concreto, el objetivo de este trabajo es presentar la experiencia de la creación de unos equipos de bajo coste programables con la herramienta MATLAB/Simulink utilizando Arduino como pasarela y que puedan ser utilizados para la realización de prácticas docentes.

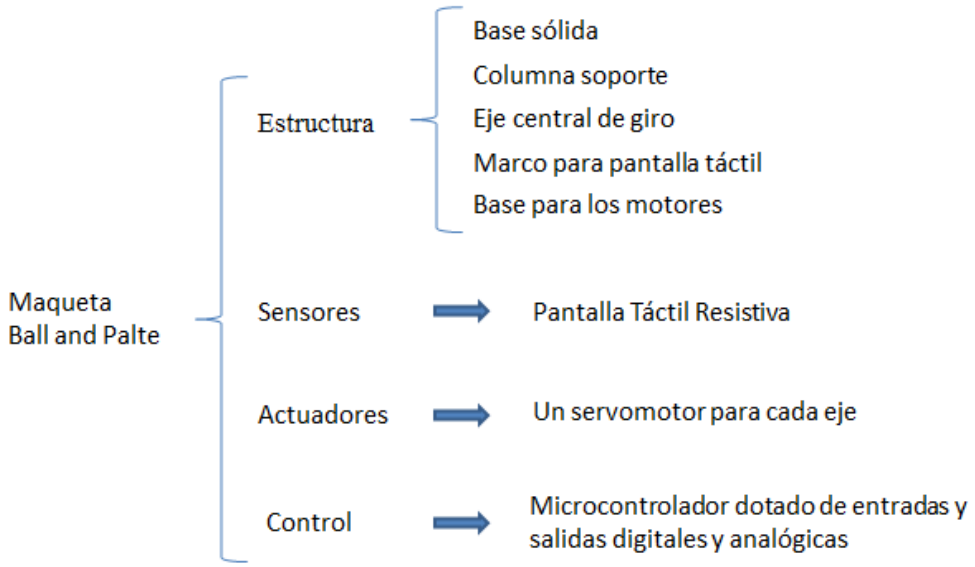
El sistema de balance Ball and Plate es una forma extendida del sistema Ball and Beam en la que una bola se desplaza por una viga, por tanto, el sistema tiene la libertad en la dirección que se desplaza la bola (un eje de coordenada), el movimiento es controlado por medio de un actuador que recibe las señales para el control del movimiento de la viga para

posicionar la bola. El sistema Ball and Plate es un sistema no - lineal multivariable. Está compuesto por una bola que rota alrededor de los ejes perpendiculares X e Y de un plato, de esta forma la bola se mueve bajo la influencia de la fuerza de gravedad sobre la superficie del plano. El objetivo de este proyecto es mover el plato para llevar la pelota a la posición deseada, es decir, controlar una bola que rueda libremente y fijarla en una posición específica predefinida con un error y en un tiempo mínimo. [1]

Este documento va a contener una visión detallada de cómo controlar un sistema Ball and Plate. Primero se hará un estudio teórico del modelado matemático obteniendo así el modelo de la planta del sistema para su posterior linealización. Se usará el modelo linealizado para diseñar el controlador del sistema y así poder simular los resultados usando el modelo no lineal del sistema y evaluar los resultados para su posterior diseño y control en tiempo real de la maqueta diseñada.

2. HARDWARE DEL SISTEMA

La maqueta realizada para poner en práctica el sistema estudiado deberá constar de las siguientes partes descritas a continuación:



Para el montaje del hardware del sistema se han utilizado los siguientes materiales que a continuación se van a describir con detalle.

- **Placa de acero** para la base del sistema. Se ha optado por este material ya que es bastante pesado y se necesita una estructura sólida para evitar vibraciones u otro tipo de perturbaciones que puedan afectar al sistema. En este caso se ha optado por un trozo de placa reciclado, se ha cortado a la medida necesaria y ha pintado para que tenga buen aspecto.

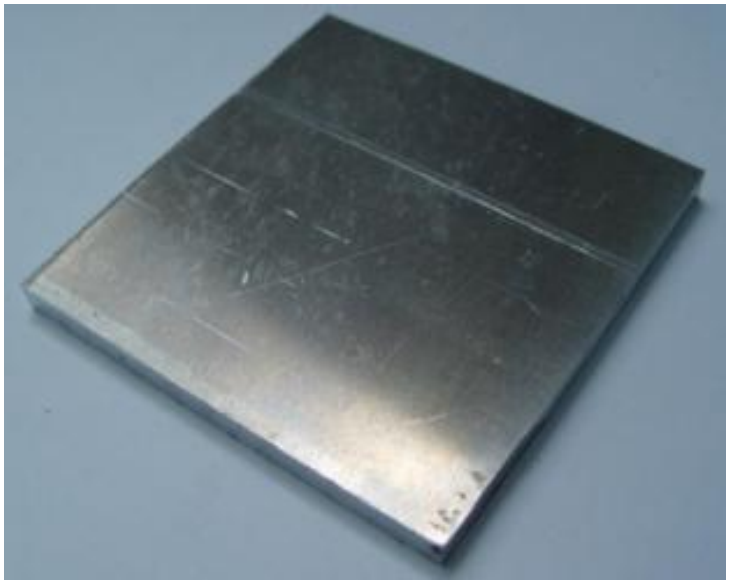


Figura 2.1. Placa de acero reciclada

- **Llave de tubo cilíndrico**. Estará sujeto en un extremo a la estructura de la base y por el otro extremo tendrá soldado un cardán. Según la altura del tubo habrá que determinar la altura a la que se colocan los motores.

Figura 2.2. Llave de tubo cilíndrica

- **Cardán**. Es un componente mecánico que permite unir dos ejes no coaxiales. Su objetivo es transmitir el movimiento de rotación desde un eje conductor a otro conducido a pesar de su no colinealidad [2]. Su funcionalidad en este proyecto será sujetar la plataforma en la que se va a apoyar la bola y permitir que pueda girar tanto en el eje X como en el eje Y. En un extremo irá soldado a la llave de tubo y por el otro extremo se ha soldado un tonillo que irá atornillado a la plataforma del sistema.

Figura 2.3. Cardán.

- **Pantalla táctil resistiva** de 4 hilos 12”. Será utilizada como plataforma para la bola pero también actuará como sensor para el control del sistema. Proporcionará las coordenadas en X e Y de la bola en cada momento. [3]

Figura 2.4. Pantalla Táctil resistiva de 4 hilos 12”.

- **Marco para la pantalla**. Marco diseñado a la medida exacta de la pantalla táctil utilizada, consta de un rail de la medida del grosor de la pantalla para que esta encaje perfectamente y quede bien sujeta. También se le ha añadido un pasante en el centro que es por donde se introducirá el tornillo soldado en el cardán. Se ha diseñado con software CAD 3D y se ha imprimido con la impresora 3D.

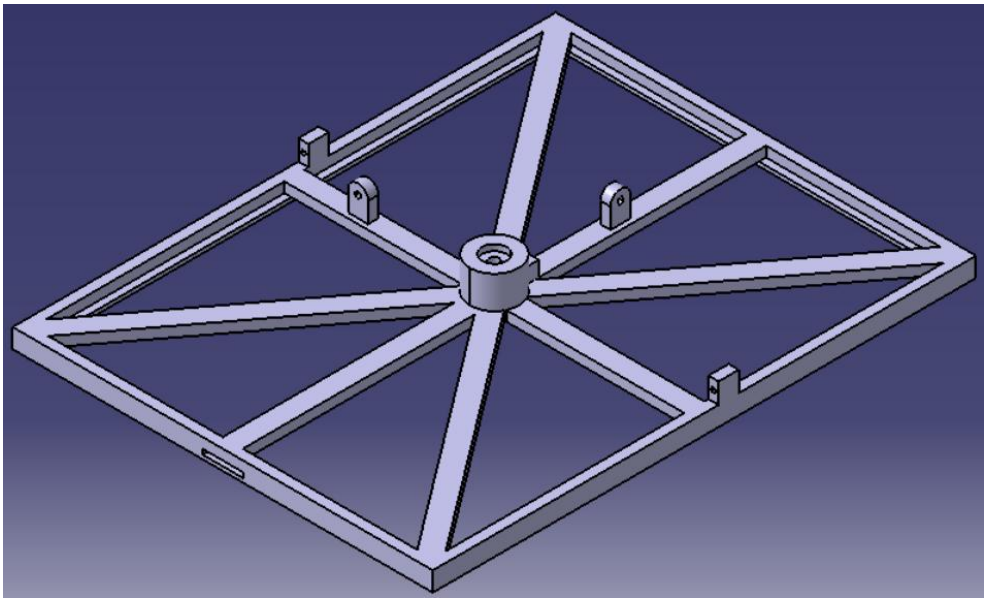


Figura 2.5. Diseño Marco Pantalla Táctil.

- **Servomotor** digital estándar HS-5485HB. Será el actuador del sistema y se necesitará dos servomotores, uno para cada eje de giro de la plataforma. Estarán colocados perpendicularmente uno del otro.

Figura 2.6. Servomotor digital estándar.

- **Soporte para los motores**. Es un soporte diseñado para elevar el motor a una altura deseada, con una base diseñada a la medida de la base de los motores para que queden perfectamente apoyados y preparado para que puedan atornillarse a él y se queden fijos. Este soporte irá atornillado a la estructura de la base del proyecto con dos tornillos para así evitar que pueda girar y se mantenga siempre en la misma posición. Se ha diseñado con software CAD 3D y se ha imprimido con la impresora 3D.

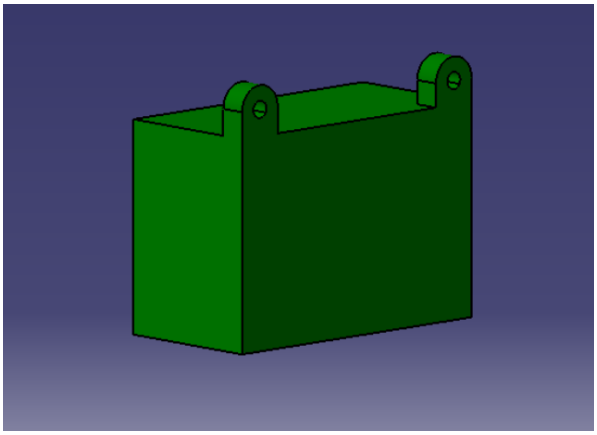


Figura 2.7. Diseño soporte para servomotor

- **Aluminum Servo Arm.** Elemento mecánico que será la biela del motor que permitirá transmitir un giro de 1,5 cm de radio a otro elemento. Se colocará uno por cada motor que se disponga, en este caso se necesitarán dos.

Figura 2.8. Aluminum Servo Arm.

- **Servo Rod**. Barra regulable con un máximo de 10 cm que está articulada en los extremos mediante rótulas giratorias. Estará conexionado a un extremo de la biela del motor y por el otro extremo a la panta táctil. Su función será transmitir el giro proporcionado por el motor a la plataforma del sistema y así hacer que esta gire también. Se dispondrá de una barra por cada motor.

Figura 2.9. Servo Rod.

- **Arduino DUE.** Arduino Due es una placa de microcontrolador basada en la CPU Atmel SAM3X8E ARM Cortex-M3 con núcleo ARM de 32 bits. Tiene 54 pines digitales de entrada / salida (de los cuales 12 se pueden usar como salidas PWM), 12 entradas analógicas, 4 UART (puertos serie de hardware), un reloj de 84 MHz, una conexión compatible con OTG USB, 2 DAC (digital a analógico), 2 TWI, un conector de alimentación, un encabezado SPI, un encabezado JTAG, un botón de reinicio y un botón de borrado. Será utilizado como controlador y como pasarela.

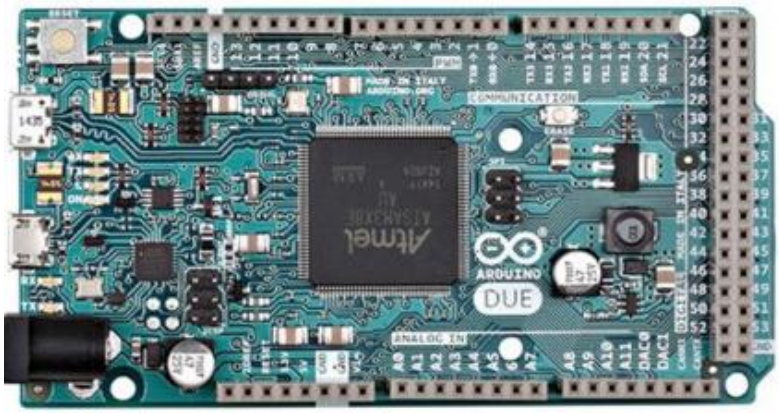


Figura 2.10. Arduino DUE.

- **Bola.** La bola utilizada será una bola de ratón, aunque podrá utilizarse cualquier otra bola siempre y cuando se tenga en cuenta sus parámetros de peso y tamaño.

Figura 2.11. Bola de Ratón.

Una vez realizado el ensamblaje de todo el conjunto de piezas descritas, el resultado final del hardware completo del sistema se muestra en las *Figura 2.12-2.13*.

Figura 2.12. Hardware real del sistema.

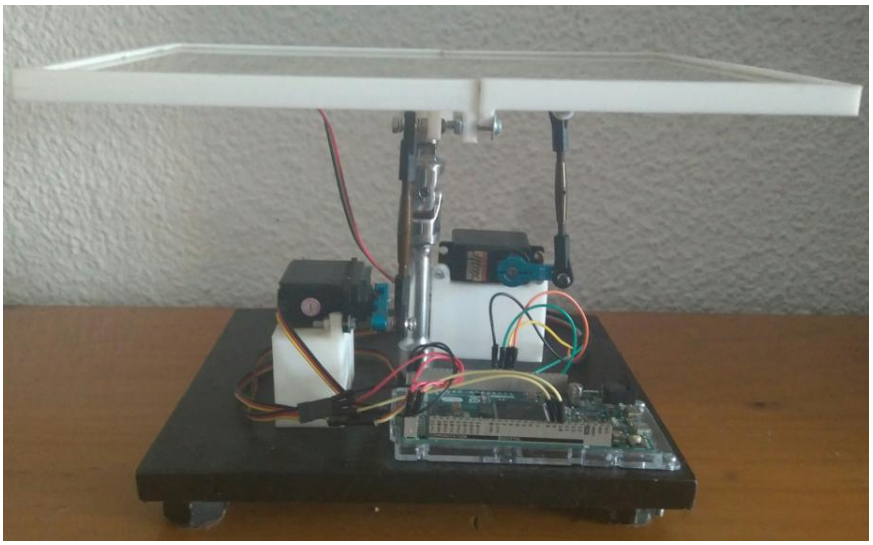


Figura 2.13. Hardware real del sistema visto de frente.

3. MODELADO MATEMÁTICO

A continuación se va a realizar el desarrollo del modelo matemático del sistema planteado. En este caso se asumirán las siguientes condiciones:

- La bola es completamente simétrica y homogénea.
- No hay rozamiento de la bola.
- Todas las fricciones son despreciadas.
- La bola y el plato están en contacto todo el tiempo.

Es importante identificar cuáles serán las variables a controlar del sistema, para ello se tendrán que definir los grados de libertad existentes que serán el número de coordenadas independientes (escalares) necesarias para determinar simultáneamente la posición de cada elemento en un sistema dinámico:

- Posición de la bola respecto al eje x y al eje y. [x_b , y_b]
- Ángulo de inclinación del plato respecto al eje x y al eje y. [α , β]

Para realizar la demostración matemática de este proceso se va a aplicar la ecuación general de Euler-Lagrange. [4]

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}i} - \frac{dL}{dqi} = Qi \qquad (3.1)$$

Dónde:

Q → Fuerza compuesta.

qi → Posición en dirección i.

Se puede desarrollar esta ecuación usando el Lagrangiano. Se trata de una función escalar a partir de la cual se puede obtener la evolución temporal, las leyes de conservación y otras propiedades importantes de un sistema dinámico. De hecho, en física moderna el Lagrangiano se considera un operador fundamental para describir un sistema físico. La trayectoria de un objeto es obtenida encontrando la trayectoria que minimiza la acción, que es la integral del Lagrangiano en el tiempo; siendo éste la energía cinética del objeto menos la energía potencial del mismo.

$$L = T - V \tag{3.2}$$

Dónde:

L → Lagangiano.

T → Energía cinética del sistema.

V→ Energía potencial del sistema.

Sustituyendo el Lagrangiano en la *Ecuación 2.1* se obtiene la siguiente ecuación.

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}i} - \left(\frac{\partial T}{\partial qi} - \frac{\partial V}{\partial qi}\right) = Qi \;\; ; \tag{3.3}$$

A continuación se muestran las ecuaciones generales de la energía cinética y de la energía potencial de las que se partirá para aplicarlas según las necesidades del sistema estudiado y que posteriormente se estudiarán para el caso de la bola y el plato del sistema.

$$T = \frac{1}{2}\, m\, v^2 \tag{3.4}$$

$$V = mgh \tag{3.5}$$

Dónde:

m → Masa del elemento a estudiar.

v → Velocidad del elemento a estudiar.

h → Altura a la que se encuentra dicho elemento.

g → Aceleración de la gravedad.

Habrá que definir ahora la ecuación de la energía cinética de la bola en cuestión, que se compone de energía traslacional y de su rotación con respecto a su centro de masa. Para ello habrá que tener en cuenta que el sistema tendrá velocidad rotacional y velocidad translacional tanto en el eje x como en el eje y.

$$T_b = \frac{1}{2}\, m_b\left(\dot{x_b}^{\,2} + \dot{y_b}^{\,2}\right) + \frac{1}{2} I_b\left(\omega_x^2 + \omega_y^2\right) \qquad (3.6)$$

Dónde:

$T_b \rightarrow$ Energía Cinética de la bola

$m_b \rightarrow$ Masa de la bola

$I_b \rightarrow$ Momento de inercia de la bola

$\dot{x_b} \rightarrow$ Velocidad translacional de la bola con respecto al eje x

$\dot{y_b} \rightarrow$ Velocidad translacional de la bola con respecto al eje y

$\omega_x \rightarrow$ Velocidad rotacional de la bola respecto al eje x

$\omega_y \rightarrow$ Velocidad rotacional de la bola respecto al eje y

Para simplificar esta ecuación se podrá usar la siguiente relación entre velocidades.

$$\dot{x_b} = r_b\, \omega_y \quad ; \quad \dot{y_b} = r_b\, \omega_x$$

$$T_b = \frac{1}{2}\left[m_b\left(\dot{x_b}^{\,2} + \dot{y_b}^{\,2}\right) + \frac{I_b}{r_b^{\,2}}\left(\dot{x_b}^{\,2} + \dot{y_b}^{\,2}\right)\right] = \frac{1}{2}\left(m_b + \frac{I_b}{r_b^{\,2}}\right)\left(\dot{x_b}^{\,2} + \dot{y_b}^{\,2}\right) \qquad (3.7)$$

Dónde:

$r_b \rightarrow$ Radio de la bola

La energía cinética del plato, considerando que la bola es una masa puntual que se coloca en el punto $[x_b\ ,\ y_b]$, consiste en su energía de rotación con respecto al centro de masas.

$$T_p = \frac{1}{2}(I_p + I_b)\left(\dot{\alpha}^{\,2} + \dot{\beta}^2\right) + \frac{1}{2}\, m_b(x_b\dot{\alpha}\ + y_b\,\dot{\beta})^2\ ;$$

$$T_p = \frac{1}{2}(I_p + I_b)(\dot{\alpha}^{\,2} + \dot{\beta}^{\,2}) + \frac{1}{2}\, m_b\ (x_b{}^2\dot{\alpha}^{\,2} + y_b{}^2\dot{\beta}^{\,2} + 2\ x_b\ \dot{\alpha}\ y_b\ \dot{\beta}) \qquad \textit{(3.8)}$$

Dónde:

$T_p \rightarrow$ Energía cinética del plato

$I_p \rightarrow$ Momento de inercia del plato

$\dot{\alpha} \rightarrow$ Velocidad rotacional del plato con respecto al eje x

$\dot{\beta} \rightarrow$ Velocidad rotacional del plato con respecto al eje y

Una vez obtenida la energía cinética de la bola *Ecuación 3.7* y la energía cinética del plato *Ecuación 3.8* por separado, se definirá la energía cinética del sistema.

$$T = T_b + T_p$$

$$T = \frac{1}{2}\left(m_b + \frac{I_b}{r_b{}^2}\right)(\dot{x_b}^{\,2} + \dot{y_b}^{\,2}) + \frac{1}{2}(I_p + I_b)(\dot{\alpha}^2 + \dot{\beta}^2) + \frac{1}{2}\, m_b\ (x_b{}^2\,\dot{\alpha}^2 + y_b{}^2\,\dot{\beta}^2 + 2\ x_b\dot{\alpha}\ y_b\dot{\beta})$$

(3.9)

Una vez definida la ecuación de la energía cinética *Ecuación 3.9*, se va a plantear la energía potencial del sistema. La energía potencial de la bola con respecto a la horizontal del plano en el centro de inclinación del plato puede ser calculado como se muestra en la *Ecuación 3.10*.

$$V = m_b g h = m_b g(x_b \sin\alpha + y_b\ \sin\beta) \qquad \textit{(3.10)}$$

Dónde:

$V \rightarrow$ Energía potencial de la bola respecto a la horizontal del plano

$h \rightarrow$ Altura de la bola respecto al eje horizontal del plano

Una vez hecho el desarrollo de la energía cinética y potencial, se podrá resolver las derivadas parciales que aparecen en la ecuación inicial (*Ecuación 3.3*) con respecto a las variables del sistema $[x_b\,, y_b, \alpha, \beta]$ y sus derivadas $[\dot{x_b}\,, \dot{y_b}, \dot{\alpha}, \dot{\beta}]$.

$$\frac{\partial T}{\partial \dot{\alpha}} = \left(I_p + I_b\right)\dot{\alpha} + m_b x_b\left(x_b\,\dot{\alpha} + y_b\,\dot{\beta}\right); \quad \frac{\partial T}{\partial \alpha} = 0; \quad \frac{\partial V}{\partial \alpha} = m_b\, g\, x_b \cos\alpha\ ; \qquad (3.11)$$

$$\frac{\partial T}{\partial \dot{\beta}} = \left(I_p + I_b\right)\dot{\beta} + m_b y_b\left(x_b\,\dot{\alpha} + y_b\,\dot{\beta}\right); \quad \frac{\partial T}{\partial \beta} = 0; \quad \frac{\partial V}{\partial \beta} = m_b\, g\, y_b \cos\beta; \qquad (3.12)$$

$$\frac{\partial T}{\partial \dot{x_b}} = \left(m_b + \frac{I_b}{{r_b}^2}\right)\dot{x_b}\,; \quad \frac{\partial T}{\partial x_b} = m_b\,\dot{\alpha}\left(x_b\,\dot{\alpha} + y_b\,\dot{\beta}\right); \quad \frac{\partial V}{\partial x_b} = m_b\, g \sin\alpha\,; \qquad (3.13)$$

$$\frac{\partial T}{\partial \dot{y_b}} = \left(m_b + \frac{I_b}{{r_b}^2}\right)\dot{y_b}\,; \quad \frac{\partial T}{\partial y_b} = m_b\,\dot{\beta}\left(x_b\,\dot{\alpha} + y_b\,\dot{\beta}\right); \quad \frac{\partial V}{\partial y_b} = m_b\, g \sin\beta\,; \qquad (3.14)$$

Por tanto, se procederá a sustituir estos valores en la ecuación genérica de Euler-Lagrange (*Ecuación 3.3*) y así se llegará a obtener el sistema de ecuaciones correspondiente al sistema estudiado. Para ello habrá que derivar la primera derivada parcial obtenida anteriormente en cada variable y sustituir el resto de derivadas parciales.

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\alpha}} - \left(\frac{\partial T}{\partial \alpha} - \frac{\partial V}{\partial \alpha}\right) = \frac{d}{dt}\frac{\partial T}{\partial \dot{\alpha}} + \frac{\partial V}{\partial \alpha} = \left(I_p + I_b\right)\ddot{\alpha} + m_b {x_b}^2\ddot{\alpha} + 2\, m_b x_b \dot{x_b}\dot{\alpha} + m_b x_b\, y_b\,\ddot{\beta} + m_b \dot{x_b}\, y_b\,\dot{\beta} + m_b\, g\, x_b \cos\alpha = 0 \qquad (3.15)$$

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\beta}} - \left(\frac{\partial T}{\partial \beta} - \frac{\partial V}{\partial \beta}\right) = \frac{d}{dt}\frac{\partial T}{\partial \dot{\beta}} + \frac{\partial V}{\partial \beta} = \left(I_p + I_b\right)\ddot{\beta} + m_b {y_b}^2\ddot{\beta} + 2\, m_b y_b \dot{y_b}\,\dot{\beta} + m_b x_b\, y_b\,\ddot{\alpha} + m_b \dot{y_b}\, x_b\dot{\alpha} + m_b\, g\, y_b \cos\beta = 0 \qquad (3.16)$$

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{x_b}} - \left(\frac{\partial T}{\partial x_b} - \frac{\partial V}{\partial x_b}\right) = \frac{d}{dt}\frac{\partial T}{\partial \dot{x_b}} - \frac{\partial T}{\partial x_b} + \frac{\partial V}{\partial x_b} = \left(m_b + \frac{I_b}{{r_b}^2}\right)\ddot{x_b} - m_b\,\dot{\alpha}\left(x_b\,\dot{\alpha} + y_b\,\dot{\beta}\right) + m_b\,g\sin\alpha = 0; \quad (3.17)$$

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{y_b}} - \left(\frac{\partial T}{\partial y_b} - \frac{\partial V}{\partial y_b}\right) = \frac{d}{dt}\frac{\partial T}{\partial \dot{y_b}} - \frac{\partial T}{\partial y_b} + \frac{\partial V}{\partial y_b} = \left(m_b + \frac{I_b}{{r_b}^2}\right)\ddot{y_b} - m_b\,\dot{\beta}\left(x_b\,\dot{\alpha} + y_b\,\dot{\beta}\right) + m_b\,g\sin\beta = 0; \quad (3.18)$$

Agrupando y simplificando estas ecuaciones se llegará a obtener las ecuaciones diferenciales no lineales que definen el sistema Ball and Plate.

$$\left(I_p + I_b + m_b{x_b}^2\right)\ddot{\alpha} + 2\,m_b x_b \dot{x_b}\dot{\alpha} + m_b x_b\,y_b\,\ddot{\beta} + m_b \dot{x_b}\,y_b\,\dot{\beta} + m_b\,g\,x_b\cos\alpha = 0 \quad (3.19)$$

$$\left(I_p + I_b + m_b{y_b}^2\right)\ddot{\beta} + 2\,m_b y_b \dot{y_b}\,\dot{\beta} + m_b x_b\,y_b\,\ddot{\alpha} + m_b \dot{y_b}\,x_b\dot{\alpha} + m_b\,g\,y_b\cos\beta = 0 \quad (3.20)$$

$$\left(m_b + \frac{I_b}{{r_b}^2}\right)\ddot{x_b} - m_b\left(x_b\,\dot{\alpha}^2 + y_b\,\dot{\alpha}\,\dot{\beta}\right) + m_b\,g\sin\alpha = 0 \quad (3.21)$$

$$\left(m_b + \frac{I_b}{{r_b}^2}\right)\ddot{y_b} - m_b\left(x_b\,\dot{\alpha}\,\dot{\beta} + y_b\,\dot{\beta}^2\right) + m_b\,g\sin\beta = 0 \quad (3.22)$$

4. IDENTIFICACIÓN DE LA DINÁMICA DE LA PLANTA DEL SISTEMA

En este apartado se va a proceder a linealizar el modelo matemático desarrollado para obtener la dinámica de cada una de las partes del sistema con el objetivo de obtener la dinámica de la planta completa en lazo abierto y con ello poder diseñar el controlador adecuado.

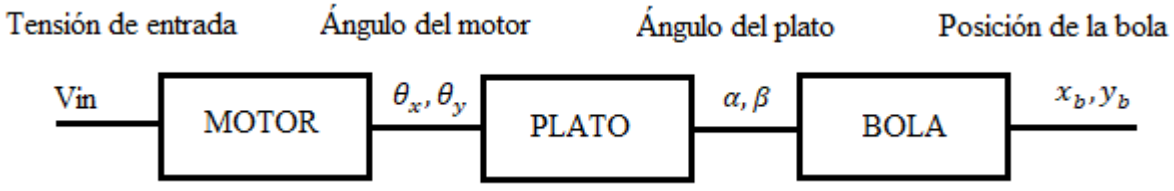


Figura 4.1. Dinámica de la planta del sistema

4.1. VARIABLES SIGNIFICATIVAS

Las variables significativas del nuestro serán las indicadas anteriormente en el la *Figura 4.1*.

- Variable controlada: Ángulo de giro de los motores (θ_x, θ_y).
- Variable manipulada: Posición de la bola en cada eje (x_b, y_b).
- Variable de proceso manipulada: Ángulo de inclinación del plato en cada eje (α, β).
- Posibles perturbaciones: posible inclinación en la base de apoyo que pueda perturbar la posición de equilibrio de la bola.

4.2. DINÁMICA DE LOS MOTORES

Se va a disponer de un motor de CC controlado por armadura cuyo sistema electromecánico es como el que se muestra en la *Figura 4.2.1*.

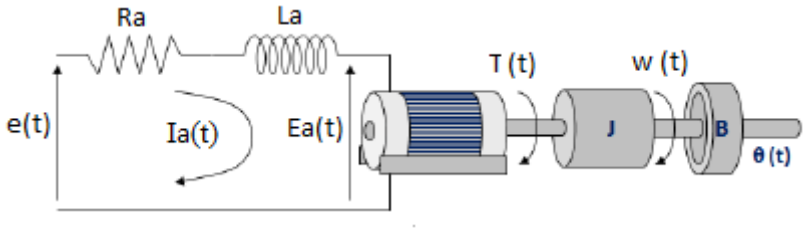


Figura 4.2.1. Esquema electromecánico de un motor CC.

El motor utiliza la corriente de armadura, entre otras cosas, como variable de control. El campo del estator puede establecerse mediante una corriente de bobina de campo. Cuando hay una corriente constante en la bobina de campo, se pueden deducir las siguientes ecuaciones.

$$V(t) = R_a I_a(t) + L_a \frac{di_a(t)}{dt} + E_a(t)\,; \qquad E_a(t) = K_b \frac{d\theta(t)}{dt}$$

$$e(t) = R_a I_a(t) + L_a \frac{di_a(t)}{dt} + K_b \frac{d\theta(t)}{dt} \qquad (4.2.1)$$

Dónde:

$e(t)$ → Tensión de alimentación.

R_a → Resistencia de armadura.

L_a → Bobina.

$I_a(t)$ → Intensidad de armadura.

$E_a(t)$ → Tensión de armadura.

$\theta(t)$ → Ángulo de giro del motor.

K_b → Constante.

$$T(t) = J\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta(t)}{dt}; \qquad T(t) = K_p\, I_a(t)$$

$$K_p\, I_a(t) = J\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta(t)}{dt} \tag{4.2.2}$$

Dónde:

J → Momento de Inercia.

B → Amortiguamiento de la torsión.

K_p → Constante de torque del motor.

Al despejar $I_a(t)$ en la *Ecuación 4.2.1* y sustituirla en la *Ecuación 4.2.2* se obtendrá una sola ecuación que relaciona la variable de entrada al motor, en este caso sería la tensión e(t), con la variable de salida del motor que sería el ángulo de giro θ(t). Aplicando el método de Laplace a dicha ecuación, se obtendrá la función de transferencia que define la dinámica del sistema del motor.

$$G(s) = \frac{\theta(s)}{E(s)} = \frac{K_p}{s[L_a\, Js^2 + (L_aB + R_a\, J)s + R_aB + K_bK_p]} \tag{4.2.3}$$

Se va a proceder a hacer el siguiente despeje en la *Ecuación 4.2.3* para simplificar la función de transferencia obtenida y que quede de una forma más ordenada.

$$K_m = \frac{K_p}{R_a B + K_b K_p} \tag{4.2.4}$$

Dónde:

$K_m \rightarrow$ Ganancia del Motor.

$$T_m = \frac{R_a J}{R_a B + K_b K_p} \tag{4.2.5}$$

Dónde:

$T_m \rightarrow$ Constante de tiempo del Motor.

Por tanto, la dinámica del motor será la que se muestra a continuación en la *Ecuación 4.2.6*.

$$G_M(s) = \frac{\theta(s)}{E(s)} = \frac{K_m}{s[T_m s + 1]} \tag{4.2.6}$$

4.3. DINÁMICA DEL PLATO

Las ecuaciones que definen la dinámica del plato en el sistema son las ecuaciones obtenidas en el modelo matemático *(Ecuación 3.19-3.22).* Se trata de ecuaciones no lineales que deberán ser linealizadas para para poder obtener la dinámica del plato. Para ello habrá que tener en cuenta las siguientes consideraciones:

- El ángulo de inclinación del plato será muy pequeño, por tanto se podrá considerar esta aproximación:

$$\alpha \ll 1 \rightarrow \sin\alpha = \alpha$$
$$\beta \ll 1 \rightarrow \sin\beta = \beta \tag{4.3.1}$$

Aplicando estas consideraciones se podrá simplificar las ecuaciones hasta quedar de la siguiente forma.

$$\left(m_b + \frac{I_b}{{r_b}^2}\right)\ddot{x_b} - m_b\left(x_b\,\dot{\alpha}^2 + y_b\,\dot{\alpha}\,\dot{\beta}\right) + m_b\,g\alpha = 0 \qquad (4.3.2)$$

$$\left(m_b + \frac{I_b}{{r_b}^2}\right)\ddot{y_b} - m_b\left(x_b\,\dot{\alpha}\,\dot{\beta} + y_b\,\dot{\beta}^2\right) + m_b\,g\beta = 0 \qquad (4.3.3)$$

Una vez linealizadas estas ecuaciones se podrá definir la dinámica del plato utilizando el siguiente método.

$$\dot{X} = Ax + Bu$$

Donde,

$$\dot{X} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \end{bmatrix};\ X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} x_b \\ \dot{x_b} \\ \alpha \\ \dot{\alpha} \\ y_b \\ \dot{y}_b \\ \beta \\ \dot{\beta} \end{bmatrix};\ U = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \ddot{\alpha} \\ \ddot{\beta} \end{bmatrix}$$

$$\dot{x}_1 = \dot{x_b} = x_2 \qquad (4.3.4)$$

$$\begin{aligned} \dot{x}_2 = \ddot{x}_b &= \frac{m_b}{m_b + \frac{I_b}{r_b}}\left(x_b\dot{\alpha}^2 + y_b\dot{\alpha}\dot{\beta} - g\alpha\right) \\ &= \frac{m_b}{m_b + \frac{I_b}{r_b}}\left(x_1\,{x_4}^2 + x_5\,x_4\,x_8 - g x_3\right) \end{aligned} \qquad (4.3.5)$$

$$\dot{x}_3 = \dot{\alpha} = x_4 \qquad (4.3.6)$$

$$\dot{x}_4 = \ddot{\alpha} = u_x \qquad (4.3.7)$$

$$\dot{x}_5 = \dot{y_b} = x_2 \tag{4.3.8}$$

$$\begin{aligned}\dot{x}_6 = \ddot{y}_b &= \frac{m_b}{m_b + \frac{I_b}{r_b}} \left(y_b \dot{\beta}^2 + x_b \dot{\alpha} \dot{\beta} - g\beta\right) \\ &= \frac{m_b}{m_b + \frac{I_b}{r_b}} (x_1 {x_4}^2 + x_5 x_4 x_8 - g x_7)\end{aligned} \tag{4.3.9}$$

$$\dot{x}_7 = \dot{\beta} = x_8 \tag{4.3.10}$$

$$\dot{x}_8 = \ddot{\beta} = u_y \tag{4.3.11}$$

Para hallar $\dot{x}_1$, $\dot{x}_3$, $\dot{x}_5$, $\dot{x}_7$ simplemente se ha tomado el valor correspondiente en la matriz X, para hallar $\dot{x}_5$, $\dot{x}_8$ se ha hecho lo mismo pero en este caso el valor obtenido corresponde a la matriz U y para $\dot{x}_2$, $\dot{x}_4$ el valor obtenido no corresponde con ningún valor de las matrices anteriores, entonces se ha obtenido despejando $\ddot{\alpha}$ y $\ddot{\beta}$ de la *(Ecuación 4.3.2)* y *(Ecuación 4.3.3)* respectivamente. El sistema matricial quedará entonces expresado de la siguiente forma.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{m_b}{m_b + \frac{I_b}{r_b}} (x_1 {x_4}^2 + x_5 x_4 x_8 - g x_3) \\ x_4 \\ 0 \\ x_6 \\ \frac{m_b}{m_b + \frac{I_b}{r_b}} (x_5 {x_8}^2 + x_1 x_4 x_8 - g x_7) \\ x_8 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix}$$

Se puede considerar también que la velocidad rotacional del plato es muy pequeña ($\dot{\alpha} \ll 1 \; y \; \dot{\beta} \ll 1$), por tanto se podrá omitir y así separar el sistema en dos subsitemas de control independientes para cada eje.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{m_b}{m_b + \frac{I_b}{r_b}} (x_1 {x_4}^2 - g x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} [\, u_x \,]$$

$$\begin{bmatrix} \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \end{bmatrix} = \begin{bmatrix} x_6 \\ \frac{m_b}{m_b + \frac{I_b}{r_b}} (x_5 {x_8}^2 - g x_7) \\ x_8 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} [\, u_y \,]$$

Con estos dos subsistemas habrá quedado definida la dinámica del plato del sistema para cada eje.

Por otro lado, para este caso en concreto existe otra forma mucho más sencilla para hallar la dinámica del plato. Se puede simplificar todo este proceso relacionando el ángulo de giro del motor con el ángulo de inclinación del plato, ya que el motor es el actuador que va a hacer que el plato se mueva y por tanto se puede afirmar que el ángulo del plato depende del ángulo de giro del motor como se muestra en la *Figura 4.3.1*. [5]

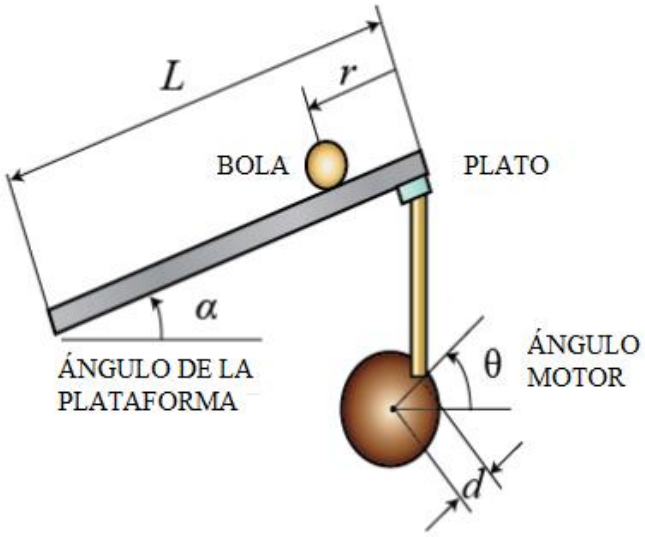


Figura 4.3.2. Plano de perfil del sistema Ball and Plate.

La imagen refleja cómo sería el proyecto en un plano de perfil. Puede aplicarse tanto para el eje x como para el eje y ya que la plataforma utilizada es simétrica, los motores son iguales y están colocados a la misma distancia del eje de giro de la plataforma en su correspondiente eje. La relación entre los dos ángulos será la siguiente.

$$\alpha = \frac{d}{L}\theta_x \tag{4.3.12}$$

Dónde:

$d \rightarrow$ Longitud del bazo del motor.

$L \rightarrow$ distancia del punto de aplicación del motor en el plato al punto de apoyo sobre el que gira.

$\theta_x \rightarrow$ Ángulo de giro del motor colocado en el eje x.

$$\beta = \frac{d}{L}\theta_y \tag{4.3.13}$$

Dónde:

$\theta_y \rightarrow$ Ángulo de giro del motor colocado en el eje y.

En conclusión, la función de transferencia del ángulo de inclinación del plato con respecto al ángulo de giro del motor será simplemente una ganancia tal y como se indica a continuación en la *Ecuación 4.3.14* y *Ecuación 4.3.15*.

$$G_\alpha = \frac{\alpha(s)}{\theta_x(s)} = \frac{d}{L} \tag{4.3.14}$$

$$G_\beta = \frac{\beta(s)}{\theta_y(s)} = \frac{d}{L} \tag{4.3.15}$$

4.4. DINÁMICA DE LA BOLA

Considerando que el valor aproximado del momento de inercia para una bola maciza es $I_b = \frac{2}{5}\, m_b r_b^2$, se va a reescribir la *Ecuación 3.21* y la *Ecuación 3.22* de la siguiente forma.

$$\left(m_b + \frac{2}{5}\,\frac{m_b r_b^2}{{r_b}^2}\right)\ddot{x_b} - m_b\left(x_b\,\dot{\alpha}^2 + y_b\,\dot{\alpha}\,\dot{\beta}\right) + m_b\,g\sin\alpha =$$
$$m_b\left(\frac{7}{5}\ddot{x_b} - x_b\,\dot{\alpha}^2 - y_b\,\dot{\alpha}\,\dot{\beta} + g\sin\alpha\right) = 0 \qquad (4.4.1)$$

$$\left(m_b + \frac{m_b r_b^2}{{r_b}^2}\right)\ddot{y_b} - m_b\left(x_b\,\dot{\alpha}\,\dot{\beta} + y_b\,\dot{\beta}^2\right) + m_b\,g\sin\beta =$$
$$m_b\left(\frac{7}{5}\ddot{y_b} - x_b\,\dot{\alpha}\,\dot{\beta} - y_b\,\dot{\beta}^2 + g\sin\beta\right) = 0 \qquad (4.4.2)$$

Para linealizar la *Ecuación 4.4.1* y la *Ecuación 4.4.2* habrá que tener en cuenta las consideraciones descritas en la *Ecuación 4.3.1* además de las simplificaciones que se describe a continuación.

- El Slow rate de cambio del plato tendrá un valor muy pequeño, entonces se podrá afirmar que:

$$\dot{\alpha} \ll 1,\ \dot{\beta} \ll 1 \rightarrow \dot{\alpha}\,\dot{\beta} \approx 0\,,\ \dot{\alpha}^2 \approx 0\,,\ \dot{\beta}^2 \approx 0 \qquad (4.4.3)$$

Aplicando estas consideraciones se podrá simplificar las ecuaciones del sistema hasta quedar de la siguiente forma.

$$\frac{7}{5}\,\ddot{x}_b + g\,\alpha = 0 \qquad (4.4.4)$$

$$\frac{7}{5}\,\ddot{y}_b + g\,\beta = 0 \qquad (4.4.5)$$

Quedaría entonces una ecuación diferencial lineal por cada eje X e Y, por lo que se podrá pasar a una expresión algebraica con condiciones iniciales nulas y así obtener la función de transferencia de la bola en cada eje, considerando el ángulo del plato α y β como entradas y la posición de la bola x_b e y_b como salidas.

$$G_{xb}(s) = \frac{X_b(s)}{\alpha(s)} = \frac{5}{7}\frac{g}{s^2} \qquad (4.4.8)$$

$$G_{yb}(s) = \frac{Y_b(s)}{\beta(s)} = \frac{5}{7}\frac{g}{s^2} \qquad (4.4.9)$$

4.5 DINÁMICA DE LA PLANTA

Una vez obtenida la función de transferencia de cada una de las partes del sistema planteadas en la *Figura 4.1*, el control en lazo abierto quedará de la siguiente forma.

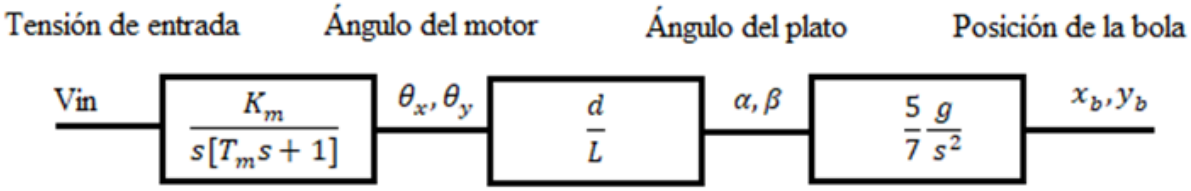


FIGURA 4.5.3. Dinámica de la planta del sistema.

Agrupando todas estas funciones de transferencia se podrá obtener la dinámica de la planta del sistema definida con una única función de transferencia en la que podremos representar la posición de la bola en cada eje en función de la tensión aplicada a la entrada del motor.

$$G_p = \frac{x_b(s), y_b(s)}{E(s)} = \frac{5}{7}\frac{K_m\, g\, d}{s^3[T_m s+1]L} \qquad (4.5.1)$$

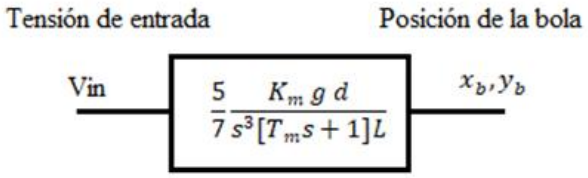


Figura 4.5.2.Dinámcia de la Planta del sistema simplificada.

5. DISEÑO DEL LAZO DE CONTROL

Para diseñar el control en lazo cerrado se va a realizar una realimentación del sistema en la que se obtendrá la diferencia de la posición de la bola con respecto a una posición de referencia. Este dato será la entrada del controlador PID del sistema que se debe diseñar acorde al sistema estudiado para enviar una señal de posición precisa al motor. El algoritmo del control PID consiste de tres parámetros distintos: el proporcional, el integral, y el derivativo, el valor Proporcional depende del error actual. El Integral depende de los errores pasados y el Derivativo es una predicción de los errores futuros. La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control [6].

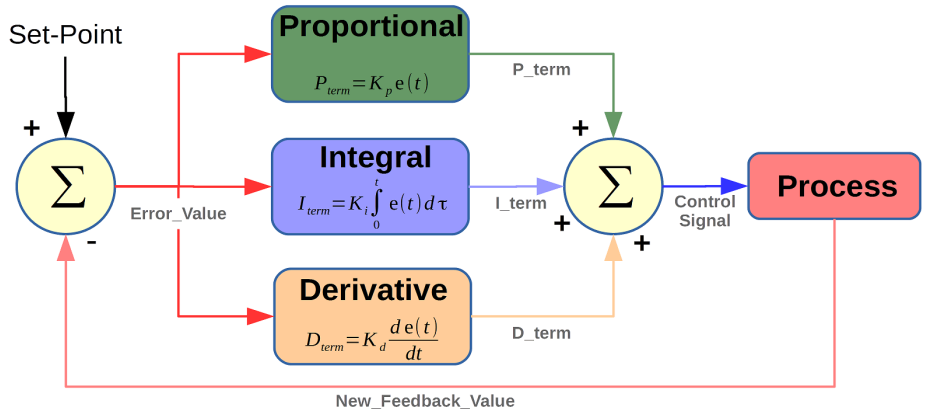


Figura 5.1. Controlador PID.

Ajustando estas tres variables en el algoritmo de control del PID, el controlador puede proveer una acción de control diseñado para los requerimientos del proceso en específico. La respuesta del controlador puede describirse en términos de la respuesta del control ante un error, el grado el cual el controlador sobrepasa el punto de ajuste, y el grado de oscilación del sistema. Algunas aplicaciones pueden solo requerir de uno o dos modos de los que provee este sistema de control. Un controlador PID puede ser llamado también PI, PD, P o I en la ausencia de las acciones de control respectivas.

Para el control en lazo cerrado se va a proceder a realizar un diseño de control en cascada que involucra sistemas de control de retroalimentación o circuitos que estén ordenados uno dentro del otro. Se caracteriza por dos controladores realimentados anidados, siendo la salida del primario (maestro) el punto de consigna del controlador secundario (esclavo). La salida del controlador secundario es la que actúa sobre el proceso. Con este tipo de control se puede lograr una reducción considerable de la variable obtenida a la salida del primer controlador a través del controlador secundario [7].

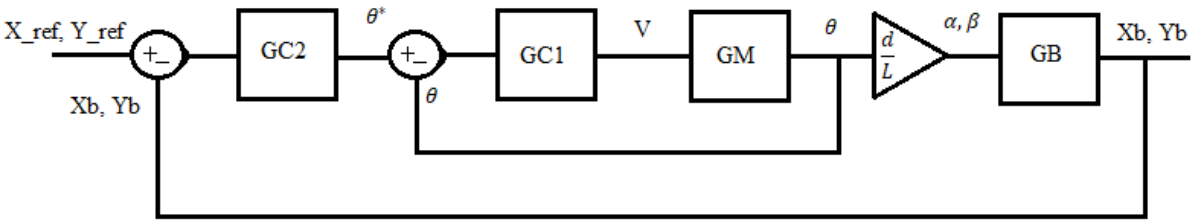


Figura 5.2. Control en cascada en Lazo cerrado.

Siendo GM y GP las funciones de transferencia de los motores y de la bola respectivamente y la ganancia $\frac{d}{L}$ la relación para obtener el ángulo del plato. GC1 y GC2 serán los controladores a diseñar. Los controladores utilizados serán PDs ya que la función de transferencia de la planta no tiene ningún cero por tanto es necesario añadir el integrador.

$$G_{C1}(s) = \frac{V(s)}{\theta^*(s) - \theta(s)} = Kp_1\, Td_1 \left(s + \frac{1}{Td_1}\right) \tag{5.1}$$

$$G_{C2}(s) = \frac{\theta^*(s)}{\mathrm{X}_{ref}, \mathrm{Y}_{\mathrm{ref}}(s) - x_{b,}y_b(s)} = Kp_2\, Td_2 \left(s + \frac{1}{Td_2}\right) \tag{5.2}$$

Dónde:

$V(s)$ → Tensión de entrada al motor.

$\theta^*(s)$ → Ángulo de referencia.

$\theta(s)$ → Ángulo de giro de los motores.

Kp_1 → Acción proporcional del controlador 1.

Td_1 → Acción derivativa del controlador 1.

Kp_2 → Acción proporcional del controlador 2.

Td_2 → Acción derivativa del controlador 2.

Para seleccionar los valores de sintonía de los controladores se deberá simplificar el lazo de la siguiente manera.

$$G_1(s) = G_{C1}(s)\, G_{MOTOR}(s) = Kp_1\, Td_1 \left(s + \frac{1}{Td_1}\right) \frac{K_m}{T_m\, s(s + 1/T_m)} \qquad (5.3)$$

Fijando el valor de $Td_1 = T_m$ se puede simplificar considerablemente la *Ecuación 5.3* como se muestra a continuación.

$$G_1(s) = \frac{Kp_1\, K_m}{s} \qquad (5.4)$$

Para resolver el primer lazo cerrado, se aplica la fórmula general y se obtiene el siguiente bloque.

$$G_2(s) = \frac{G_1(s)}{1 + G_1(s)} = \frac{Kp_1\, K_m}{s + Kp_1\, K_m} \qquad (5.5)$$

Por tanto ya se ha resuelto el primer lazo de control. Ahora habría que resolver el lazo principal que quedaría tal y como muestra la *Figura 5.3.*

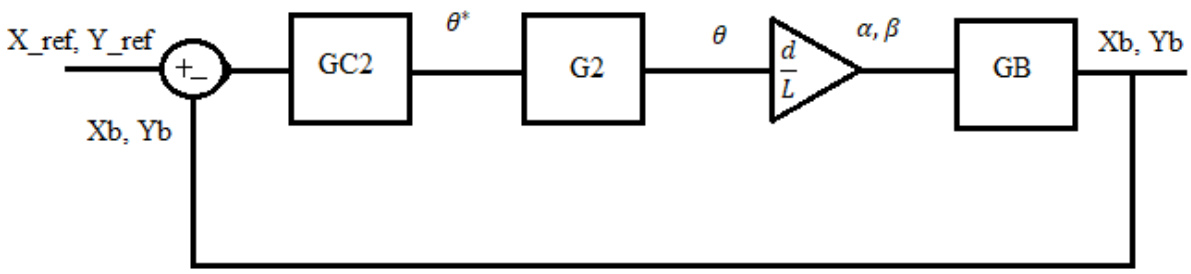


Figura 5.3. Lazo de control principal.

$$G_{BA}(s) = \frac{d}{L}\, G_{C2}\, G_2\, G_B = \frac{d}{L} Kp_2\, Td_2 \left(s + \frac{1}{Td_2}\right) \frac{Kp_1\, K_m}{s + Kp_1\, K_m} \frac{5\, g}{7\, s^2}$$

$$G_{BA}(s) = \left(\frac{5\, g\, d\, K_m}{7\, L}\right) Kp_1\, Kp_2\, Td_2\, \frac{(s + 1/Td_2)}{s^2(s + Kp_1\, K_m)} \qquad (5.6)$$

Esta función de transferencia contiene un polo doble en el origen, un polo simple en $-Kp_1\, K_m$ y un cero simple en $-1/Td_2$. En la siguiente ilustración se muestra la representación del lugar de las raíces.

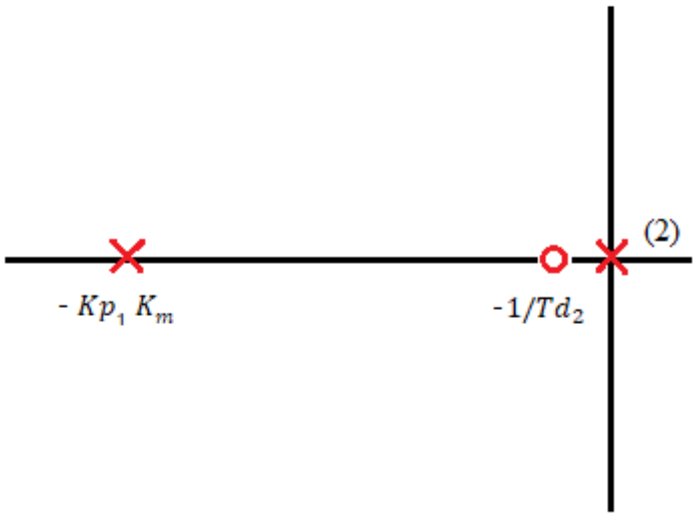


Figura 5.4. Representación del lugar de las raíces de la *Ecuación 5.6*.

Se puede apreciar que si se fija un valor muy grande para $Td_2 \approx 100$, el cero simple en $-1/Td_2$ estaría muy próximo al origen, por tanto podría anularse con uno de los polos del origen.

$$G_{BA}(s) = \left(\frac{5\, g\, d\, K_m}{7\, L}\right) Kp_1\, Kp_2\, Td_2\, \frac{1}{s(s + Kp_1\, K_m)} \qquad (5.7)$$

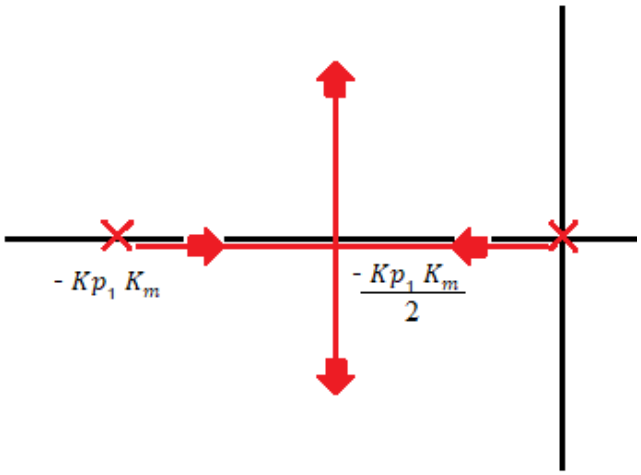


Figura 5.5. Representación del lugar de las raíces de la *Ecuación 5.7*.

Para definir Kp_1 se puede fijar por el tiempo de establecimiento.

$$ts_{5\%} = \frac{3}{\delta\omega_n} = 0{,}1\ ;\ \delta\omega_n = \frac{Kp_1\ K_m}{2};$$

$$Kp_1 = 2\ \frac{\delta\omega_n}{K_m} \qquad (5.8)$$

Ya solo quedaría fijar el valor de Kp_2 para tener definidos los controladores de nuestro sistema. Para ello se va a englobar todas las variables conocidas en una sola variable para simplificar el proceso.

$$A = \left(\frac{5\ g\ d\ K_m\ Kp_1\ Td_2}{7\ L}\right)$$

$$G_{BA}(s) = A\ Kp_2\ \frac{1}{s(s + Kp_1\ K_m)} \qquad (5.9)$$

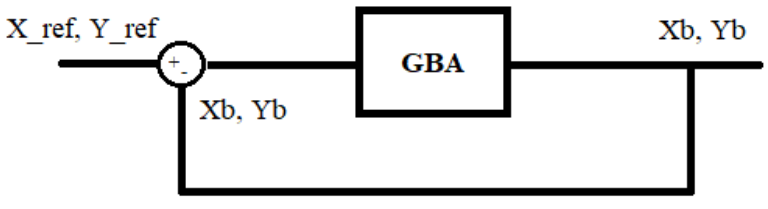


Figura 5.6. Lazo cerrado de control del sistema simplificado.

$$G_{BC}(s) = \frac{G_{BA}(s)}{1 + G_{BA}(s)} = \frac{A\ Kp_2}{s^2 + Kp_1\ K_m s + A\ Kp_2} \qquad (5.10)$$

El diseño del controlador adecuado para un sistema de segundo orden como es el caso de nuestro sistema, debe cumplir las siguientes condiciones.

$$G_{BC}(s) = \frac{A\ Kp_2}{s^2 + Kp_1\ K_m s + A\ Kp_2} \equiv \frac{K{\omega_n}^2}{s^2 + 2\ \delta\omega_n\ s + {\omega_n}^2} \qquad (5.11)$$

Para que la respuesta del sistema alcance el régimen permanente lo antes posible sin sobreoscilaciones, el valor del coeficiente de amortiguamiento tiene que ser 1 ($\delta = 1$), es decir, el sistema tiene que ser críticamente amortiguado. Igualando los términos de la *Ecuación 5.11* se obtendrá el valor de la variable Kp_2.

$$Kp_1\ K_m = 2\ \omega_n\ ;\ \omega_n = \frac{Kp_1\ K_m}{2} \qquad (5.12)$$

$$A\ Kp_2 = {\omega_n}^2;$$

$$Kp_2 = \frac{{\omega_n}^2}{A} = \frac{(Kp_1\ K_m)^2}{4\ A} = \frac{7\ Kp_1\ K_m\ L}{20\ g\ d\ Td_2} \qquad (5.13)$$

Por tanto, ya se han definido los controladores de nuestro sistema. Con los datos obtenidos se podrán realizar las simulaciones correspondientes mediante la herramienta SIMULINK y así observar el comportamiento del sistema.

6. SIMULACIONES DEL LAZO DE CONTROL DISEÑADO

En este apartado se va a poner en práctica los resultados obtenidos teóricamente mediante la herramienta de simulación SIMULINK para así comprender mejor el funcionamiento del proyecto y demostrar que las ecuaciones desarrolladas en el modelado matemático realmente son acordes al sistema estudiado en este proyecto.

Para ello se realizará una hoja con los parámetros de las variables reales de los componentes usados en el proyecto que puedan afectar al comportamiento del sistema con el fin de obtener resultados acordes con la realidad y poder aplicarlos para programar el controlador del motor del sistema real.

6.1. VARIABLES REALES DEL SISTEMA

Las variables que habrá que definir para realizar las simulaciones serán todas las características de los servomotores utilizados, la dimensión y peso de la bola empleada y las distancias significativas.

Los servomotores utilizados en este caso son HI-TEC HS-5485HB cuyos parámetros son las siguientes.

Parámetros	Símbolo	Valor	Unidad de Medida
Resistencia	R	0,4	Ω
Inductancia	L	100	uH
Constante Electromotriz	k_b	10	$V/(rad/s)$
Constante de torque	k_p	10	Nm/A
Constante de inercia	j	0,5	$kg\ m^2$
Constante de fricción	B	0,2	Nms/rad
Constante de Coulomb	k_c	0,5	Nms

Tabla 6.1.1. Variables de los motores utilizados.

La bola utilizada será una bola extraída de un ratón antiguo. Las variables a controlar de dicha bola serán las siguientes.

Parámetros	Símbolo	Valor	Unidad de Medida
Diámetro de la bola	D_b	0,015	m
masa de la bola	m_b	0,11	kg

Tabla 6.1.2. Distancias significativas.

Otros parámetros importantes que habrá que definir previamente serán las distancias significativas del sistema. Para este caso tendremos las indicadas a continuación.

Parámetros	Símbolo	Valor	Unidad de Medida
Longitud de brazo de motor	d	0,015	m
Distancia del punto de aplicación del motor en el plato al punto de apoyo sobre el que gira.	L	0,07	m

Tabla 6.1.3.

Antes de realizar las simulaciones se deberá crear un archivo en MATLAB que englobe todos estos parámetros y variables que se han obtenido de los componentes utilizados. Con esto se consigue que el algoritmo de control realizado no solo funcione para este proyecto con estos materiales específicos, sino que si se decide cambiar alguno de los materiales utilizados por otro con otras características diferentes o modificar alguna distancia significativa para realizar cualquier tipo de prueba o mejora futura, sólo habrá que modificar el archivo de valores con las nuevas variables y el algoritmo de control de SIMULINK trabajará con estas nuevas variables sin necesidad de modificarlo.

Variables.m

```
%Caracteristicas del motor:
Kp=10; %(N*m)/A
Kb=10; %V/(rad/s)
J=0.5; %(Kg*m^2)
B=0.2;%(N*m)/(rad/s)
R=0.4; %ohm
L=100*10^-6;%(H)
n=32;

% Caracteristicas de la bola
mb=0.11; %Kg
rb=0.015; %m
g=9.81; %m/s^2
Ib=(2/5)*mb*rb^2;

%Distancia del centro del plato al extremo:
l=0.07; %m

%Distancia del centro del eje del motor al extremo:
d=0.015; %m

%Parámetros de Motor:

Km=Kp/((R*B)+(Kb*Kp));

Tm=(R*J)/((R*B)+(Kb*Kp));
```

Figura 6.1.4. Archivo de MATLAB de variables reales predefinidas.

En este mismo archivo se incluirá a continuación los parámetros y variables necesarios para calcular la acción proporcional y derivativa de cada uno de los controladores diseñados anteriormente y que serán necesarios para realizar las simulaciones en lazo cerrado.

```
27
28      %Especificaciones PID
29 -    ts=0.1;
30 -    dwn=3/ts;
31
32      %Controlador 1:
33 -    kp1=(dwn*2)/Km;
34 -    Td1=Tm;
35
36      %Controlador 2:
37 -    Td2=100;
38 -    kp2=(7*kp1*Km*1)/(20*d*g*Td2);
```

Figura 6.1.5. Cálculo de acción proporcional y derivativa de los controladores diseñados.

6.2. SIMULACIONES DEL COMPORTAMIENTO DE LA PLANTA

En este apartado se realizarán las simulaciones de los diagramas de bloques obtenidos mediante linealización del modelado matemático. Se estudiará la evolución de la posición de la bola con respecto una entrada escalón.

Para justificar las simplificaciones en las ecuaciones correspondientes a la bola propuestas en el apartado 4.4, se realizará una simulación despejando la variable de la posición de la bola (x_b, y_b) de la ecuación completa *(Ecuación 3.21 – 3.22)* obtenida en el modelo matemático y otra con el bloque de la función de transferencia obtenida simplificando dicha ecuación.

- Comportamiento de posición la bola en respuesta a una señal escalón despejando x_b en la *(Ecuación 3.21)* y considerando un solo eje.

$$\ddot{x_b} = \frac{m_b[\,(x_b\,\dot{\alpha}^2) - \,g\sin\alpha]}{\left(m_b + \frac{I_b}{r_b{}^2}\right)} \qquad (6.2.1)$$

Implementamos dicha ecuación mediante bloques en SIMULINK.

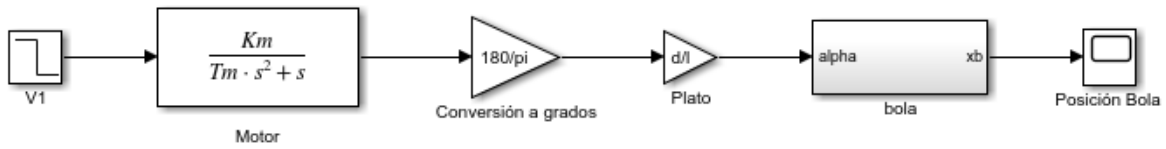


Figura 6.2.1. Diagrama de bloques del sistema en lazo abierto.

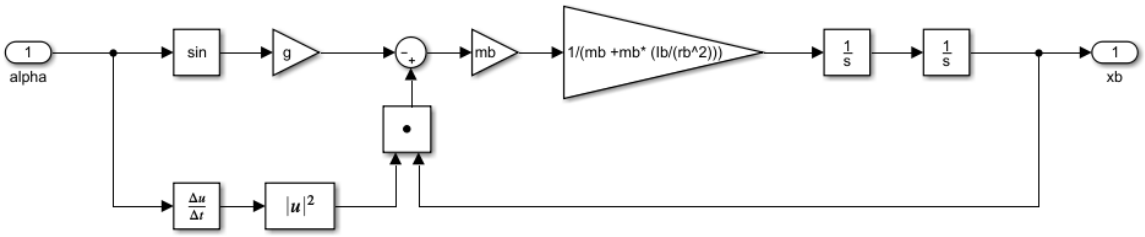


Figura 6.2.2. Diagrama de bloques del subsistema bola implementando la *(Ecuación 6.2.1)*.

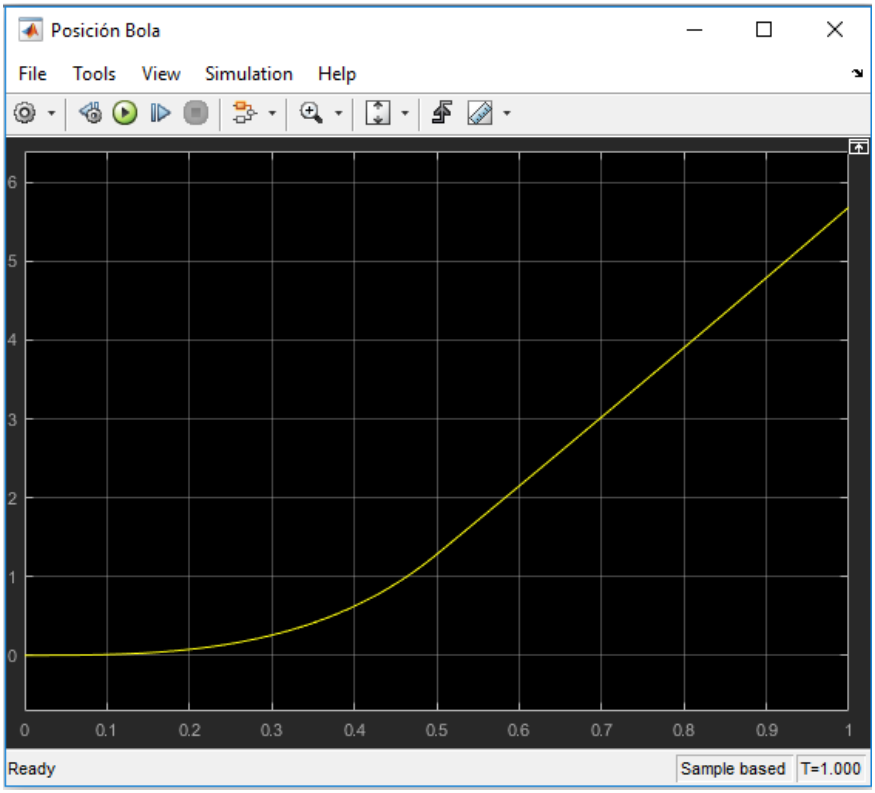


Figura 6.2.3. Evolución de la posición de la bola ante una señal escalón.

- Comportamiento de posición la bola en respuesta a una señal escalón utilizado el bloque de función de transferencia hallado considerando las simplificaciones planteadas anteriormente en un solo eje.

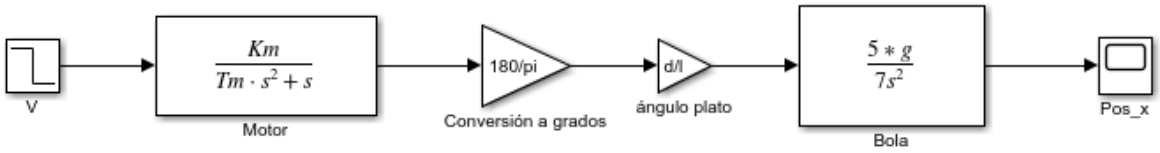


Figura 6.2.4. Diagrama de bloques del sistema con la función de transferencia de la bola.

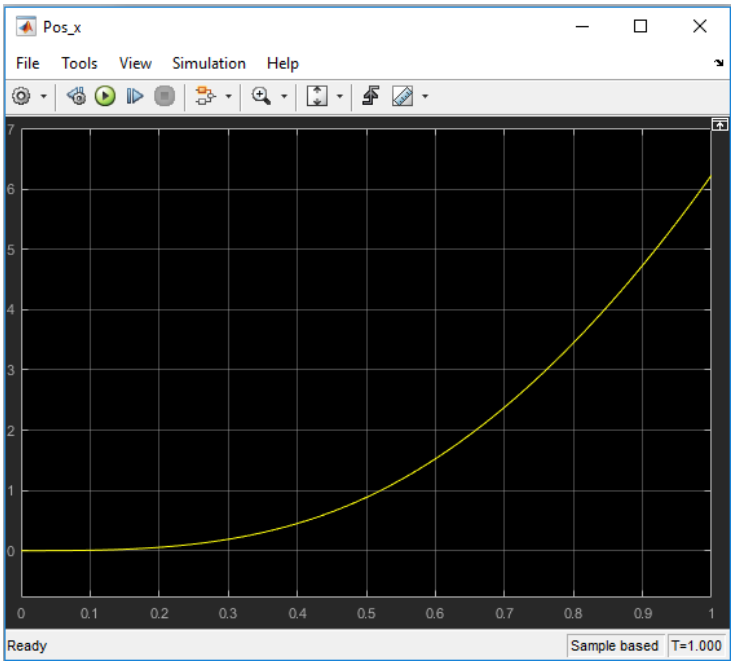


Figura 6.2.5. Evolución de la posición de la bola ante una señal escalón.

Se puede observar que la evolución de la posición de la bola en los dos casos es muy similar.

- Comportamiento de posición la bola en respuesta a una señal escalón utilizado el bloque de función de transferencia de la planta en un eje.

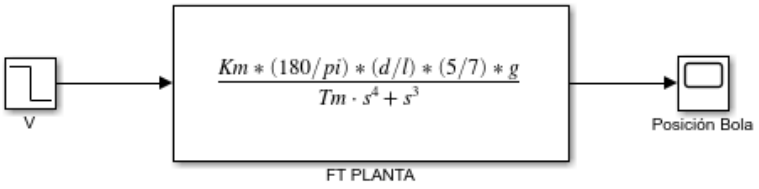


Figura 6.2.6. Función de transferencia de la planta del sistema.

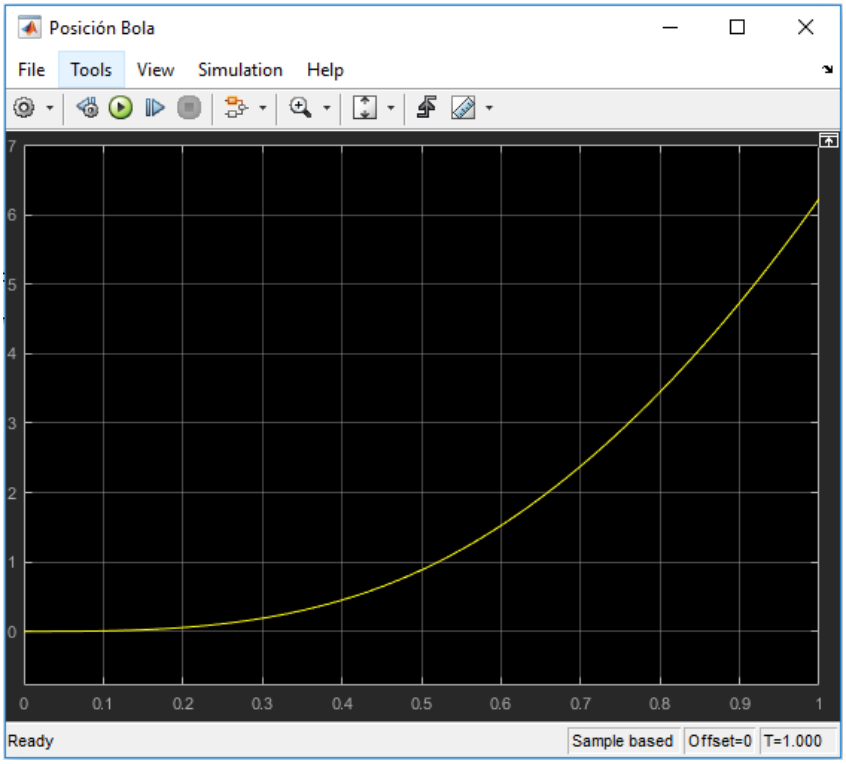


Figura 6.2.7. Evolución de la posición de la bola simulando el diagram de bolques de la Figura 6.2.6.

- Comportamiento de posición la bola en respuesta a una señal escalón despejando x_b, y_b en la *(Ecuación 3.21-3.22)* considerando el sistema que se va a implementar controlando los dos ejes.

$$\ddot{x_b} = \frac{m_b[\left(x_b\,\dot{\alpha}^2 + y_b\,\dot{\alpha}\,\dot{\beta}\right) - \,g\sin\alpha]}{\left(m_b + \frac{I_b}{{r_b}^2}\right)} \qquad (6.2.2)$$

$$\ddot{y_b} = \frac{m_b[\left(x_b\,\dot{\alpha}\,\dot{\beta} + y_b\,\dot{\beta}^2\right) - \,g\sin\beta]}{\left(m_b + \frac{I_b}{{r_b}^2}\right)} \qquad (6.2.3)$$

Representación del diagrama de bloques correspondiente al sistema Ball and Plate controlando tanto el eje X como el eje Y.

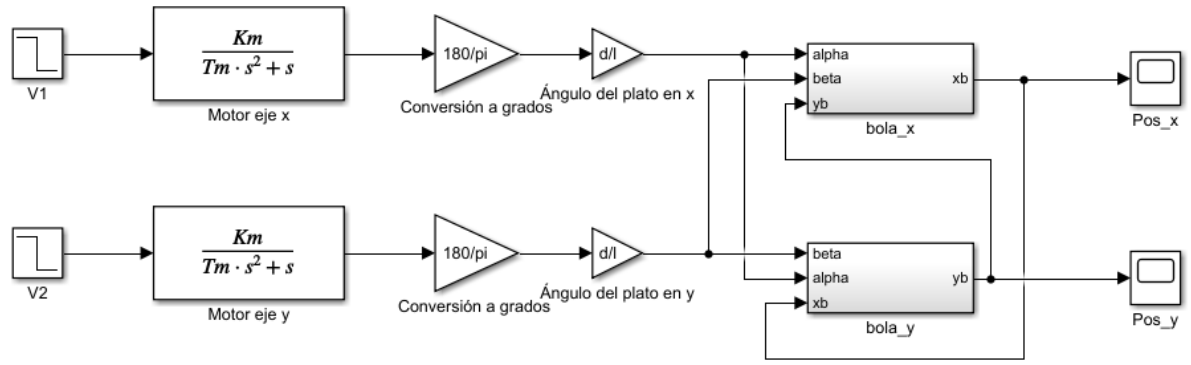


Figura 6.2.8. Planta del sistema Ball and Plate en lazo abierto.

Los subsistemas bola_x y bola_y contienen el diagrama de bloques correspondiente a cada ecuación de la bola según el eje correspondiente. A continuación se mostrará el diagrama de bloques que define a cada subsistema.

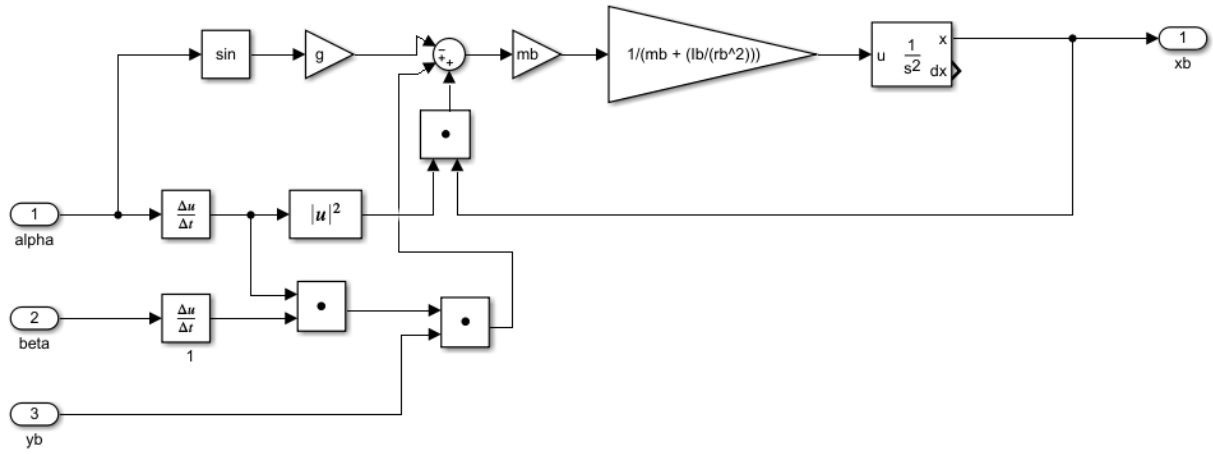


Figura 6.2.9. Representación de la *Ecuación 6.6.2* que define la posición de la bola en el eje x.

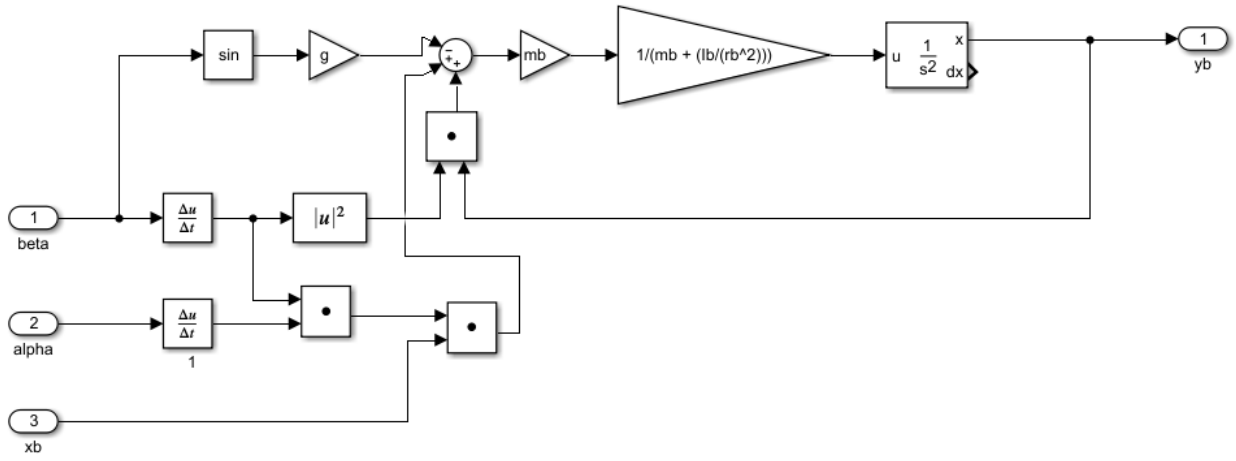


Figura 6.2.10. Representación de la *Ecuación 6.2.3* que define la posición de la bola en el eje y.

Al simular este sistema aplicando una entrada escalón unitario a la entrada en caca uno de los ejes, se obtienen los siguientes resultados.

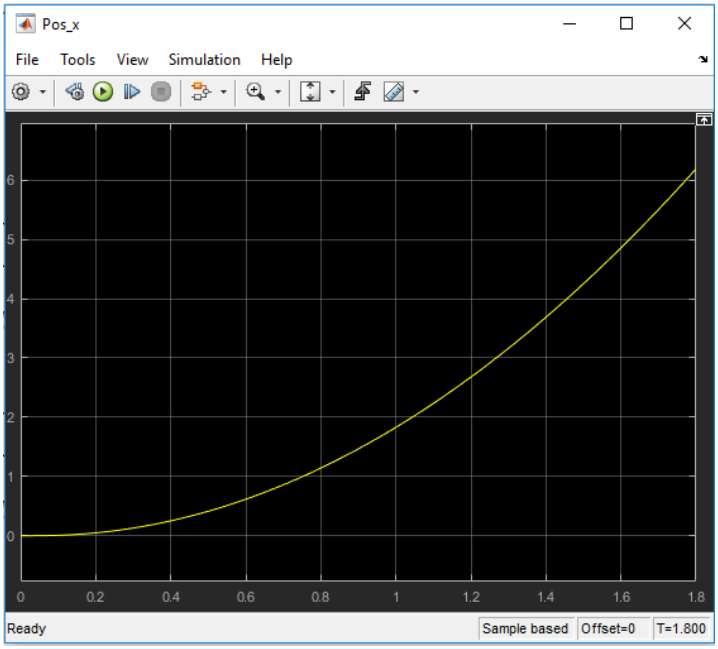


Figura 6.2.11. Evolución de la posición de la bola ante una señal escalón en el eje x.

El resultado será idéntico en cada uno de los dos ejes ya que, como se ha comentado anteriormente, los dos motores son iguales y están colocados a la misma distancia con respecto al eje de giro del plato.

- Comportamiento de la posición ante una entrada de escalón unitario utilizando el bloque de la función de transferencia hallada en el apartado de la dinámica de la bola para cada eje.

Teniendo en cuenta las simplificaciones empleadas para linealizar las ecuaciones y obtener así las funciones de transferencia, se llega a la conclusión de que las variables de uno de los ejes no van a influir en el otro eje y por tanto, se puede hacer el control de cada eje por separado sin que dependa el uno del otro. A continuación se realizará la simulación para este caso y los resultados obtenidos se compararán con los obtenidos en el caso anterior en el que sí están relacionados los dos ejes.

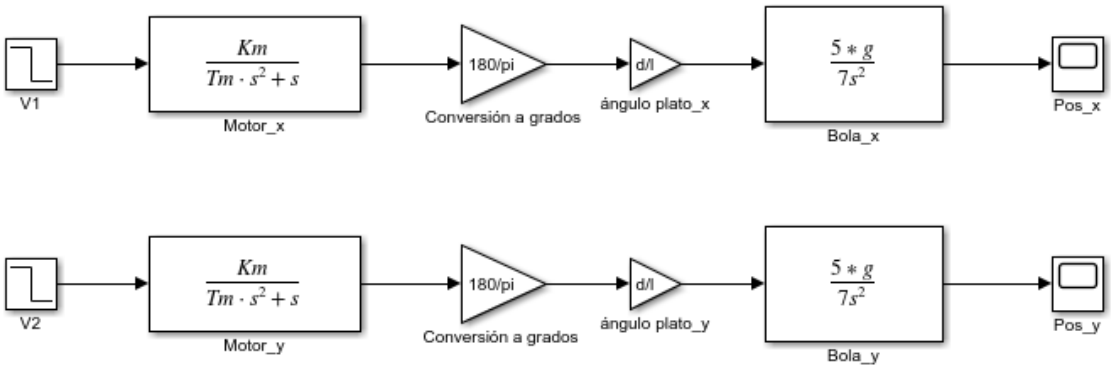


Figura 6.2.12. Planta del sistema Ball and Plate en lazo abierto con ejes independientes.

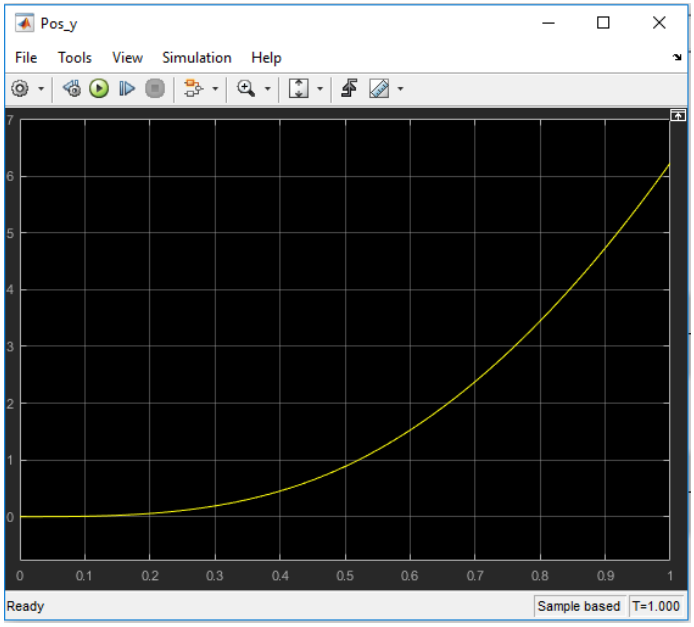


Figura 6.2.13. Evolución de la posición de la bola ante una señal escalón en el eje y.

En este caso se muestra el resultado obtenido en el eje Y para así ir alternando simulaciones las simulaciones de cada eje. De esta forma se puede mostrar que los resultados serán iguales en los dos ejes evitando repetir la imagen en cada simulación realizada.

Se puede observar que en los dos casos planteados los resultados son similares, **se confirma entonces que las simplificaciones consideradas para linealizar las ecuaciones del modelo matemático son correctas y por tanto, que se puede hacer el control de cada eje por separado sin que dependa el uno del otro.**

- Comportamiento de posición la bola en respuesta a una señal escalón utilizado el bloque de función de transferencia de la planta del sistema para cada eje.

En este caso la función de transferencia de cada eje se va a obtener a partir del modelado que usa el diagrama de bloques para definir las ecuaciones de la dinámca de la bola para así demostrar que el resultado es el mismo. Como se puede observar, en este caso no podemos obtener la función de transferencia de la planta completa directamente como se hizo en el caso estudiado anteriormente ya que en este caso la dinámica de la bola no será lineal. Se ralizará por tanto mediante un análisis lineal con la herramienta MATLAB-SIMULINK.

Primero habrá que introducir todo proceso de la planta del sistema mostrado en *Figura 6.2.8* en un solo bloque subsistema que defina toda la planta del sistema como se muestra a continuación.

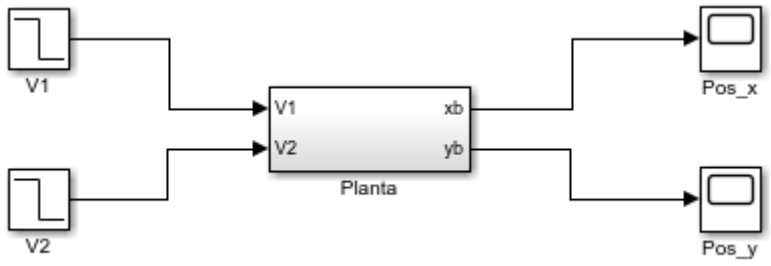


Figura 6.2.14. Planta del sistema Ball and Plate englobada en un subsistema.

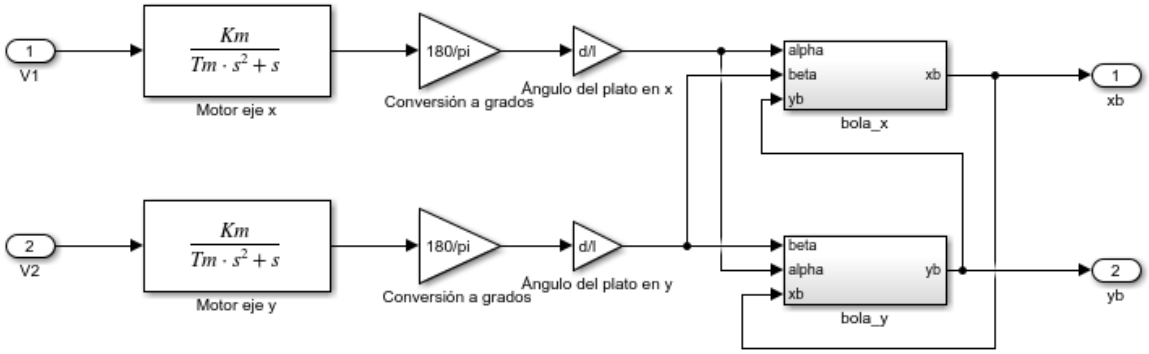


Figura 6.2.15. Subsitema Planta.

Para linealizar el sistema y obtener la función de transferencia de la planta habrá que realizar las siguientes acciones. Se realizará primero para un eje y después para el otro. Definimos la entrada y salida del eje x como open-loop de la siguiente manera.

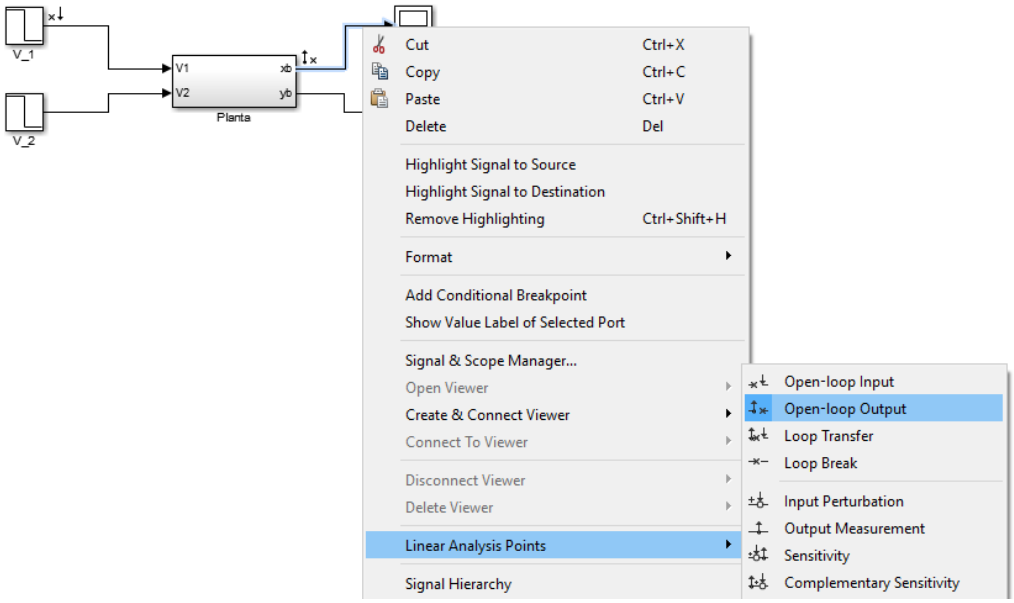


Figura 6.2.16. Definir entrada y salida como open-loop.

A continuación, se realizará el análisis linear pinchando en el botón “analysis” de la barra de herramientas de Simulink.

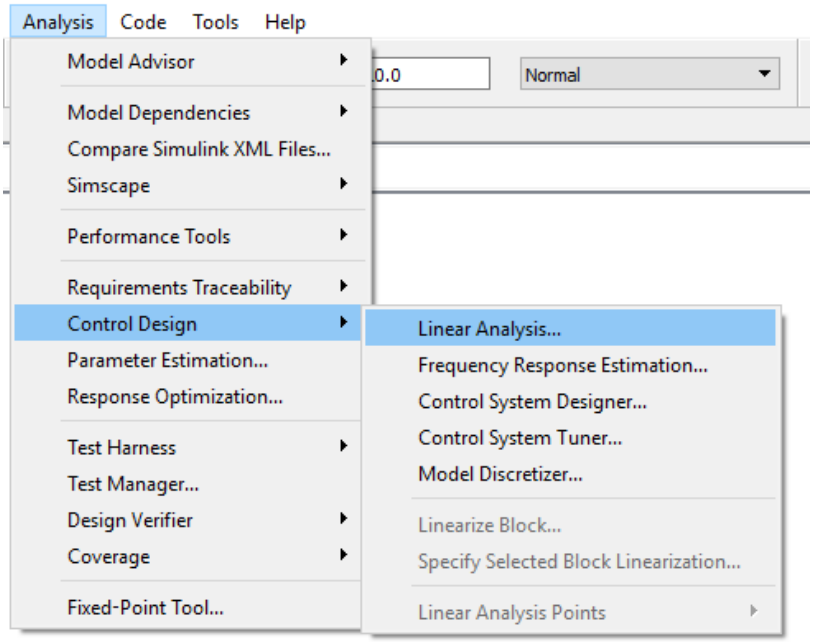


Figura 6.2.17. Análisis lineal de diseño de control.

Se abrirá una ventana nueva como la que se muestra en la figura y para que se realice el análisis lineal habrá que pinchar en el botón “STEP”.

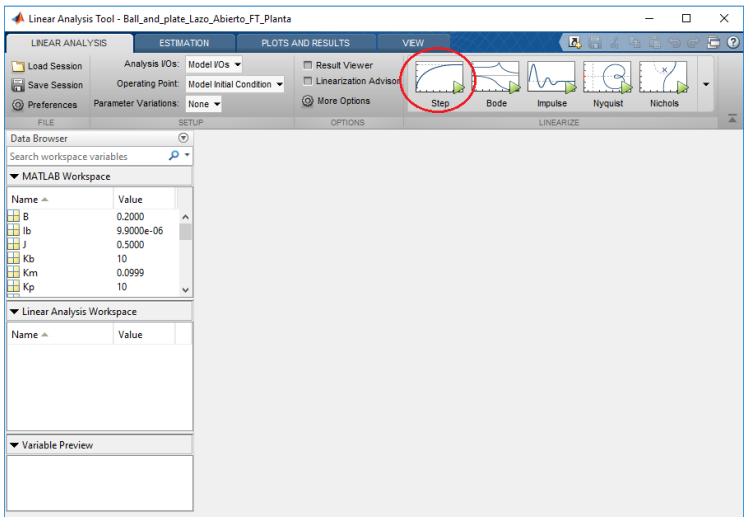


Figura 6.2.18. Ventana generada para el análisis lineal.

Aparecerá ahora una gráfica como las obtenidas anteriormente en la que se muestra la respuesta de la posición de la bola ante una entrada de señal escalón.

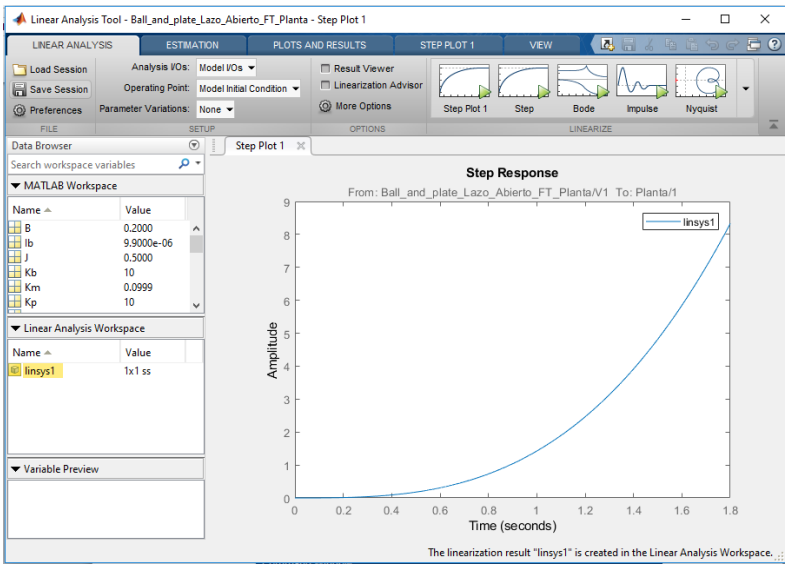


Figura 6.2.19. Evolución de la posición de la bola ante una señal escalón en el eje x.

Para obtener la función de transferencia correspondiente a esta gráfica, solo habrá que copiar el resultado obtenido en esta ventana (linsys1) y pegarlo en el workspace de MATLAB, es decir, en la ventana que aparece justo encima.

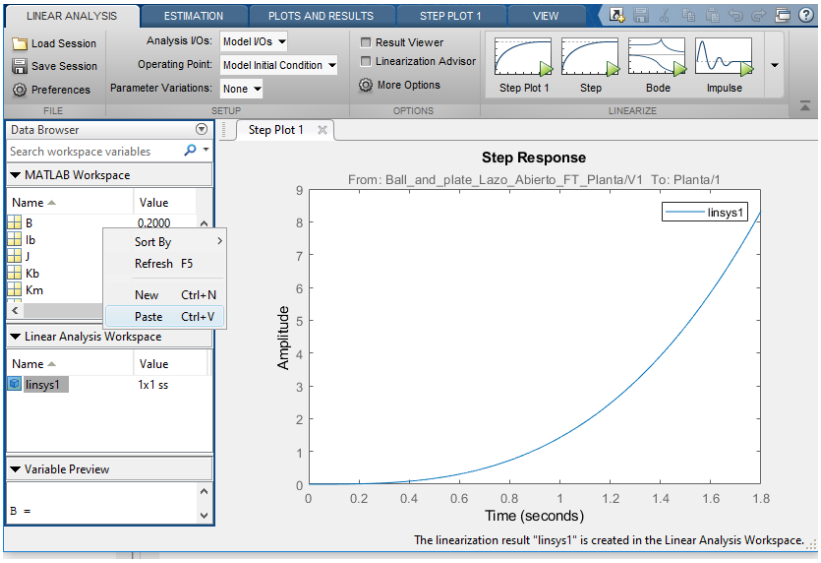


Figura 6.2.20. Pegar linsys1 en workspace de MATLAB.

Una vez que aparezca la variable sinsys1 en el workspace, se podrá obtener la función de transferencia de la planta en el eje x escribiendo en la ventana de comandos de MATLAB la siguiente instrucción.

```
>> PLANTA_X=tf(linsys1)

PLANTA_X =

  From input "Ball_and_plate_Lazo_Abierto_FT_Planta/V1" to output "Planta/1":
       4302
  ---------------
  s^4 + 500.4 s^3

Name: Linearization at model initial condition
Continuous-time transfer function.
```

Figura 6.2.21.Función de transferencia de la planta obtenida mediante Análisis lineal en MATLAB.

Ahora se procederá a comprobar que la ecuación obtenida define a la planta de nuestro sistema. Para ello se sustituye el bloque subsistema que engloba a la planta del sistema y se sustituye por la función de transferencia obtenida y se realiza la simulación.

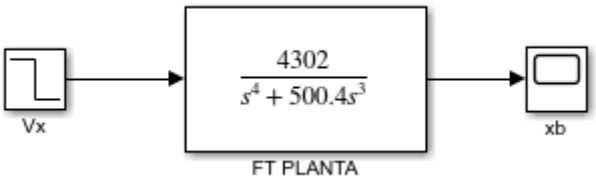


Figura 6.2.22. Función de transferencia de la planta del sistema.

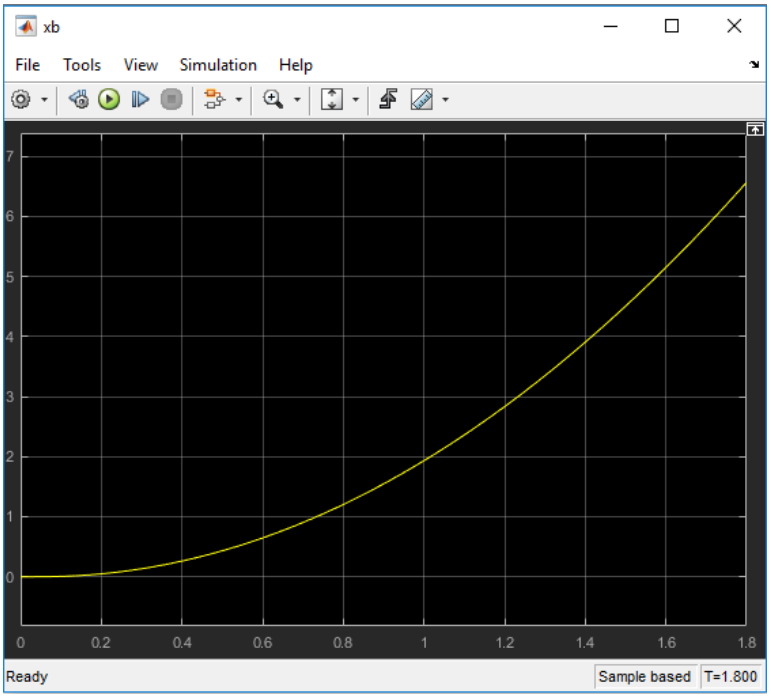


Figura 6.2.23. Gráfica de la posición de la bola obtenida de la function de transferencia de la planta eje x.

Para el eje Y habrá que realizar el mismo procedimiento definiendo la entrada y salida correspondientes a este eje como open-loop. Se obtiene el mismo resultado en el eje X que en el eje Y, que como se comentó anteriormente este resultado es lógico debido a que se utilizan los mismos parámetros y las mismas variables en los dos ejes. Con este proceso quedaría definida la planta del sistema en lazo abierto.

Se puede observar que la gráfica obtenida es exactamente igual que la obtenida utilizando los bloques de las dinámicas de cada una de las partes del sistema y que los resultados de las simulaciones obtenidos en la *Figura 6.2.23* y en la *Figura 6.2.7* son iguales**. Con esta comparativa se ha querido demostrar que se puede obtener la función de transferencia de la planta completa del sistema de dos maneras diferentes**, tanto para el caso en el que su tiene un sistema lineal como para el caso en el que el sistema es no lineal, obteniendo el mismo resultado en los dos casos.

6.3. SIMULACIONES DEL SISTEMA REALIMENTADO

En este apartado se pondrá en práctica los resultados obtenidos en el apartado *5. Control en lazo cerrado* donde se diseñó los controladores del sistema. Esto nos permitirá representar gráficamente los parámetros controlados y nos ayudará a entender mejor el funcionamiento del sistema. Una vez introducidos los parámetros en el archivo de MATLAB como se indicó en el apartado de variables significativas, se va a proceder a las simulaciones mediante algoritmos de bloques con la herramienta SIMULINK.

Como ya se ha demostrado, se puede controlar el sistema Ball and Plate controlando cada eje por separado. Por tanto se aplicará el control en cascada diseñado a cada uno de los ejes por separado y se representará gráficamente como se estabiliza la posición de bola en cada eje en la posición de referencia que se tiene a la entrada.

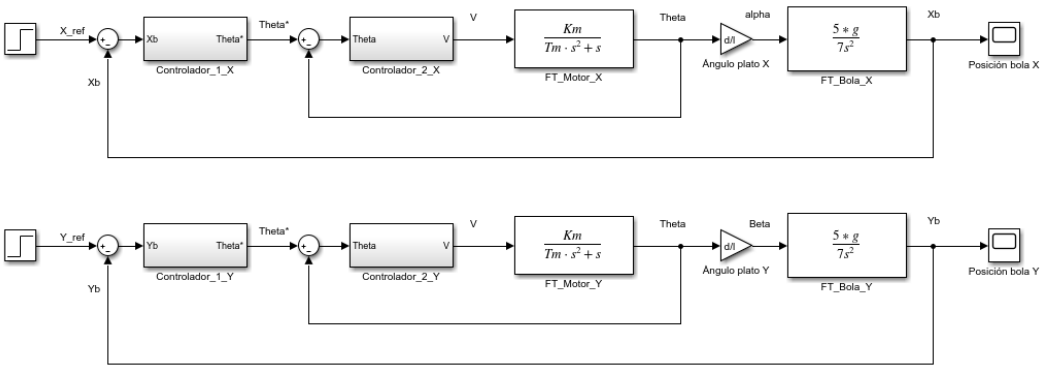


Figura 6.3.1. Algoritmo de control en cascada en SIMULINK.

Dentro de los bloques subsistema del controlador 1 y controlador 2 habrá el siguiente algoritmo de bloques. Se usarán los mismos controladores tanto para el eje x como para el eje y ya que partimos de las mismas condiciones.

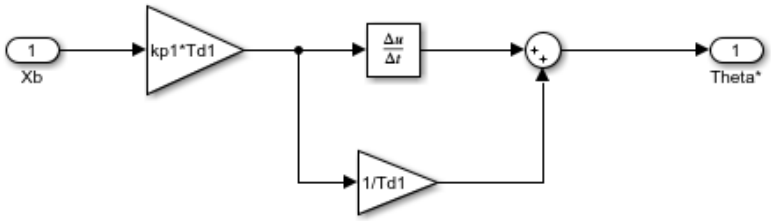


Figura 6.3.2. Bloque controlador 1.

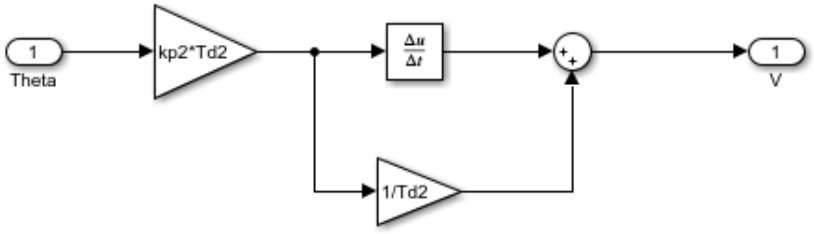


Figura 6.3.3. Bloque controlador 2.

Para la simulación de este algoritmo se aplicará como referencia a la entrada una señal escalón de valor unitario en cada eje. El objetivo será mantener estable la posición de la bola de cada eje estable en dicho valor de referencia con el mínimo amortiguamiento posible.

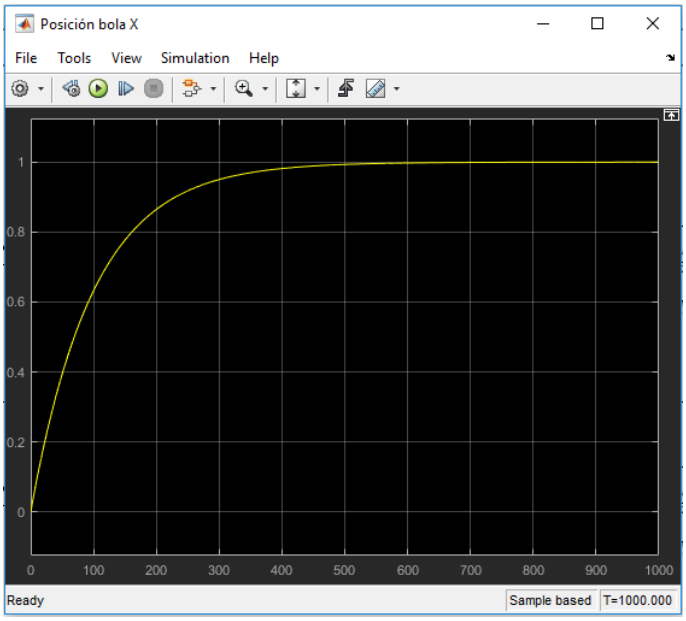


Figura 6.3.4. Simulación lazo cerrado en cascada eje x.

El resultado para el eje Y sería el mismo que el obtenido en el eje X. Como se puede observar en las gráficas obtenidas en las simulaciones, en cada eje se cumple la condición de que la bola se quede estable en la posición de referencia impuesta inicialmente.

Otra forma de diseñar un controlador adecuado para el sistema Ball and Plate es utilizar la función de transferencia de la planta del sistema obtenida en lazo abierto. Habría que calcular el lugar de las raíces de dicha función de transferencia para obtener los polos y ceros que contiene. Para ello se va a utilizar la ventana de comandos de MATLAB en la que se introducirá la función tf() para crear la función de transferencia y la función rlocus() para mostrar el lugar de las raíces de dicha función de transferencia.

```
>> Gp=tf([4302],[1 500.4 0 0 0])

Gp =

        4302
  ---------------
  s^4 + 500.4 s^3

Continuous-time transfer function.
```

Función 6.3.5. Función de transferencia de la planta del sistema en lazo abierto.

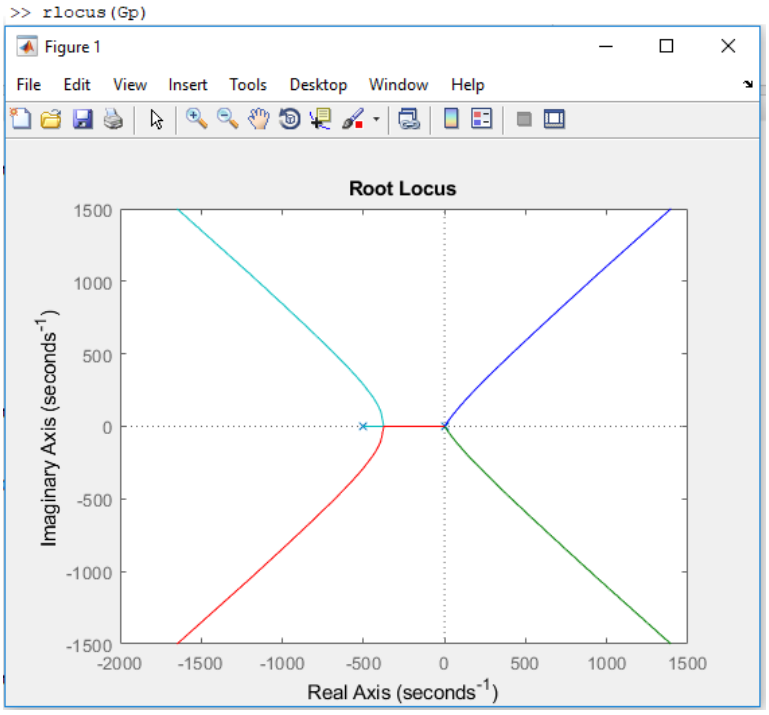


figura 6.3.6. Lugar de las raíces de la Función de transferencia de la planta del sistema en lazo abierto.

Como se puede apreciar en el lugar de las raíces el sistema tiene tres polos en el origen y un polo en -500.4. Con esto se llega a la conclusión que necesitamos un controlador PD, ya que un PID nos añadiría otro polo más. Para el diseño de este controlador se van a añadir dos ceros en el origen en la función de transferencia de la planta para así anular dos de los tres polos que hay situados en el origen. Añadiendo una acción derivativa que introduzca otro cero entre el origen y el punto -500.4 se consigue que el sistema sea estable. Para el obtener el valor de la acción derivativa y proporcional se irán asignando valores hasta que se obtenga la respuesta deseada en el sistema. El algoritmo de bloques de control en este caso quedaría de la siguiente manera.

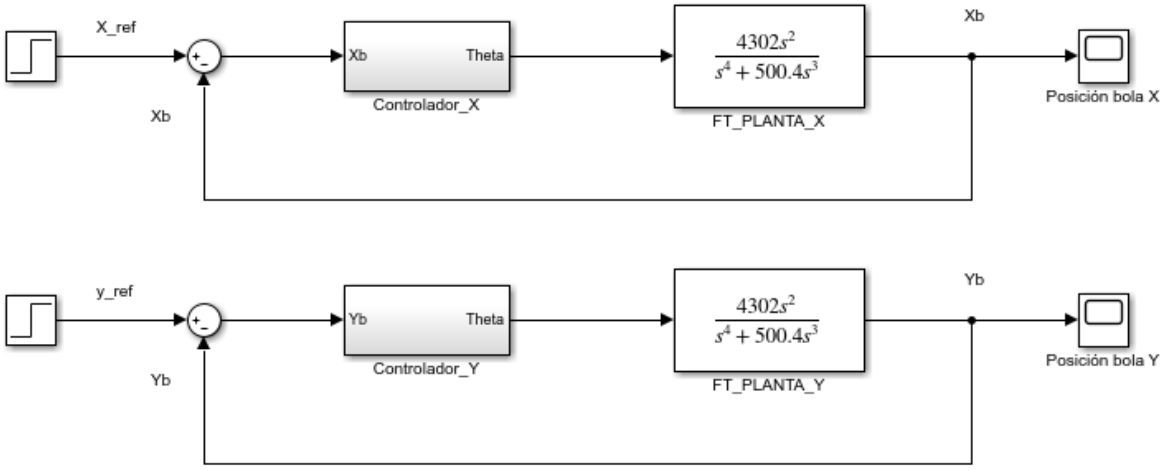


Figura 6.3.7.Diseño controlador Lazo cerrado.

Dentro del subsistema del controlador se encuentra el diagrama de bloques correspondiente a un controlador PD con los valores de la acción proporcional y derivativa que se han considerado adecuados para este sistema según las simulaciones realizadas.

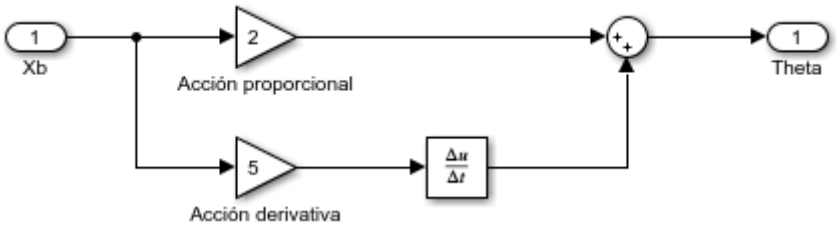


Figura 6.3.8. Controlador del eje X.

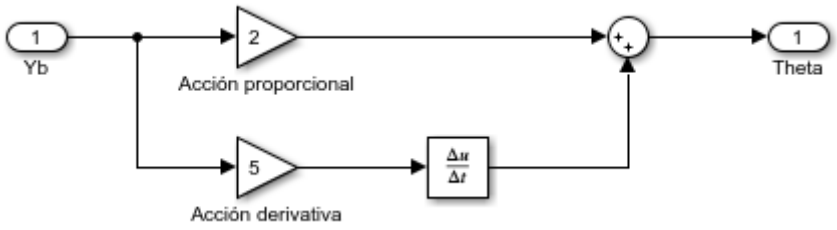


Figura 6.3.9. Controlador del eje Y.

El objetivo de este algoritmo es similar al diseñado con control en cascada, se debe estabilizar la posición de la bola en la posición de referencia establecida a la entrada. En este caso se ha aplicado también una señal escalón de valor unitario a la entrada para marcar la referencia. Al simular mediante la herramienta SIMULINK obtenemos los siguientes resultados.

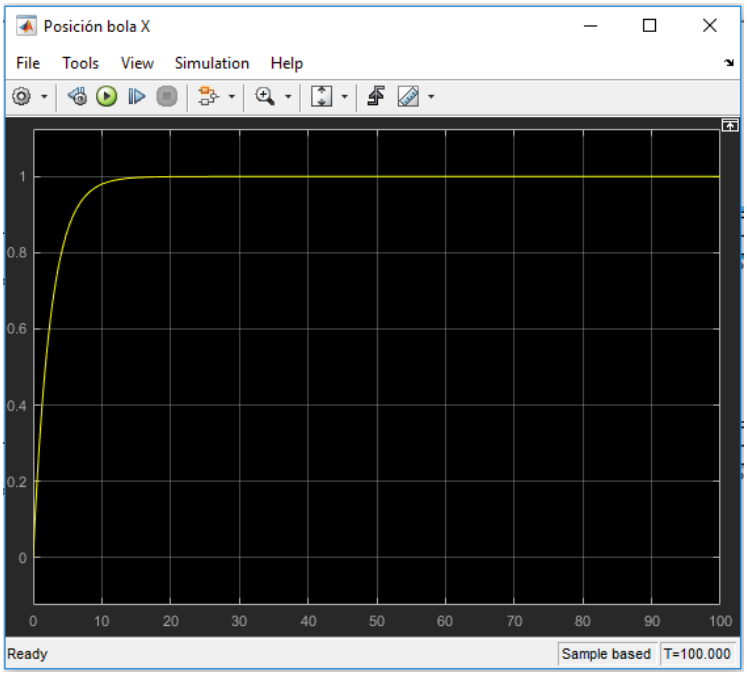


Figura 6.3.10. Posición estabilizada de la bola en el eje X.

En el eje Y se obtiene el mismo resultado de simulación que se ha obtenido en el eje X.

En conclusión, comparando los resultados obtenidos de los dos métodos en lazo cerrado, se observa que con este método se consigue estabilizar la bola más rápido y tiene la ventaja de poder ajustarlo a medida del sistema según nos interese. Sin embargo, el método de control en cascada es más preciso y real ya que el conctrolador se diseña acorde a los parámetros y variables del sistema real en cuestión.

7. CONTROL DE LA MAQUETA EN TIEMPO REAL: SIMULINK-ARDUINO

Una vez realizado el montaje mecánico del Hardware propuesto mostrado en las *Figuras 2.12-2.13* y se va a proceder a realizar un control real del sistema. Para ello surge la idea de realizar el control mediante la herramienta SIMULINK utilizando Arduino como pasarela, es decir, MATLAB proporciona una plataforma para diseñar un proyecto en Simulink y cargarlo en Arduino con la posibilidad de visualizar resultados obtenidos en tiempo real.

El soporte de Arduino en MATLAB incluye una biblioteca de bloques específicos que se encargan de gestionar las entradas/salidas. Internamente están programados en C y hacen la función de drivers del hardware externo. El resto de las funciones se implementan mediante los bloques estándar de Simulink. El conjunto de bloques proporciona acceso a: entradas/salidas digitales, entradas analógicas, salidas PWM, comunicaciones serie y entradas/salidas de pulsos para servos RC. Para ello se debe instalar el paquete ArduinoIO de MathWorks para manejar la interfaz entre Simulink y el hardware. Se usa este paquete porque permite una comunicación más transparente y en tiempo real entre Simulink y Arduino sin profundizar en las complejidades de la comunicación en puerto serie. El paquete ArduinoIO que se empleará consiste principalmente en un simple programa de bloques especial para Simulink que se ejecuta en la placa Arduino. El programa actúa como un servidor para pasar información entre el hardware y el ordenador que ejecuta un modelo Simulink. Estos elementos nos permiten acceder a entradas y salidas digitales de Arduino y entradas analógicas, y codificadores de lectura, todo desde Simulink (o la línea de comandos de MATLAB).

7.1. INSTALACIÓN ARDUINO SUPPORT PACKAGES IN MATLAB

Para la instalación de este paquete se necesita una versión de MATLAB 2013 o superior. Una vez se disponga de esta herramienta se procede a abrir el programa y dirigirse a la barra de menú de la pantalla de inicio del programa y seleccionar HOME → Add Ons → Get Hardware Support Packages, como se muestra en la *Figura 7.1.1*. [8].

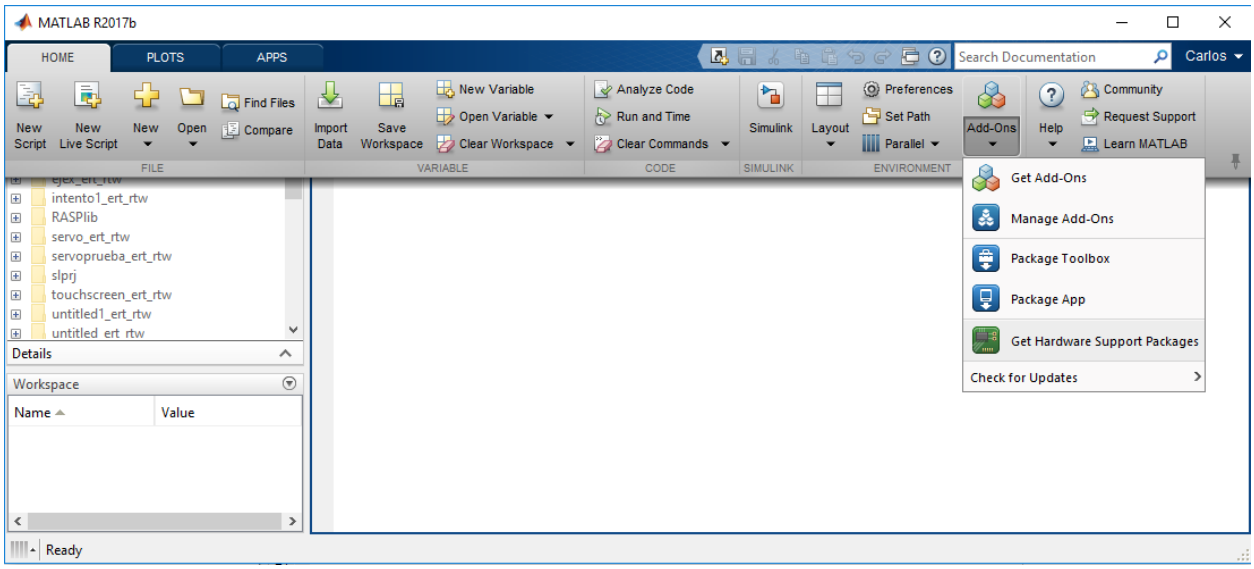


Figura 7.1.1. Paso 1. Iinstalación de Arduino Support Packages.

A continiación se abrirá una nueva ventana con todos los paqutes soportados por MATLAB y que pueden instalarse, en este caso los paquete que se debe instalar serán los tres que están marcados en la *Figura 7.1.2.*

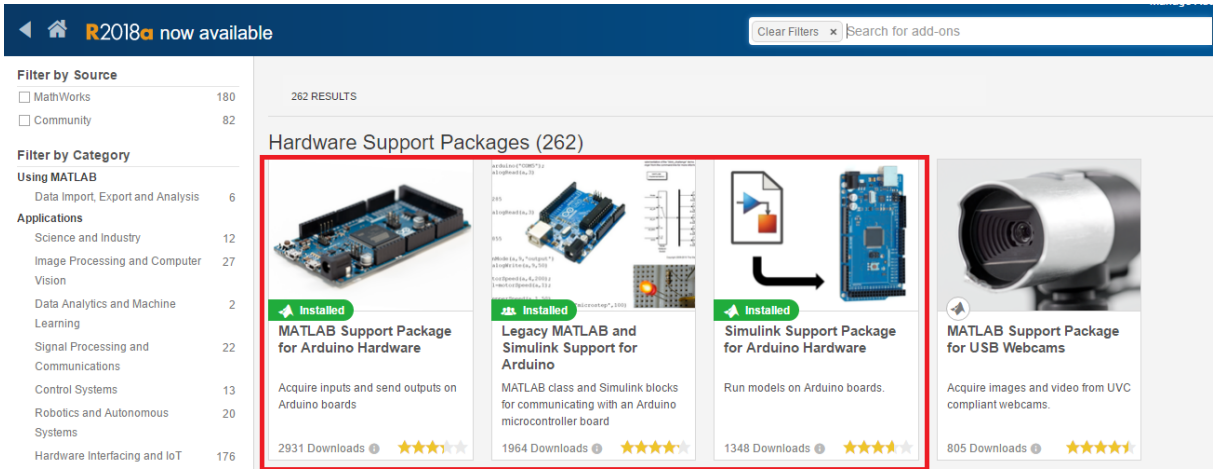


Figura 7.1.2. Paso 2. Iinstalación de Arduino Support Packages.

Para ello se habrá que seleccionar cada uno de ellos y aparecerá una nueva ventana que se muestra en la *Figura 7.1.3*. En la parte derecha de esta ventana aparecerá un botón en azul con la inscripción ‘Add’, habrá que seleccionar este botón y a continuación ‘Add to MATLAB’ del menú que se despliega.

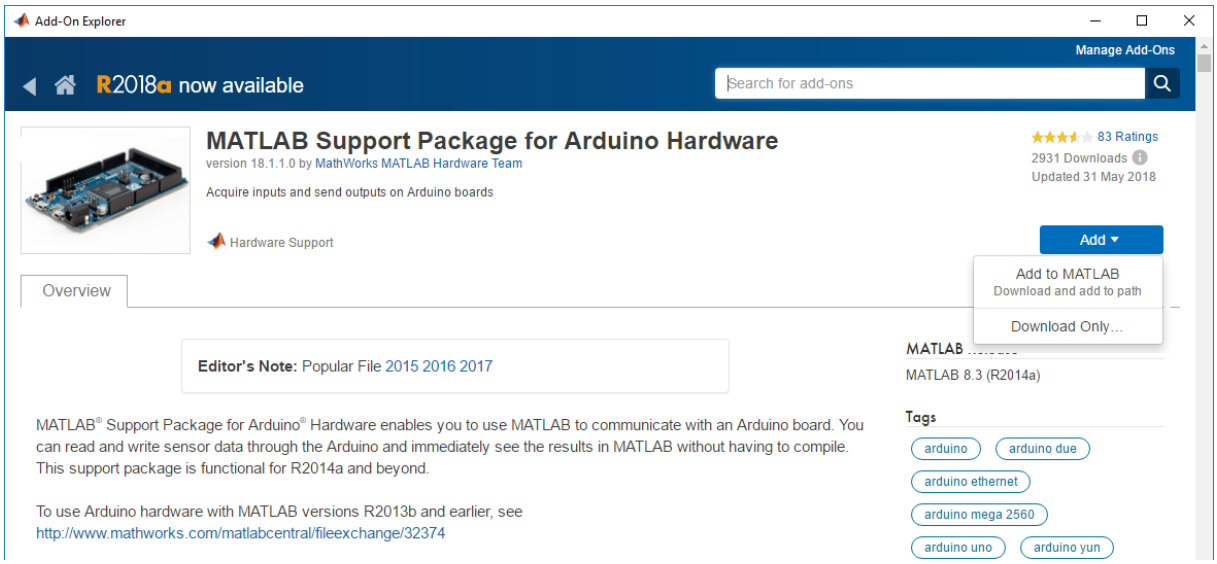


Figura 7.1.3. Paso 3. Iinstalación de Arduino Support Packages.

Habrá que registrarse en la cuenta de Mathworks y si no se dispone de ella, habrá que registrarse para poder descargar el paquete de Arduino.

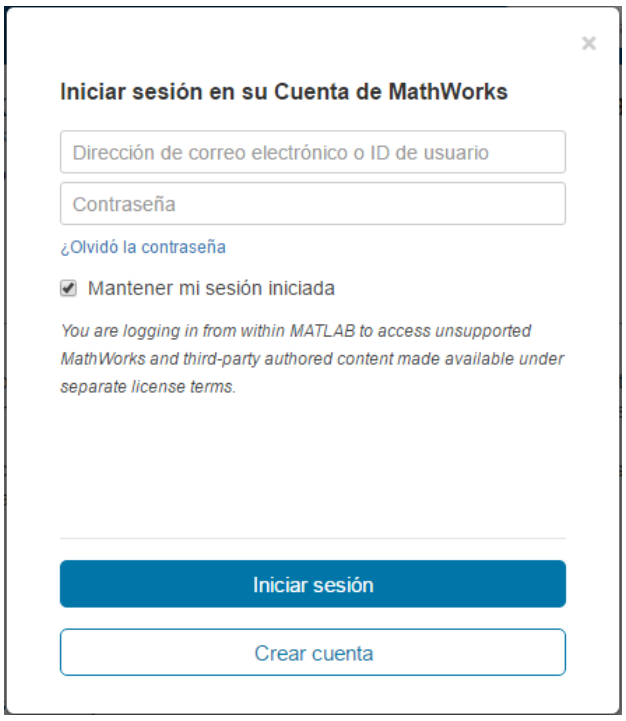


Figura 7.1.4. Paso 4. Iinstalación de Arduino Support Packages.

Una vez registrado, se deberá aceptar las condiciones de acuerdo de licencia que se muestran en la *Figura 7.1.5* y ya estará instalado el paquete deseado.

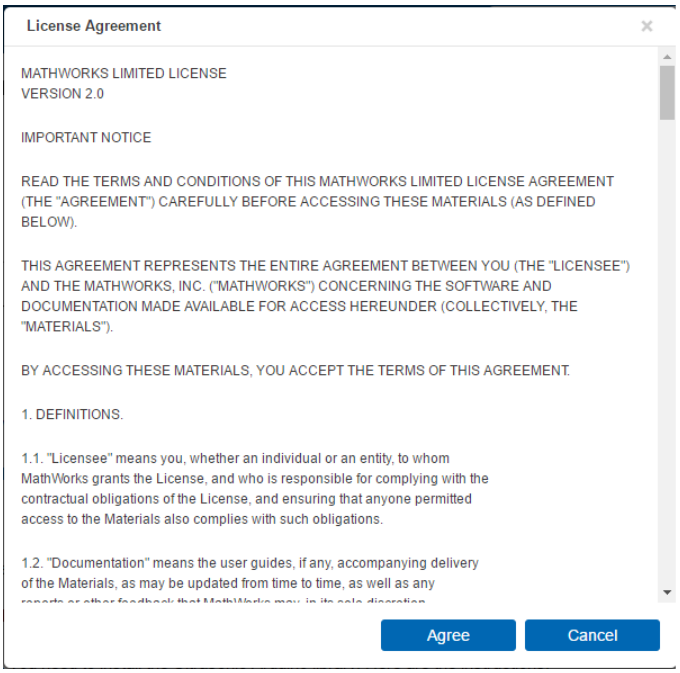


Figura 7.1.5. Paso 5. Iinstalación de Arduino Support Packages.

Después de la instalación exitosa del paquete de soporte, se agregarán las bibliotecas Arduino en MATLAB Simulink. Abrir la Biblioteca Simulink haciendo clic en la opción del menú SIMULINK en la pestaña INICIO. Simulink Library se abrirá en el lado izquierdo y, junto con todas las bibliotecas disponibles, habrá dos bibliotecas Arduino IO library y Simulink Support Package para la biblioteca Arduino. (Figura 7.1.6)

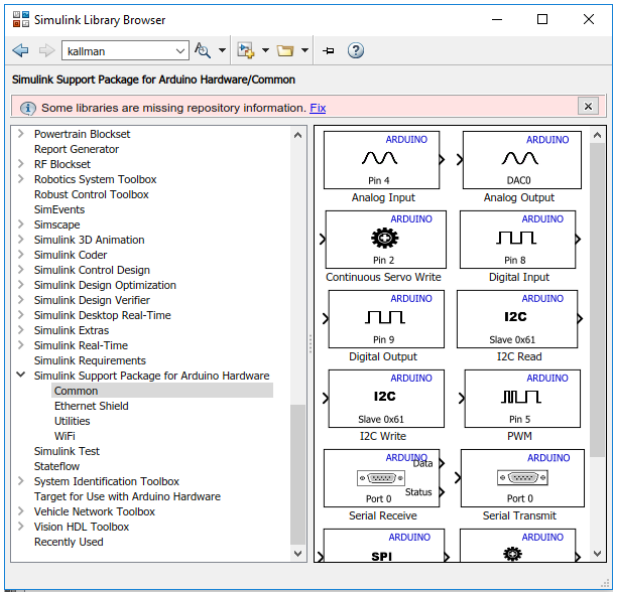


Figura 7.1.6. Librería Support Package for Arduino Hardware.

7.2. DIAGRAMA DE BLOQUES DE CONTROL

Se va a tratar de adaptar el algoritmo de control diseñado en la parte de simulación al dispositivo Hardware del sistema, con el objetivo de controlar la posición de la bola en un sistema real. Para ello se modificará el algoritmo con los nuevos bloques necesarios para comunicación con Arduino (*Figura 7.2.1*).

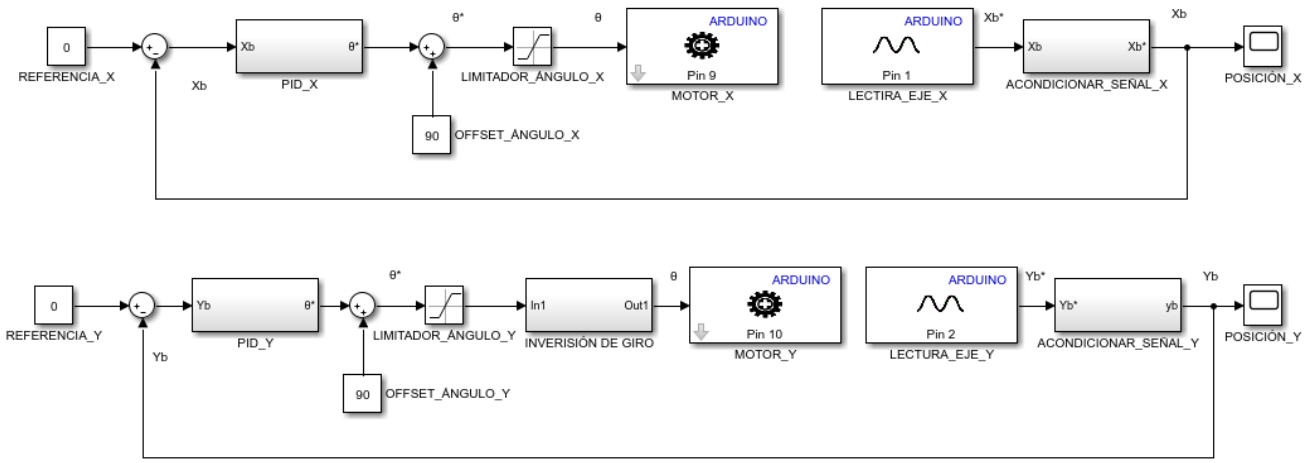


Figura 7.2.1. Control Hardware real.

Como se aprecia en la figura, el control se realiza controlando cada eje por separado tal y como se demostró con las simulaciones. Se ha partido de la misma disposición diseñada anteriormente aunque relaizando los cambios necesarios añadiendo los bolques de ardunio para adaprarlos al Hardware real. Se ha sustituido el bolque que contenía la dinámica del motor por el bolque del motor real y el bloque que contenía la dinámica de la relación del ángulo del plato junto con la dinámica de la posición de la bola, deducidas en el modelado matemático, por el bloque que representa la lectura analógica de la pantalla táctil, la cuál indica la posición de la bola en cada momento. La *Figura 7.2.2* hace referencia al interior del bloque de los controladores, en el que se realiza la suma de la acción proporcional, derivativa e integral del sistema con la novedad de que se ha añadido un switch manual en cada acción para tener opción de añadirla o quitarla según interese y así poder hacer un control PID, PD, PI o P en función de las necesidades del sistema. El bloque del controlador será el mismo tanto en X con en Y.

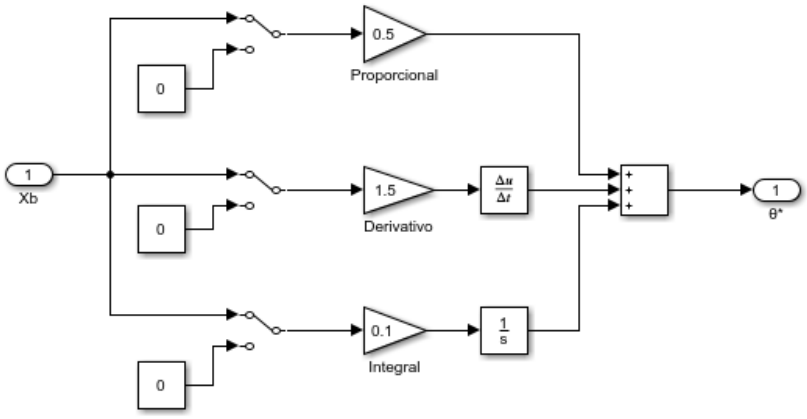


Figura 7.2.2. Controlador

El valor de la constante de referencia en la que la bola estará en equilibrio en este caso será 0. La posición real de la bola se restará al valor de referencia y este dato será la entrada al controlador que tratará de ajustar el sistema para obtener una salida de valor igual al de referencia, que será el punto de equilibrio. El offset de los ángulos que se aprecia en la *Figura 7.2.1* será de 90º. Esto es debido a que cuando el controlador consiga estabilizar la posición de la bola en la posición 0, el ángulo de los motoros de los motores deberá ser 90º para mantener la posición del plato totalmente en horizontal. Se eligió este ángulo de reposo para los motores ya que el valor de la acción para mover el servo motor está comprendida entre 0º y 180º, entonces 90º será el punto intermedio. Es importante considerar que según la superficie donde se apoye el hardward podrá variar el ángulo de reposo de los motores. Se aprecia también en la *Figura 7.2.1* que en el eje Y justo antes de enviar el dato del ángulo al bloque del motor hay un bloque llamado 'Inversión de giro' que en el eje X no aparece. Esto es debido a que el sistema se ha diseñado con la intención de que cuando el ángulo de giro esté comprendido entre el valor de 90º y 180º el bazo del motor gire hacia arriba y cuando se encuetre en el intervalo comprendido entre 0º y 90º el brazo del motor se desplace hacia abajo. Según la disposición en la que se encuentran colocados los motores en el Hardware del sistema, el brazo del motor del eje X si se desplaza según esta condición, sin embargo, el brazo del motor del motor del eje Y realiza el movimiento en sentido contratio por tanto, se diseñó un bolque para invertir su giro que se muestra en la *Figura 7.2.3.*

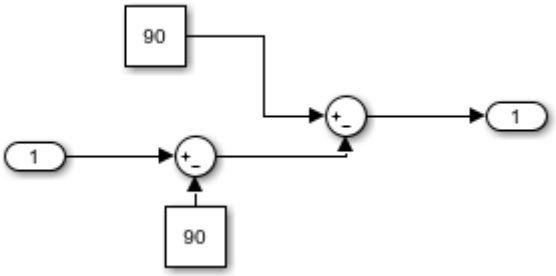


Figura 7.2.3. Inversión de giro del motor del eje Y.

El funionamiento de este bloque es muy sencillo, consiste en que si el controlador envía una orden al motor de desplazarse a 100º por ejemplo, la intención es que el brazo del motor se desplace 10º hacia arriba pero como se ha explicado anteriormente, debido a la posición en la que se encuentra colocado el motor en el Hardware, el brazo del motor en este caso se desplazaría 10º hacia abajo. La orden para que este motor funcione de manera correcta debería ser 80º para ello se hace un doble offest con respecto a la posición de equilibrio que es 90º, de esta manera si al ángulo recibido se le resta 90º se obtiene el ángulo de giro que debe de realizar el motor, entonces se cambia de signo dicho ángulo y se corrige la posición con el segundo offset. A continuación se muestra el ejemplo práctico.

$$100º - 90º = 10º \quad ; \quad 90º - 10º = 80º$$

Otro bloque a tener en cuenta para controlar el ángulo de los motores será el saturador llamado 'LIMITADOR_ÁNGULO', su función será limitar el giro del motor a ±15º con respecto a los 90º (posición de equilibrio) para evitar giros bruscos y así facilitar el control de la bola. Habrá que hacer doble click sobre él y fijar el límite superior e inferior deseado (Figura 7.2.4). Se aplicará el mismo límite de giro tanto al eje X como al eje Y.

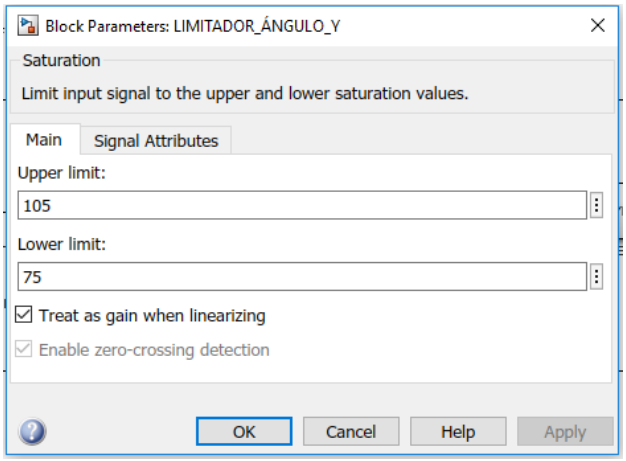


Figura 7.2.4. Límite superior e inferior de los motores.

La lectura de las coordenadas X e Y de la pantalla se realiza directamente a través de los bloques llamados 'LECTURA_EJE_X' y 'LECTURA_EJE_Y' que se pueden observar en la *Figura 7.2.1*. Las pantallas táctiles resistivas están construidas en base a dos capas de un material conductor (óxido de indio y estaño) que presentan una resistencia a la corriente eléctrica que es función de la longitud de dicho material, por tanto simplemente habrá que realizar una lectura analógica en cada eje y se obtendrá el valor que marca según la distancia donde se encuentre la bola. Para el caso de las pantallas táctiles resistivas de 4 hilos como la utilizada en este proyecto *(Figura 7.2.5)*, **no se puede realizar la lectura de los dos ejes a la vez** ya que hay dos hilos que comparten los dos ejes y se comportarán de una manera diferente según la posición del eje que se esté leyendo.

Figura 7.2.5. Nomenclatura de los 4 hilos de la pantalla táctil.

Para realizar la lectura del eje X, habrá que conectar X1 a +5V, X2 a GND y se lee Y2 mediante una entrada analógica de Arduino, Y1 quedaría sin conexión (*Figura 7.2.6*).

Figura 7.2.6. Configuración pantalla para lectura eje X

Para realizar la lectura del eje Y, habrá que conectar Y1 a +5V, Y2 a GND y se lee X2 mediante una entrada analógica de Arduino, X1 quedaría sin conexión (*Figura 7.2.7*).

Figura 7.2.7. Configuración pantalla para lectura eje Y

Una vez obtenido el dato correspondiente en cada eje habrá que acondicionar la señal recibida para adecuarla al sistema, esto se realiza dentro de los bloques descritos como 'ACONDICIONAR_SEÑAL_X' y 'ACONDICIONAR_SEÑAL_Y' que se muestran en la *Figura 7.2.1*. A continuación se muestra detalladamente el diagrama de bloques que se encuentra en el interior de estos subsistemas (*Figura 7.2.8 y 7.2.9*). En este caso se van a mostrar los diagramas de bloques de cada eje ya que cada uno tiene una peculiaridad que es conveniente comentar.

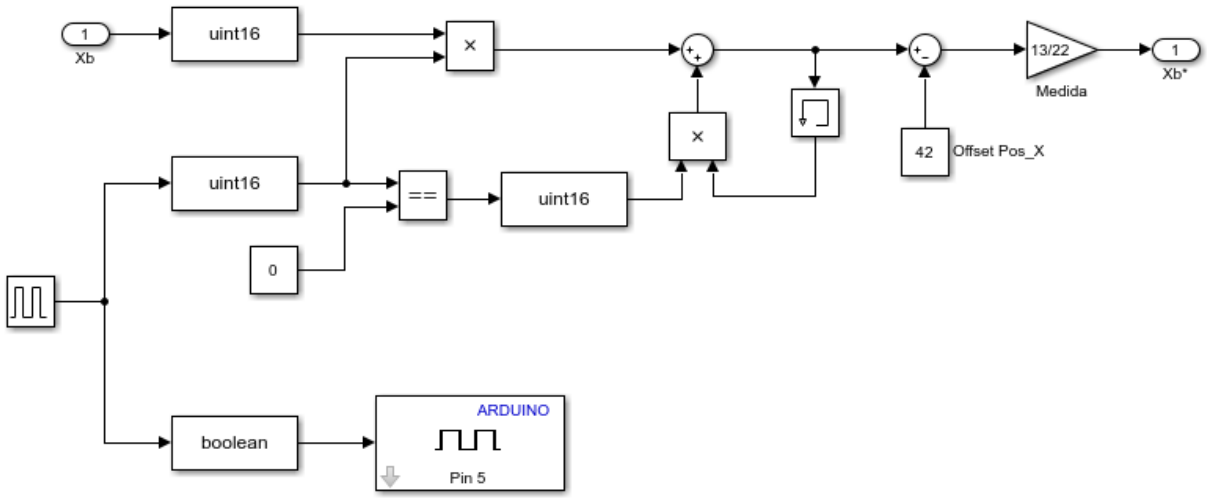


Figura 7.2.8. Acondicionamiento de la señal pantalla táctil eje X

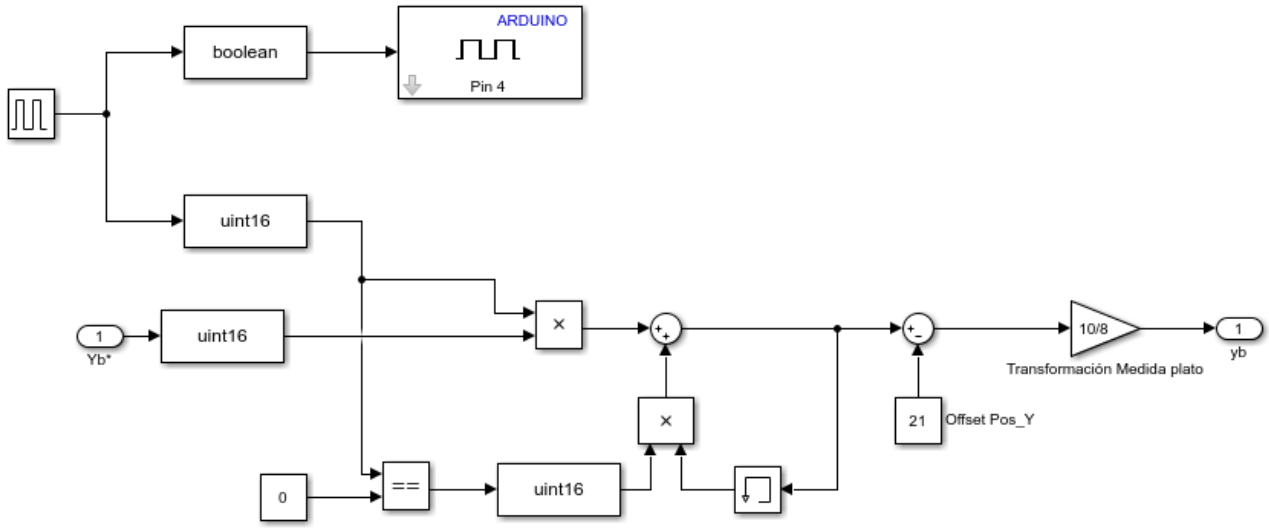


Figura 7.2.9. Acondicionamiento de la señal pantalla táctil eje Y

Dentro de cada subsistema de acondicionamiento de señal se pueden distinguir dos partes en el diagrama de bloques, Una de ellas consiste en una señal booleana que se envía a un pin digital de Arduino que está configurado como salida. Se trata de una señal cuadrada que será igual en los dos ejes pero en el Eje Y está desfasada de tal manera que cuando una de las señales está en nivel alto la otra señal estará en nivel bajo. Esto se hace porque como ya se ha comentado, no se puede realizar la lectura de los dos ejes a la vez ya que hay dos hilos comunes a los dos ejes y SIMULINK no permite asignar a un pin dos configuraciones diferentes dentro de una misma simulación, por tanto la configuración del cableado para cada eje se contralará por Hardware mediante circuitos integrados CD74HC4066 que contienen switchs internos. Cada uno de estos circuitos integrados dispone de 4 switch por lo que se necesitará uno por cada eje. La configuración de los pines se muestra en la *Figura 7.2.10*

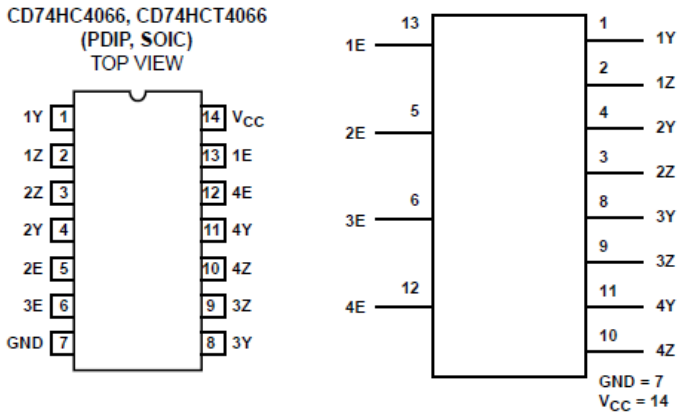


Figura 7.2.10. Pines circuito integrado CD74HC4066

Disponen de cuatro entradas E que controlan las salidas Y-Z, es decir, si en la entrada E se tiene un nivel alto, las salidas correspondientes Y-Z quedan puenteadas actuando como un interruptor. Los pines digitales 4 y 5 que generan la señales cuadradas de las *Figuras 7.2.8 y 7.2.9* controlarán cuando estará disponeble la configuración del eje X o la del eje Y, para ello todas las entradas E de cada circuito integrado estarán puenteadas de tal manera que cuando la señal cuadrada correspondiente al eje X se encuentre en nivel alto se conectará a las entradas de su circuito integrado y se cortocircuitará las 4 salidas Y-Z que estarán preparadas para formar el conexionado correspondiente a la *Figura 7.2.6.* Cuando esta señal cambie a nivel bajo, la señal cuadrada del eje y pasará a nivel alto y se conectará a las entradas E del circuito integrado que controla el conexioado del eje Y cortocircuitando todas las salidas Y-Z que formarán el conexionado de la *Figura 7.2.7.* A continuación se muestra el esquema de conexiones del circuito comentado (*Figura 7.2.11).*

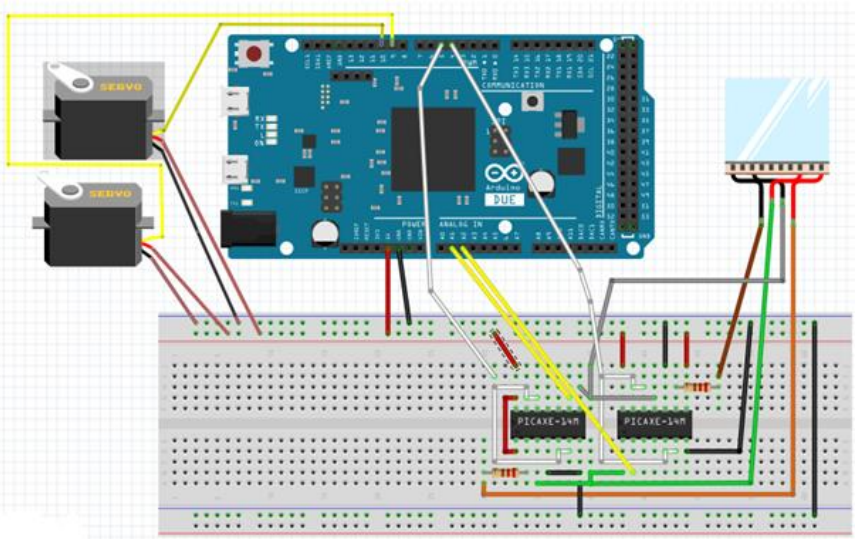


Figura 7.2.11. Esquema de conexiones.

La otra parte del diagrama de bloques correspondiente a las *Figuras 7.2.8 y 7.2.9* consiste en una serie de operaciones lógicas que serán las encargadas de acondicionar la señal de la pantalla leída por las entradas analógicas. Dentro del bloque de entrada analógica de SIMULINK se puede configurar cada cuanto tiempo se quiere obtener una lectura del sensor conectado a ese pin, pero esto condiciona a que el resto de entradas analógicas que haya en el programa deberán realizar la lectura al mismo tiempo por tanto, si se realizara la lectura de las dos entradas analógicas conectadas a los sensores de la pantalla del eje X e Y en el mismo tiempo, se obtendría en un ciclo el valor de la posición del eje X y en el eje Y en ese momento no se obtendría nada y al ciclo siguiente obtendríamos el valor de eje Y y en el eje X nada y así sucesivamente, lo cual no sería viable para la realización del proyecto. Para solucionar este problema, se ha añadido un bloque que guarda en memoria el resultado obtenido en el ciclo anterior y, realizando una serie de operaciones lógicas, se consigue que si el valor de la señal cuadrada de un eje está en nivel alto se muestre en la salida el valor real leído en el eje correspondiente en ese momento y que cuando la señal cuadrada esté a nivel bajo se mantenga a la salida el valor leído en el ciclo anterior, así nunca se obtendrá un valor nulo en la lectura.

Por último, los bloques correspondientes a ‘offset Pos_X’ y ‘offset Pos_Y’ se encargarán de realizar una corrección de la medida de la pantalla para obtener valores positivos a un lado del centro y valores negativos al otro lado del centro del eje dejando el valor 0 en el justo en el centro. Para este caso en concreto, se comprobó experimentalmente que la media de la pantalla en el eje X iba desde el 0 hasta el valor 84, por tanto se le asignó un offset de valor 42 para obtener un rango de medidas entre -42 y 42 a lo largo del eje X y que en el centro quedara el valor 0. En el eje Y se comprobó experimentalmente también y se

aplicó un offset de 21. La ganancia 'Medida' divide la medida máxima real de la pantalla desde el centro hasta el final de cada eje (13 cm en el eje X, 10 cm en el eje Y) por la medida máxima obtenida experimentalmente para obtener a la salida una medida real de la posición de la bola en centímetros.

7.3. SHIELD ARDUINO PARA HARDWARE DEL SISTEMA

Para realizar el diseño de la placa Shield de Arduino Due necesaria para realizar la lectura de los dos ejes de la pantalla, se ha utilizado la herramienta de diseño EAGLE que es un programa de diseño de circuitos electrónicos y PCBs.

Al ejecutar el programa se abrirá la siguiente ventana de inicio indicada en la *Figura 7.3.1.* Habrá que crear un nuevo proyecto, para ello habrá que dirigirse a la barra de herramientas del programa y seleccionar File → New → Project.

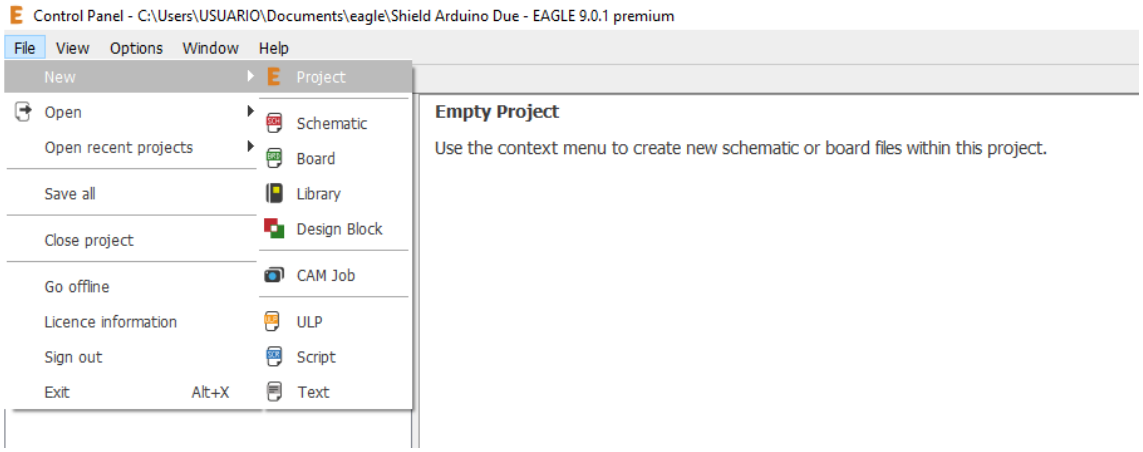


Figura 7.3.1. Ventana principal Eagle. New Project.

Una vez creado, hacer click con el botón derecho sobre la carpeta que se ha creado para el nuevo proyecto y seleccionar New → Schematic como muestra la *Figura 7.3.2*

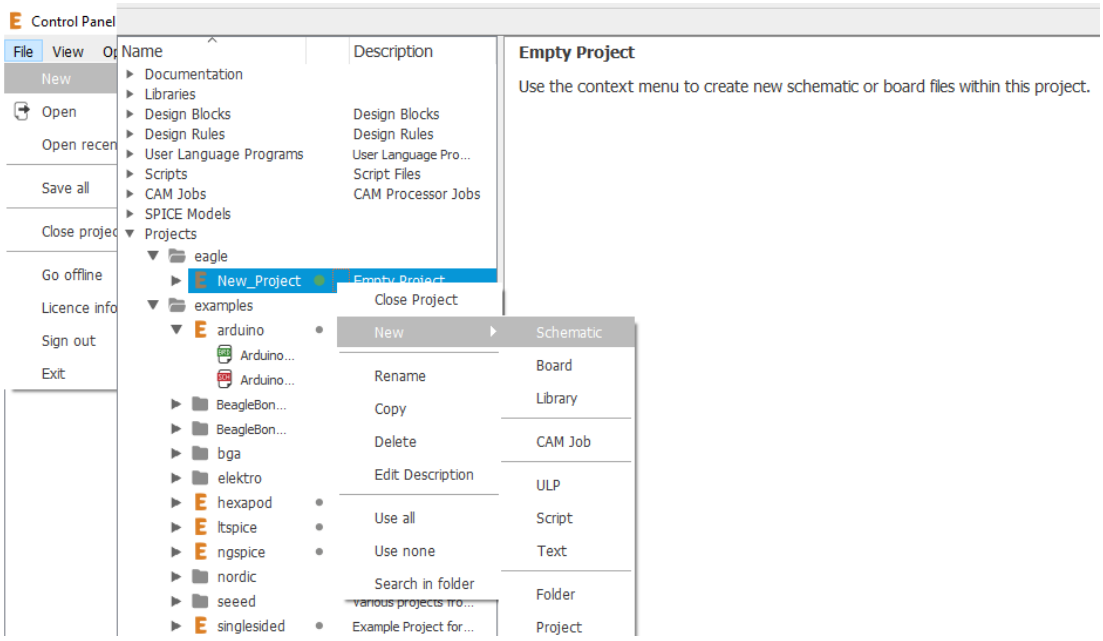


Figura 7.3.2. New Schematic

En la ventana Schematic se procederá a realizar el circuito electrónico antes diseñado. Para ello, habrá que buscar en las librerías del programa los componentes que se vayan a necesitar. Se pulsa el botón Add, señalado en rojo en la Figura 7.3.3, y en el buscador se introduce el nombre del componente que se desee añadir.

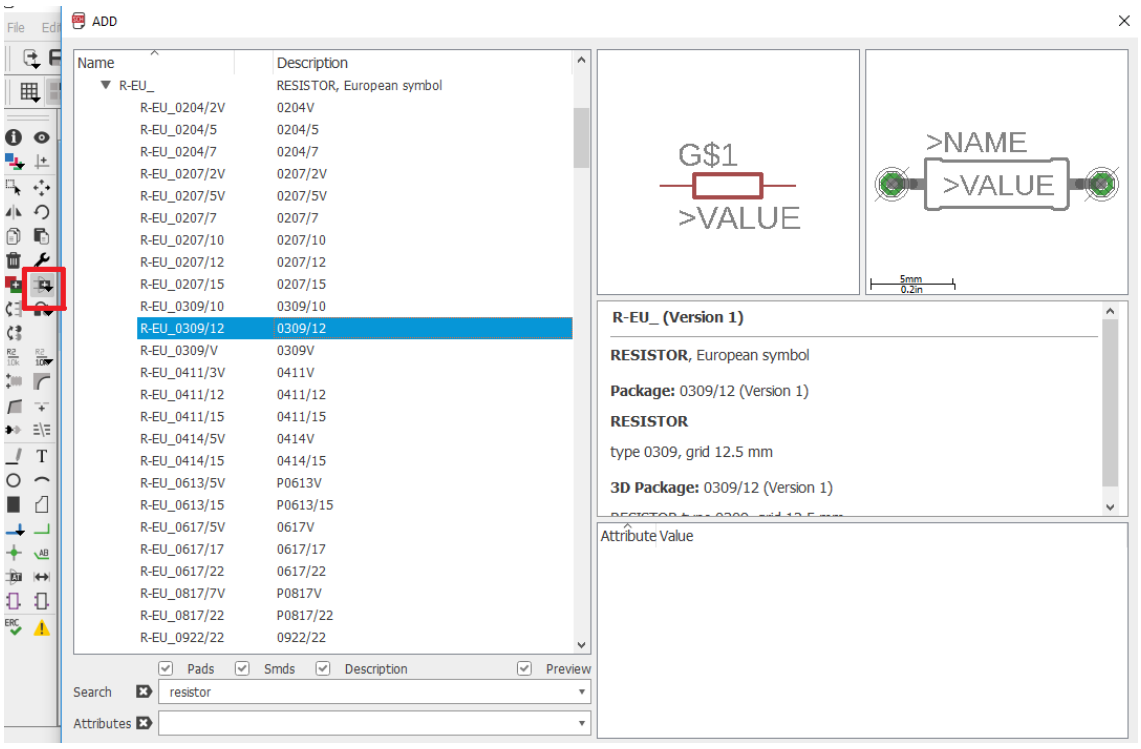


Figura 7.3.3. Añadir componentes en Schematic.

Si el componente buscado no se encuentra en las librerías, como es el caso del circuito integrado CD74HC4066 que es el que se ha usado en este proyecto, habrá que crearlo y añadirlo. En la pantalla de inicio del programa habrá que dirigirse a la barra de herramientas y seleccionar File → New → Library y nos aparecerá una nueva ventana que se muestra en la *Figura 7.3.4*.

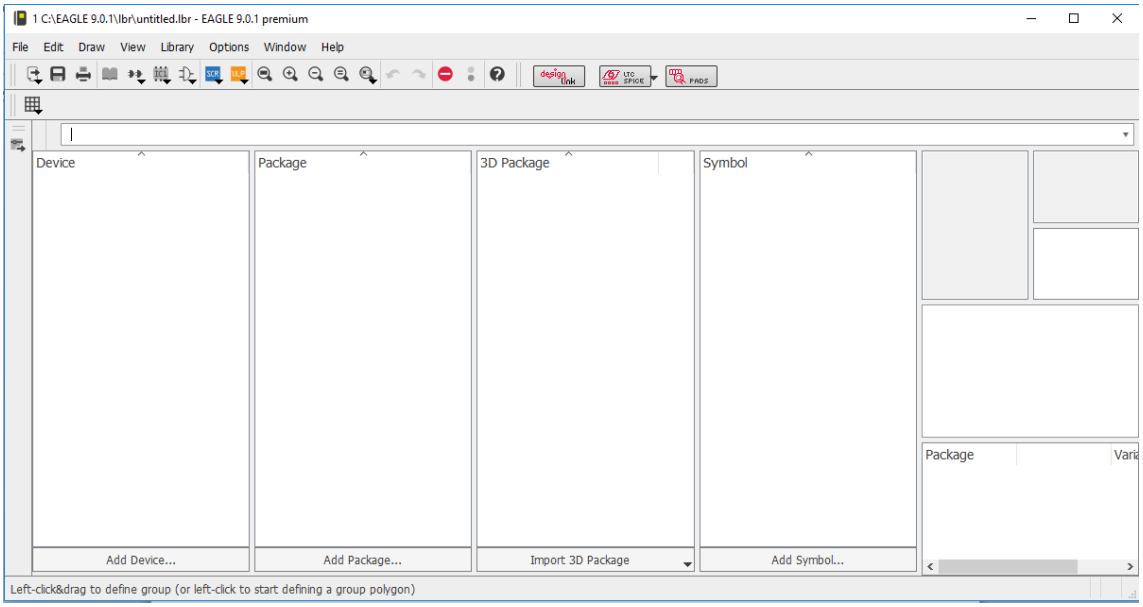


Figura 7.3.4. Añadir nueva librería.

En Symbol, habrá que pinchar en 'Add Symbol' y en la ventana que emerge dibujar el componente en cuestión como se muestra en la *figura 7.3.5*. Este dibujo será el que aparecerá en el Schematic cuando se vaya a realizar el conexionado del circuito.

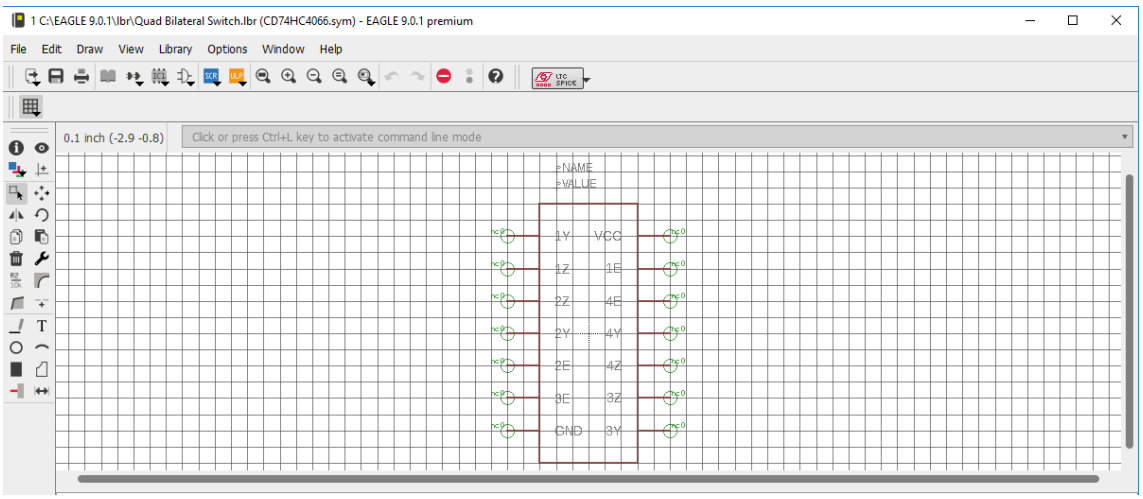


Figura 7.3.5. Add Symbol.

Una vez dibujado el componente se guarda y el siguiente paso será asignar una huella a este componente. En este caso el encapsulado del circuito integrado es un empaquetado de doble hilera o dual in-line package (DIP o DIL), consiste en un bloque con dos hileras paralelas de pines. La nomenclatura normal para designarlos es «DILn», donde "n" es el número de pines totales del circuito. Por tanto, la huella de este circuito integrado será DIL14 que quiere decir que tiene 14 pines, con 7 en cada fila. La separación estándar entre dos pines o terminales es de 0,1 pulgadas (2,54 mm). Habrá que hacer click ahora en 'Add Package'→Import y buscar la huella DIL14 que se muestra en la Figura 7.3.6.

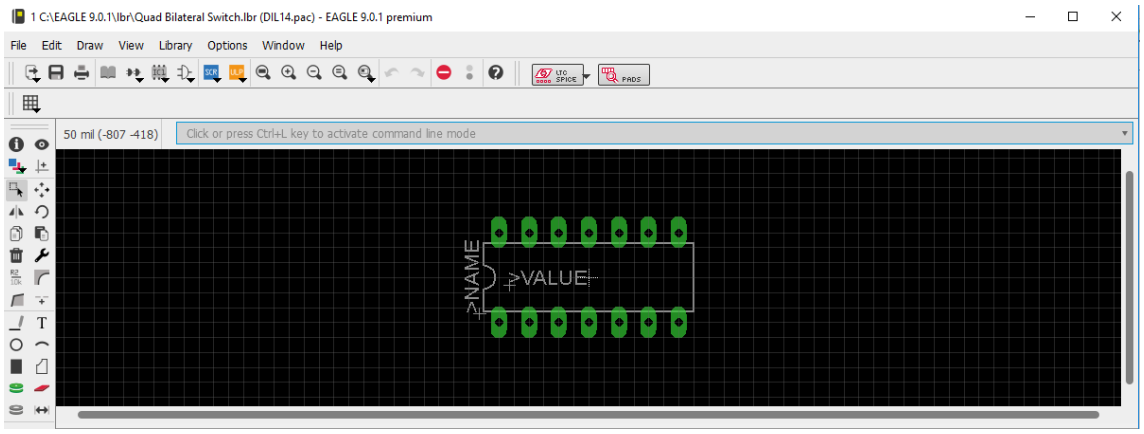


Figura 7.3.6. Package.

El último paso sería relacionar los pines del dibujo creado en 'Symbol' con los pines correspondientes de la huella 'Package'. Para ello habrá que entrar en 'Add Device' y pinchar sobre el botón conectar como se muestra en la Figura 7.3.7.

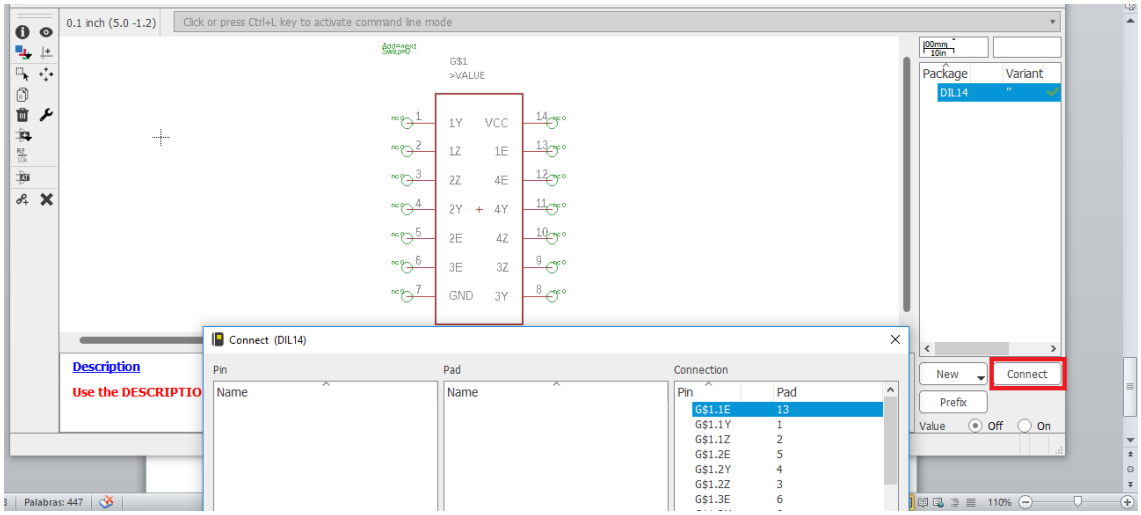


Figura 7.3.7. Device.

Una vez creada la librería con el nuevo componente ya se podrá realizar las conexiones del circuito electrónico diseñado. Se necesitará dos circuitos integrados CD74HC4066 (nuevo componente creado), un Shield de Arduino Due (Se puede encontrar en las librerías), dos conectores de tres pines para los motores, dos resistencias y un conector de cuatro pines para los 4 hilos de la pantalla. A continuación se realiza el conexionado tal y como se diseñó y el circuito quedará como se muestra en la *Figura 7.3.8.*

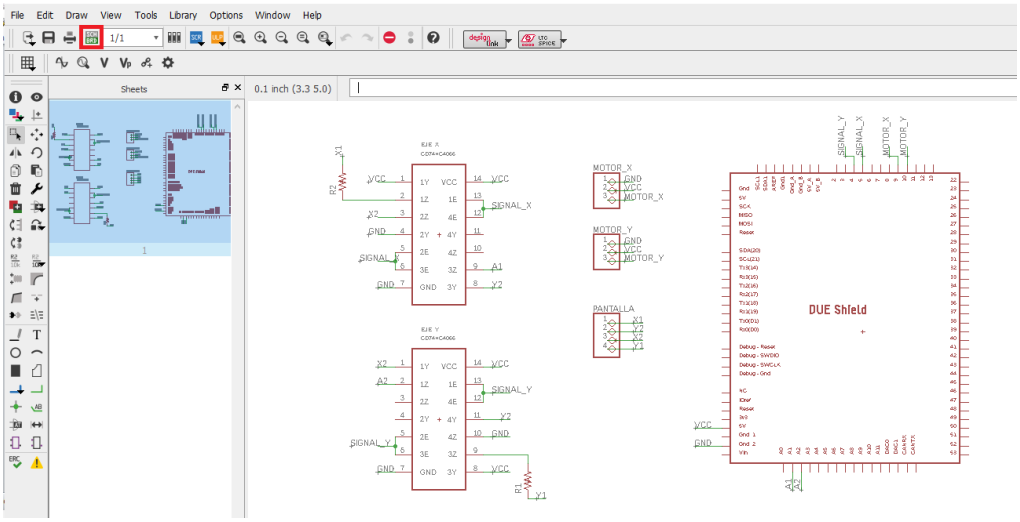


Figura 7.3.8. Conexionado Schematic.

Una vez realizado el esquemático, habrá que pulsar el botón de la barra de herramientas señalado en rojo en la figura anterior *Figura 7.3.8* 'Generate/Switch to board' y aparecerá una nueva ventana en la que habrá que habrá que definir el tamaño de la placa diseñada, distribuir todos las huellas de los componentes encima de la placa y realizar el ruteo de las pistas. Para el caso de la placa diseñada para este proyecto, quedaría de la siguiente manera indicada en la *Figura 7.3.9.*

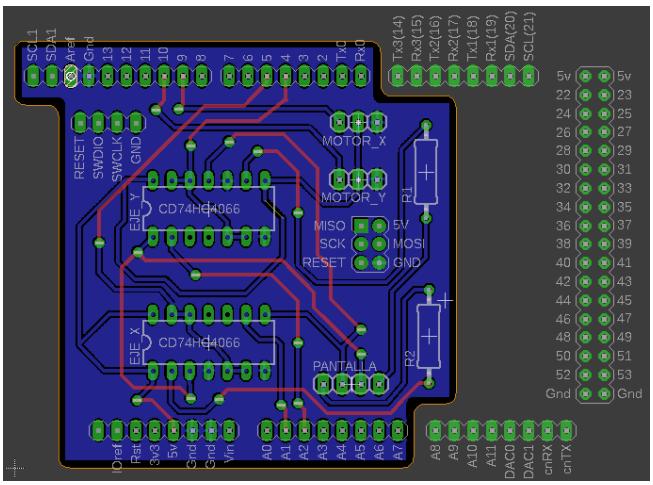


Figura 7.3.9. Shild PCB para Arduino Due.

7.4. COMUNICACIÓN ARDUINO-SIMULINK

Para ejecutar el algoritmo de control realizado en Simulink en la placa de Arduino conectada al Hardware del sistema habrá que modificar los parámetros de configuración. Para ello habrá que dirigirse a la barra de herramientas de Simulink y seleccionar la opción Tools → Run on Target Hardware → Options… tal y como muestra la *Figura 7.4.1.*

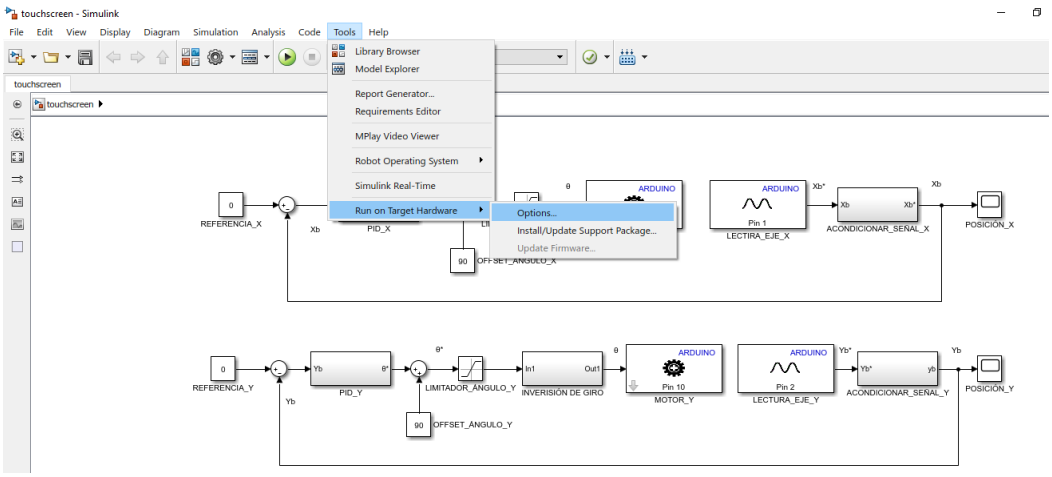


Figura 7.4.1. Acceder a parámetros de configuración.

A continuación se abrirá la ventana emergente 'Configuration Parameters' que se muestra en la figura 7.4.2. Primero habrá que dirigirse a la pestaña 'Solver' y configurar el 'start time' y 'stop time', en este caso habrá que configurarlo con tiempo inicial 0 y tiempo final infinito para así poder visualizar el proceso en tiempo real sin que pare de avanzar en el tiempo hasta que se desee. También habrá que desplegar la opción 'Additional parameters' y configurar el tiempo de paso fijo como automático.

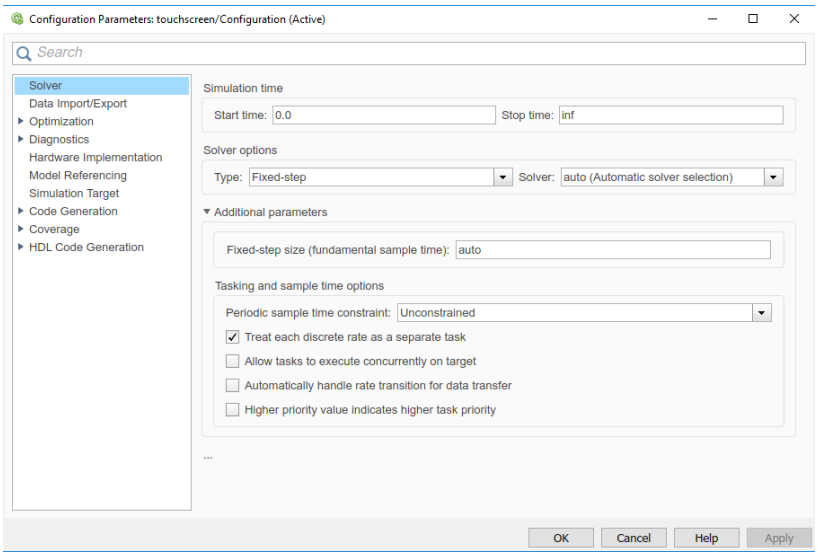


Figura 7.4.2. Configuration Parameters. Solver.

Ahora habrá que dirigirse a la pestaña 'Hardware Implementation' mostrada en la *Figura 7.4.3*. Aquí habrá que abrir el desplegable 'Hardware board' y buscar en la lista el tipo de tarjeta de Arduino que se esté utilizando, en este caso será Arduino Due. El microcontrolador saldrá por defecto según el tipo de Arduino elegido. En esta pestaña también habrá que hacer click en 'Target Hardware resources' y en el menú que se despliega seleccionar 'Host-board connection', desplegar la lista emergente y elegir la opción 'Automatically'. Esto es para que al ejecutar el programa se reconozca el puerto donde está conectado el Arduino de forma automática.

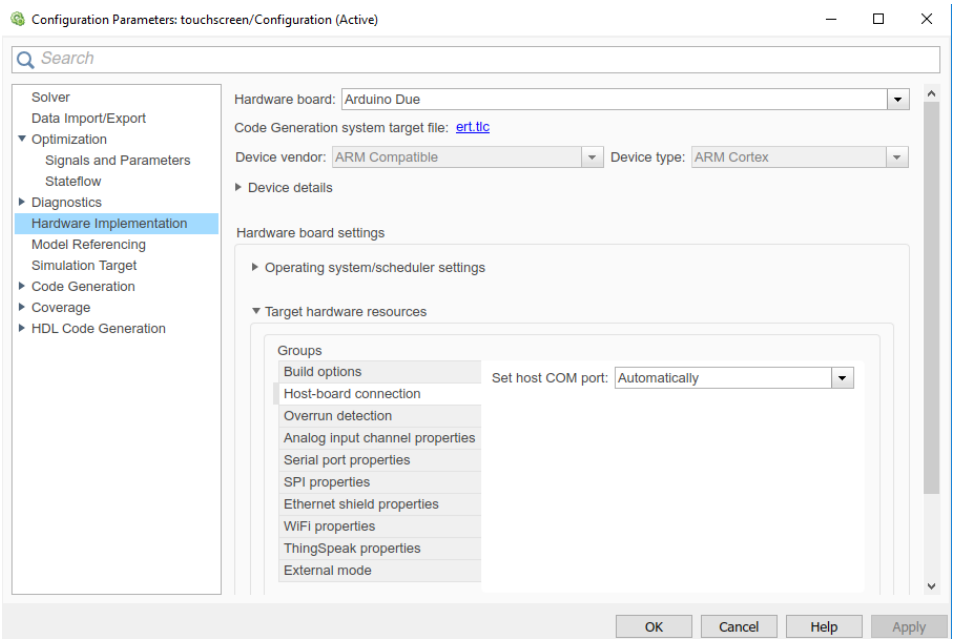


Figura 7.4.3. Configuration Parameters. Hardware Implementation.

Todo lo demás que aparece en la configuración de parámetros se deja por defecto. Por tanto, una vez realizados los cambios descritos se debe seleccionar 'Apply' y 'OK' para guardar los cambios y volver al algoritmo de bloques de Simulink. Por último habrá que dirigirse a la barra de tareas, seleccionar el desplegable que aparece y elegir la opción 'external', se comprueba que el tiempo de simulación está en infinito y ya estará preparado para ejecutar el programa seleccionando el botón 'Run'. *Figura 7.4.4.*

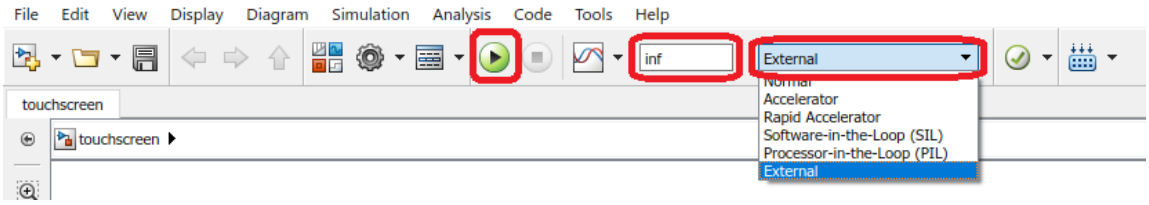


Figura 7.4.4. Configuración modo externo.

8. CONTROL DE LA MAQUETA EN TIEMPO REAL : DIRECTAMENTE CON UN MICROCONTROLADOR

Se va a proceder ahora a implementar el controlador PID mediante la consola de Arduino. La gran ventaja que se obtiene al utilizar este procedimiento es que tiene más posibilidades a la hora de programar y que se trata de un código sencillo, sin embargo se pierde la posibilidad de ver la evolución del proceso en tiempo real. Otra desventaja con respecto a la programación en Simulink es que con los bloques obtenemos un algoritmo más visual y más directo al aplicar los resultados obtenidos en el modelado matemático y en las simulaciones..

8.1. LECTURA DE LOS EJES DE LA PANTALLA

La lectura de los valores de los ejes de la pantalla tiene la peculiaridad que se ha comentado anteriormente, no se puede realizar la lectura de los dos ejes a la vez. Esto ocasionó que el algoritmo implementado mediante Simulink fuera más complejo, sin embargo si se implementa el código con Arduino se podrá realizar la lectura de la pantalla de manera mucho más sencilla ya que nos permite realizar una función para la lectura de cada eje. Para ello habrá que asignar un pin analógico a cada uno de los hilos de la pantalla y, según convenga caso, dentro de cada función se podrá definir cada pin como entrada o salida, facilitando así el proceso de lectura. En la figura 8.1.1 se muestra como se realiza el código descrito. [9]

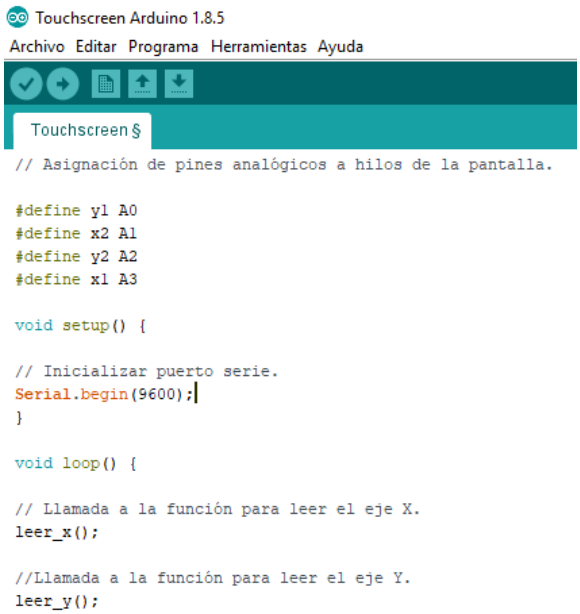


Figura 8.1.1. Programa principal para realizar la lectura de la pantalla.

Dentro de la función ‘leer_x ( )’ se van a definir los pines X1, X2 como salidas y los pines Y1, Y2 como entradas, entonces se alimenta X1 a VCC, X2 se deriva a GND y se procede a realizar una lectura analógica en el pin Y2 para obtener la posición en el eje X. Dentro de la función ‘leer_y ( )’ se siguen los mismos pasos pero definiendo Y1, Y2 como salidas, se asigna VCC a Y1 y GND a Y2 y se lee X2 para obtener el valor de la posición en el eje Y. La figura 8.1.2 muestra el algoritmo para realizar este código.

```
void leer_x()
{
// Definir X1, X2 como salidas, definir Y1, Y2 como entradas
pinMode(x1, OUTPUT);
pinMode(x2, OUTPUT);
pinMode(y1, INPUT);
pinMode(y2, INPUT);

//Asignar VCC a X1 y GND a X2
digitalWrite(x1, HIGH);
digitalWrite(x2, LOW);

//Lectyra analógica del pin Y2
int coordenada_x=analogRead(A2);

//Imprimir valor en puerto serie
Serial.print("El valor de X es: ");
Serial.println(coordenada_x);
}
```

Figura 8.1.2. Código de la function ‘leer_x( )’.

La función ‘leer_y ()’ sería igual pero introduciendo los cambios descritos en la definición de los pines.

8.2. CONTROL DE LOS SERVOMOTORES

Los servomotores permiten mantener una posición indicada, siempre que esté dentro del rango de operación del propio dispositivo. El ángulo de giro, en este caso permite hacer un barrido entre -90º y 90º, lo que viene a ser un ángulo de giro de 180º. Aunque el servo puede moverse con una resolución de más de 1 grado, este es el máximo de resolución que se va a conseguir debido a la limitación de la señal PWM que es capaz de generar Arduino. Estos motores funcionan con una señal PWM con un pulso de trabajo entre 1 ms y 2 ms y con un periodo de 20 ms (50 Hz). Este dato indica la velocidad máxima a la que se puede mover el servomotor con Arduino, solo se podrá cambiar de posición cada 20 ms. [10]

Arduino dispone de una la librería de servo en la cual ofrece diferentes opciones para el control del servo motor. Existe la función ‘Write()’ que opera solo con valores comprendidos entre 0 y 180, que son los grados de giro del servomotor, que inicializa en 90º como la

posición de inicio. Y también dispone de la función 'WriteMicroseconds()' que introduce el valor en microsegundos del pulso que se mantiene en activo. Está definido en un rango de 1000 y 2000 microsegundos dónde 1500 es el valor neutro. Para este caso en concreto se utilizará la función 'WriteMicroseconds()' ya que se obtiene un cambio de posición del servo más rápida y precisa.

Para poner en práctica el control del servo se ha realizado un pequeño algoritmo que sirve también para determinar cuál será la posición de reposo de cada uno de los servos para que el plato del sistema se mantenga en una posición horizontal con respecto al plano de apoyo. Es recomendable ejecutar este código para ajustar las posiciones de equilibrio de los servos antes de poner en funcionamiento el sistema 'Ball and Plate', ya que la inclinación de la base donde se encuentre apoyado en cada momento puede variar. En la figura 8.2.1 se va a mostrar este código en el que primero habrá que incluir la librería del servo de Arduino, definir los servos de los que se va a disponer y definir una variable tipo 'float' para cada servo que se disponga.

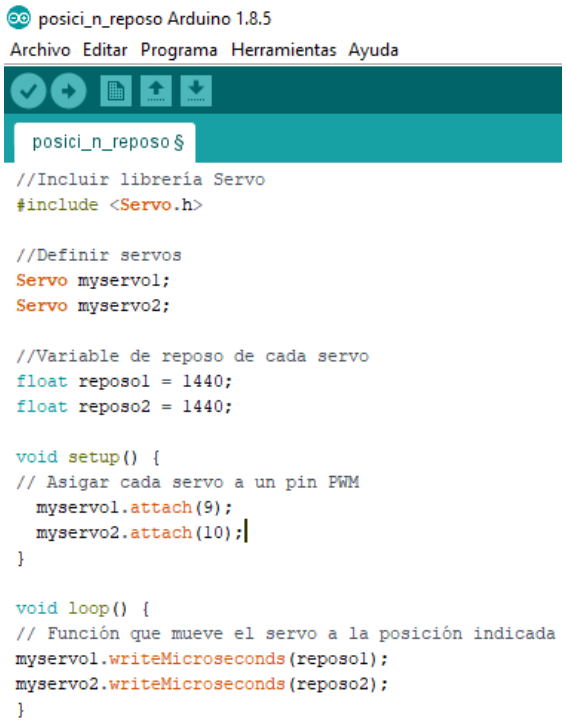


Figura 8.2.1. Código para posición de reposo de los servos.

Para encontrar la posición de reposo habrá que modificar los valores de las variables de los microsegundos en torno a un valor de 1500 microsegundos, que es valor teórico central, hasta comprobar que la bola se mantiene en equilibrio encima del plato, entonces habrá que copiar estos valores y utilizarlos en el algoritmo de control PID que se describe a continuación.

8.3. CONTROL PID

Una vez que ya se ha definido el funcionamiento del sensor y de los actuadores del sistema, se puede proceder a realizar el algoritmo de control PID. En la cabecera del programa habrá que añadir la librería del servo de Arduino, definir los pines de la pantalla táctil, definir los dos servomotores de los que se va a disponer y por último definir todas las variables que van a ser necesarias para llevar a cabo el algoritmo.

Ball_and_plate §

```
//Librería Servo
#include <Servo.h>
//Hilos pantalla táctil
#define y1 A0
#define x2 A1
#define y2 A2
#define x1 A3
//Definir Servos
Servo myservo_x;
Servo myservo_y;
//Variables valor reposo para los servos
float reposo_x = 1450;
float reposo_y = 1440;
//variables para posición
int x;
int y;
float pos_x=0;
float pos_y=0;
//Variables para la velocidad
int nvel_x=3;
int vel_x=0;
int v_x[5];
int anterior_x=0;
int nvel_y=3;
int vel_y=0;
int v_y[5];
int anterior_y=0;
//Variables para implementar el PID
float kp=0.7;
float kd=1.6;
float ki=1;
float I; // Valor Integral
//Límites para definir el centro de la pantalla
int lim1 = -50;
int lim2 = 50;
```

Figura 8.3.1. Cabecera del programa 'Ball_and_plate'.

En el 'setup' se inicializa el puerto serie, se asigna un pin PWM a cada servomotor y se llevan a la posición de reposo antes definida en el algoritmo mostrado en la figura 8.2.1.

```
void setup() {
//Inicializar puerto serie.
  Serial.begin(9600);
//Asignar Pines PWM para los servos
  myservo_x.attach(9);
  myservo_y.attach(10);
//LLevar los servos a la posición de reposo
  myservo_x.writeMicroseconds(reposo_x);
  myservo_y.writeMicroseconds(reposo_y);
```

Figura 8.3.2.Setup programa 'Ball_and_plate'

Una vez definidas todas las variables y todos los parámetros, se procederá a desarrollar el programa en el 'loop'. Para la implementación del PID se debe conocer la información de la posición y de la velocidad de la bola en todo momento, que se multiplicarán por la constante proporcional y derivativa respectivamente. Haciendo el sumatorio de estos dos productos junto con la acción integral si fuera necesaria, se obtiene el controlador PID. Este valor será positivo o negativo dependiendo para que lado interese inclinar el plato para centrar la bola y se sumará o restará al valor de reposo de los servomotores. Con esta operación se obtendrá el resultado de los microsegundos que se debe mover cada uno de los servomotores que cada vez deberá ser menor hasta llegar al valor 0, lo cual mantendría los servomotores en su posición de reposo y eso indicaría que la bola está en centrada en el punto de equilibrio.

La posición de la bola la conocemos a través de la lectura de la pantalla que se ha explicado en el apartado 8.1 de este documento pero habrá que acondicionar la señal obtenida para adaptarla al sistema estudiado. Al realizar una prueba de lectura de la pantalla se comprueba que se obtiene un valor que varía entre 130 y 830 a lo largo del eje X y un valor que comprendido entre 190 y 804 lo largo del eje Y. Para acondicionar esta señal debemos añadir un offset de -120 en el eje X para obtener un rango de posición de 0-700 y un offset de -190 para obtener un rango de 0-614 para el eje Y. Además habrá que añadir otro offset en cada eje para dejar el 0 en el centro y valores positivos hacia un lado y valores negativos hacia el otro lado. En este caso habrá que añadir un offset de -350 en el eje X y un offset de -307 en el eje Y, de esta forma ya se habrá acondicionado la señal obteniendo un rango de valores de [-350,350] en el eje X y de [-307,307] en el eje Y consiguiendo así que el 0 corresponda con el centro de cada eje. También se deberá establecer unos límites que establezcan un pequeño rango en el centro de cada eje en el que el valor sea 0 para que cuando la bola se encuentre dentro de este rango sea más fácil equilibrarla. Estos límites serán [50, -50], dentro de ese rango de valores se obtendrá valor 0 en la posición en cada eje.

```
// offset para dejar el 0 en el centro
y=coordenada_y-190-307;

if ((y>(liml)) && (y<lim2))
  {
    y=0;
  }
```

Figura 8.3.2. Offset y zona 0 eje Y

Para el eje X se aplicaría el mismo código pero aplicando el valor de los offset para este caso.

Una vez que se ha acondicionado la señal, se va a proceder a calcular la velocidad de la bola a partir de la posición. En SIMULINK es muy sencillo obtener la velocidad ya que solo basta con añadir un bloque derivativo a la señal obtenida de la posición, sin embargo en Arduino habrá que calcularla mediante código implementando la *Ecuación 8.3.1*.

$$V_x = \frac{\Delta x}{\Delta t} = \frac{x_1 - x_0}{t_1 - t_0} \quad ; \quad V_y = \frac{\Delta y}{\Delta t} = \frac{y_1 - y_0}{t_1 - t_0} \tag{8.3.1}$$

Dónde:

V_x , V_y → Velocidad real en el eje X e Y.

$\Delta x , \Delta y \rightarrow$ Variación de posición en el eje X e Y.

$\Delta t \rightarrow$ Variación en tiempo.

Habrá que hacer la diferencia de la posición de la bola tomada en el ciclo actual con la posición obtenida en el ciclo anterior. En cuanto al tiempo lo podemos obviar porque va a ser constante ya que se obtiene una medida cada ciclo de compilación de código, entonces la variación del tiempo va a ser siempre la misma. Entonces simplemente habrá que añadir las líneas de código que se muestran en la *Figura 8.3.3* justo después de las líneas de código que se encargan de obtener la posición de la bola en cada eje.

```
//VELOCIDAD X
vel_x=(x - anterior_x);
 anterior_x=x;
```

Figura 8.3.3. Cálculo de la velocidad.

La acción integral del sistema se va a integrar como se muestra en la *Figura 8.3.4*. Consiste en hacer que a medida que la bola se aleje más del centro, la acción integral se irá incrementando con una acción tipo contador y cuando la bola pase por el centro del plato la acción integral se reiniciará. Con esto se consigue que cuanto más alejada se encuentre la bola más se incremente el valor del controlador para así enviar un ángulo de giro más grande al servomotor y acercar la bola al centro lo antes posible aunque la función principal de la acción integral consiste en disminuir el error en estado estacionario. Es decir, si la bola se encuentra en una posición muy próxima al centro pero sin llegar a estarlo y el controlador no es capaz de llevarla al centro porque la señal que envía a los servomotores es casi nula debido a la proximidad de la bola, la acción integral se iría incrementando hasta conseguir obtener una señal en el controlador suficiente para elevar un poco más el servomotor y así llevar la bola al 0. Una vez que la bola se encuentre en el 0 la acción integral se reiniciaría y quedaría todo en equilibrio.

```
// Integral
    if(abs(y)>lim2){ // Solo si esta fuera de los límites liml y lim2
      I_y=I_y+y*ki;
    }
    else {
      I_y=0;
    }
```

Figura 8.3.4. Código para obtener la acción integral.

Con esto se obtendría la acción Integral del sistema, para la acción proporcional y derivativa e multiplica la posición de la bola obtenida por la constante proporcional y la velocidad de la bola calculada por la constante derivativa y ya sólo quedaría añadir el controlador al algoritmo. Para ello se partirá de la ecuación general del controlador PID Ecuación 8.3.2. La salida de estos tres términos, el proporcional, el integral, y el derivativo son sumados para calcular la salida del controlador PID. Este valor es el que se sumará o restará al valor de reposo de los servomotores para corregir su posición, hasta conseguir que la salida del controlador sea 0 y quede todo en equilibrio. Se realizará un controlador por cada eje.

$$\theta_i(t) = pos\_i * K_p + vel\_i * K_d + I \qquad (8.3.2)$$

Una vez implantado se realizó una prueba para ver el funcionamiento y se observó que los servomotores hacían movimientos muy bruscos con ángulos muy grandes que ocasionaban que la bola se desplazase más rápido de la cuenta y fuera más difícil el control. Para solucionar este problema se limitó la salida del controlador para evitar que enviara ángulos muy grandes a los servomotores. En este caso se impuso la condición que si la salida del controlador es superior a 100ms, el valor de la salida será 100ms. En la *Figura 8.3.5* se muestra el código del controlador implantado en el eje X, para el eje Y sería igual pero cambiando las variables correspondientes a dicho eje.

```
//PID_X

  pos_x=(kp*x+kd*vel_x+I_x);

//Limitación de la salida del controlador
if ((pos_x)>100)
{
 pos_x=100;
}
 if ((pos_x)<(-100))
{
 pos_x=-100;
}

//Señal corregida del servomotor
myservo_x.writeMicroseconds(pos_x+reposo_x);
```

Figura 8.3.5. Código controlador PID

9. CONCLUSIONES

La motivación para realizar este proyecto ha sido la de asumir el reto de construir un Hardware físico y trasladar los conocimientos estudiados a un sistema real para así comprender mejor los conceptos estudiados. Realmente el sistema estudiado “Ball and Plate” no es un proyecto novedoso, se han realizado varios proyectos de este tipo anteriormente y hoy en día se puede encontrar mucha información acerca de este proyecto, pero la parte diferencial ha sido la de realizar el control del sistema mediante programación de bloques y poder observar la evolución del sistema en tiempo real, además de realizar el diseño y construcción de un Hardware estable que se adapte al proceso. También hay que destacar que en este estudio se ha querido profundizar sobre todo en la parte del modelado matemático para explicarlo siguiendo todos los pasos y justificando los procedimientos utilizados en la resolución del mismo, ya que se considera que se trata de un proceso complejo y difícil de

entender. En cuanto a las simulaciones, se ha hecho un estudio relacionado con los resultados obtenidos en el modelo matemático obteniendo un resultado positivo y demostrando que las simplificaciones realizadas para poder linealizar el sistema son correctas. Con esto se demuestra que se ha realizado un amplio trabajo en la parte teórica del proyecto y que no ha consistido sólo en fabricar un Hardware y cargar el código para que funcione.

Durante la realización de este proyecto han surgido algunos puntos de bloqueo que han ocasionado ciertas dificultades citadas a continuación:

- Uno de los puntos críticos del proyecto ha sido conseguir leer la posición de los dos ejes de la pantalla a la vez mediante Simulink, esto se solucionó realizando una placa de circuito impreso que realiza el control mediante Hardware. Aun así la señal recibida era bastante ruidosa, por ello hubo que añadir un filtro digital adecuado para modular la señal.
- La parte del modelado matemático fue complicada de entender ya que se basa en principios físicos que hubo que aprender a desarrollar mediante métodos matemáticos de nivel avanzado.
- Adaptar los resultados del modelado matemático a bloques de Simulink para su posterior simulación y sobre todo, adaptar las simulaciones a bloques que corresponden al sistema real. Hubo que realizar algunos ajustes ya que todos los parámetros que influyen en el sistema real no son contemplados en las simulaciones, como por ejemplo vibraciones, holguras, condiciones climatológicas…
- Otra dificultad ha sido la de encontrar una pieza que hiciera de eje central y permitiera a la pantalla girar en los dos ejes, para ello se tuvo que recopilar varias piezas mecánicas y unirlas mediante soldadura para así conseguir la pieza deseada.
- La pantalla utilizada tiene unas dimensiones superiores a las de una impresora 3D convencional, por tanto hubo que adaptar el diseño del marco y dividirlo en dos partes con un sistema que permite unirlas una vez que se ha introducido la pantalla.
- Realmente el principal problema del proyecto ha sido el tiempo de muestreo obtenido mediante Simulink ya que para obtener un control óptimo de la posición de la bola, se necesita obtener información de la pantalla en un tiempo del orden de microsegundos y no se consiguió llegar a un tiempo de muestreo tan bajo por lo que hace el sistema sea algo lento.

En conclusión, ha sido un proyecto con un resultado positivo desde el punto de vista personal que ha servido para desarrollar y profundizar en conceptos estudiados a lo largo de la formación profesional ya que abarca varios puntos importantes para la realización de proyectos, desde el estudio teórico y simulaciones del proyecto a la construcción y puesta en marcha del mismo.

Se proponen algunas posibles mejoras futuras que podrían servir como propuesta para un futuro trabajo fin de grado o máster realizado por otro alumno:

- Reducir el tiempo de muestreo en Simulink.
- Hacer un estudio completo para realizar un filtro digital óptimo para este proyecto en concreto.
- Añadir tres interruptores en el Hardware que permitan añadir o quitar la acción proporcional, derivativa e integral del controlador y así poder observar la evolución del sistema en función de estos parámetros.
- Realizar un diseño del marco mejorado que sea de una sola pieza.
- Hacer un tipo de control diferente al realizado en este proyecto, como por ejemplo hacer que la bola se desplace de un punto específico a otro haciendo que se equilibre en cada uno de ellos.
- Programar secuencias, como por ejemplo hacer que la bola siga un movimiento circular sin pararse.
- Incluir unos Leds debajo de la pantalla y que la bola se desplace hacia el punto del Led que se encuentre encendido y se equilibre en dicho punto.

Bibliografía

[1] http://revistas.sena.edu.co/index.php/sennova/article/view/553

[2] https://es.wikipedia.org/wiki/Card%C3%A1n

[3] https://es.aliexpress.com/store/product/Win10-Compatible-4-3-12-inch-includes-USB-Controller-4-Wire-Resistive-Touch-Screen-Panel-Touch/1248879_32812023106.html?spm=a219c.12010612.0.0.4fd69ac30DN6M9

[4] http://web.engr.illinois.edu/~khashab2/files/2011_LinearControl/16.pdf

[5]http://ctms.engin.umich.edu/CTMS/index.php?example=BallBeam&section=SystemModeling

[6] https://es.wikipedia.org/wiki/Controlador_PID

[7] http://ceiisa.blogspot.com/2015/01/control-de-cascada.html

[8] http://engineerexperiences.com/arduino-matlab-installation.html

[9] https://www.youtube.com/watch?v=oRoBHfEdgpw

[10] https://es.wikipedia.org/wiki/Servomotor

Informe del Tutor/a del Trabajo Fin de Grado/Máster

Autor (Apellido1-Apellido2, Nombre)			
García Martín, Carlos			
Título del Trabajo			
DISEÑO, CONSTRUCCIÓN Y CONTROL DE UNA PLANTA BALL AND PLATE			
Titulación	Máster en Ingeniería Industrial	Especialidad/ Mención	N/A
Centro	Escuela Politécnica Superior de Jaén	Departamento	Electrónica y Automática Industrial
Tutor/a del TFG/TFM			Universidad/Institución
Elisabeth Estévez Estévez			Universidad de Jaén

Resumen Castellano (máx. 150 palabras)

En dicho trabajo el alumno ha de ser capaz de realizar un diseño mecánico de una planta Ball and Plate por otro lado también ha de ser capar de realizar el montaje eléctrico y electrónico, diseño e implementación de un sistema de control que permita controlar la posición de la bola en un plato. Para esto último el alumno tendrá que modelar el comportamiento del plato, relacionar el movimiento de la bola con respecto a la inclinación del plato así como diseñar un sistema de control que a través de dos motores (actuadores) se modifique la posición plato con el fin de controlar la posición de la bola. Para el diseño se utilizará el entorno Matlab/Simulink. Para la implementación, el controlador se implementará y ejecutará en un arduino. Finalmente, dicho algoritmo de control podrá ser parametrizable. Tanto en el laboratorio como de una forma remota.

Resumen Inglés (máx. 150 palabras)

In this work the student must be able to perform a mechanical design of a Ball and Plate plant on the other hand must also be able to perform electrical and electronic assembly, design and implementation of a control system that allows controlling the position of the ball on a plate. For this last the student will have to model the behavior of the plate, relate the movement of the ball with respect to the inclination of the plate as well as design a control system that through two motors (actuators) the plate position is modified in order to control the position of the ball. For the design the Matlab / Simulink environment will be used. For the implementation, the controller will be implemented and executed in an arduino. Finally, said control algorithm can be parameterized.

Nomenclatura Internacional de Unesco para la Ciencia y Tecnología(http://skos.um.es/unesco6/)

Códigos UNESCO	Descriptor castellano	Descriptor Inglés
3304.17	Sistemas en tienpo real	Real-time Systems
3307.03	Diseño de Circuitos	Circuit Design
3304.12	Dispositivos de Control	Control device

Observaciones y Comentarios:

Los/as Tutores/as dan el Visto Bueno para entregar y defender su Trabajo Fin de Grado/Máster

Jaén, a 03 de JULIO de 2018

ESTEVEZ ESTEVEZ ELISABET - 72578245C

Firmado digitalmente por ESTEVEZ ESTEVEZ ELISABET - 72578245C
Nombre de reconocimiento (DN): c=ES, serialNumber=72578245C, sn=ESTEVEZ ESTEVEZ, givenName=ELISABET, cn=ESTEVEZ ESTEVEZ ELISABET - 72578245C
Fecha: 2018.07.03 22:01:35 +02'00'

Fdo.:_____

SR. PRESIDENTE DEL TRIBUNAL EVALUADOR