Universidad Zaragoza
1542

Trabajo Fin de Grado

Análisis y puesta en marcha de una plataforma triaxial como demostrador tecnológico

Autor

Raúl López Martín

Director

Juan Diego Jaria Gazol / José Manuel Rodríguez Fortún

Escuela Universitaria Politécnica La Almunia

2019

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

MEMORIA

Análisis y puesta en marcha de una plataforma triaxial como demostrador tecnológico

[424.19.38]

Autor: Raúl López Martín

Director: Juan Diego Jaria Gazol / José Manuel Rodríguez Fortún

Fecha: 27/11/2019

INDICE DE CONTENIDO

1. RESUMEN

El trabajo realizado servirá para mostrar el proceso de diseño basado en modelos (MBDE) con un sistema que aúna distintos componentes y tecnologías. Para aumentar la visibilidad del dispositivo y su posterior uso como demostrador, se incluirá el control de la posición de una bola colocada sobre la plataforma, cerrándose el bucle de control mediante una cámara.

El proyecto se realizará tomando un concepto pre-existente y realizando su análisis y puesta en marcha siguiendo la metodología de diseño basado en V: verificación modular del diseño usando simulación, verificación experimental de cada componente, montaje del sistema completo y validación completa.

Los pasos que se van a seguir son: revisión de soluciones tecnológicas para plataformas triaxiales; características técnicas del diseño seleccionado; revisión y validación software; análisis cinemático basado en modelos; verificación modular del sistema de control; montaje, puesta en marcha y validación del sistema completo.

1.1. PALABRAS CLAVE

Modelo, V, análisis, verificación, validación.

2. ABSTRACT

The work done will serve to show the model-based design process (MBDE) with a system that brings together different components and technologies. In order to increase the visibility of the device and its subsequent use as a demonstrator, the control of the position of a ball placed on the platform will be included, closing the control loop by means of a camera.

The project will be carried out taking a pre-existing concept and carrying out its analysis and start-up following the V-based design methodology: modular design verification using simulation, experimental verification of each component, assembly of the complete system and complete validation.

The steps to be followed are: revision of technological solutions for triaxial platforms; technical characteristics of the selected design; software revision and validation; kinematic analysis based on models; modular verification of the control system; assembly, start-up and validation of the complete system.

3. ANTECEDENTES

Este proyecto tiene como objetivo el desarrollo y la puesta en marcha de una plataforma triaxial de tres grados de libertad con capacidad para estabilizar el movimiento de una pelota colocada sobre ella y cuya posición está controlada mediante visión por computador.

Los manipuladores paralelos planos son mecanismos que están compuestos por 2 partes, la base que es fija, y una parte superior que es móvil, ambas están unidas mediante cadenas cinemáticas cerradas, las cuales junto con los actuadores dotan a estos manipuladores de sus respectivos grados de libertad.

En la actualidad el uso de este tipo de este tipo de manipuladores está cada vez más demandado, ya que tiene diversas aplicaciones y gran precisión en su funcionamiento.

Los manipuladores paralelos, tienen una amplia gama de estructuras en función de sus cadenas cinemáticas. Para la descripción de estas estructuras, hay que definir las cadenas cinemáticas y para ello se utiliza una nomenclatura específica, las articulaciones que unen los eslabones de la cadena son representadas por letras mayúsculas, empezando por la base y terminado con la plataforma superior, en la siguiente tabla podemos observar dicha nomenclatura.

Articulación	Símbolo	Actuador	Símbolo
Rotacional: 1 g.d.l. en Rotación	R	Motor-Manivela	R
Prismática: 1 g.d.l. en Traslación	P	Lineal	P
J. Cardan: 2 g.d.l. en Rotación	K ó U ó RR		
Rótula: 3 g.d.l. en Rotación	S		

Figura 1. Notación empleada para la descripción de las cadenas cinemáticas en Robots Paralelos. (Meyer, 2014).

Algunas de las características que los diferencian de otros tipos de manipuladores son:

- Necesitan espacios de trabajo más reducidos que otros tipos de manipuladores, debido al tipo de cinemática empleada.
- Son más rígidos que un robot serie cuando se les aplica una carga sobre el elemento móvil debido a que dicha carga será soportada por varias cadenas cinemáticas en lugar de por una sola. Esto da lugar a que la relación carga soportada - masa total del robot en su elemento terminal sea superior al de un robot serie.

- Pueden realizar movimientos a velocidades y aceleraciones elevadas sin que aparezcan esfuerzos dinámicos que dificulten su realización debido a la buena relación carga soportada - masa total.

- Habitualmente este tipo de manipuladores son accionados mediante un actuador fijado sobre la base que controla la cadena cinemática, esto posibilita que la masa de los actuadores no sea considerada una masa móvil y facilite alcanzar valores de aceleración elevados.

- Otra característica positiva de este tipo de manipuladores es el hecho de que los pares cinemáticos actuados se encuentren en diferentes cadenas. Esto hace que los errores de posicionamiento no tengan una elevada amplificación en el posicionamiento del elemento superior.

Dentro de esta clase de manipuladores podemos encontrar diferentes tipologías con distintas arquitecturas.

A continuación se muestran algunas de las más representativas, el primer ejemplo es una de las más famosas, la plataforma Gough-Stewart (6 SPS), dicha plataforma es un manipulador paralelo con 6 grados de libertad, por lo que puede realizar 3 movimientos lineales, respecto a los ejes x, y, z y otros 3 movimientos rotacionales alrededor de estos mismos, denominados roll, pitch y yaw.

Esta plataforma puede trabajar con varias velocidades, altas cargas y una amplia rigidez.

Figura 2. Plataforma Gough-Stewart. (wikipedia)

Este tipo de plataformas tienen diversas aplicaciones, simulaciones de vuelo, simulaciones de diversos efectos naturales, como oleajes y terremotos, pruebas mecánicas como el estudio de vibraciones sobre un elemento y muchas otras.

Dentro de los manipuladores paralelos con 6 grados de libertad también podemos encontrar el manipulador HEXA (3 RRS), este tipo de manipulador está formado por 6 cadenas cinemáticas que unen la base con la parte superior. La parte superior es generalmente hexagonal y la base, parte de modelos geométricos triangulares. En este tipo de manipulador los actuadores están colocados en la base y cada uno acciona una articulación.

Una de las principales características del manipulador HEXA es que puede operar a altas velocidades y con cargas relativamente altas.

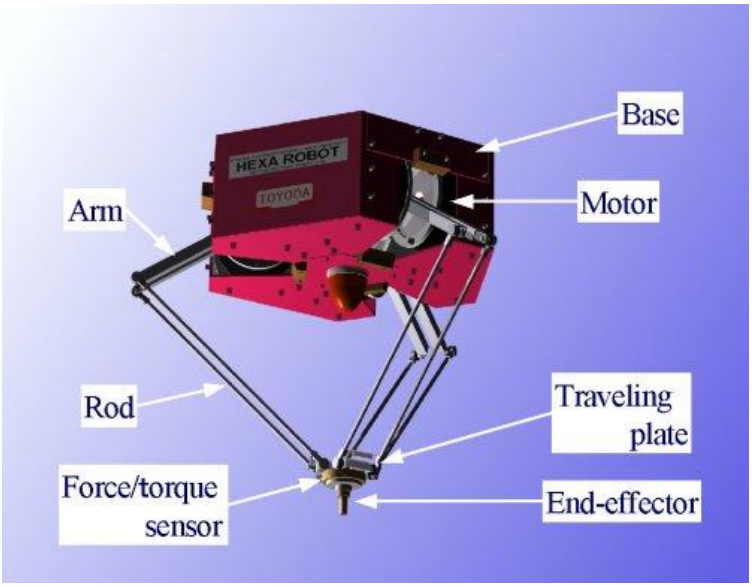


Figura 3. Hexa robot. (Picón, 2008).

Otro de los manipuladores más conocidos es el robot tipo delta (3 RRS), este robot tiene 3 grados de libertad, está compuesto por 3 cadenas cinemáticas las cuales tienen una estructura en forma de paralelogramo y están accionadas por 3 motores rotativos.

Puede soportar cargas de alrededor de 1Kg y tiene altas prestaciones en términos de aceleración y precisión.

Figura 4. Robot delta. (Picón, 2008).

El robot delta tiene diversas aplicaciones, como por ejemplo en la industria alimentaria, áreas de ensamblaje y actuaciones quirúrgicas entre otras.

El siguiente manipulador paralelo a describir es el modelo Tricept (1 UP -- 3 UPS), este tipo de manipulador tiene 3 grados de libertad y está constituido por 3 cadenas cinemáticas actuadoras y por otra cadena cinemática pasiva, la cual le permite a su elemento terminal contar con un movimiento compuesto, de 2 rotaciones independientes y una traslación, y a su vez también le aporta mayor rigidez a la estructura del mismo.

Figura 5. Robot Tricept. (Picón, 2008)

Las mayores aplicaciones y utilidades de este tipo de manipulador se dan en el campo de las operaciones de mecanizado de piezas debido a su capacidad de movimientos ya su buena precisión.

Autor: **Raúl López Martín**
[424.19.38]

También podemos encontrar otro tipo de manipuladores paralelos con 4 y 5 grados de libertad, por ejemplo el robot Adept Quattro (4 RS), el cual tiene 4 grados de libertad, este manipulador es capaz de realizar 3 traslaciones independientes y una rotación alrededor de un eje de dirección constante, una de las características más destacadas es la capacidad de realizar movimientos con altas aceleraciones, puede llegar a alcanzar aceleraciones alrededor de 200 m/s^2.

Figura 6.Robot Adept Quattro. (Picón, 2008)

Otro de los manipuladores de 4 grados de libertad que podemos encontrar, es el manipulador paralelo 4 – UPU, es un manipulador completamente paralelo y tiene una arquitectura completamente paralela y simétrica con 4 grados de libertad.

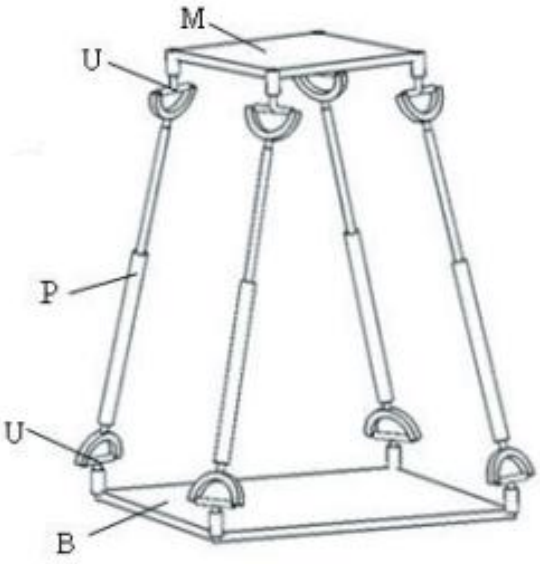


Figura 7. Manipulador paralelo 4-UPU. (Picón, 2008)

Y por ultimo podemos encontrar manipuladores de 5 grados de libertad, robots 5 – PRRRR, este tipo de robots tienen un patrón definido en el que se pueden realizar 3 traslaciones independientes, junto a 2 rotaciones independientes, definiendo un patrón 3T2R.

Figura 8. Robot HITA SST. (Picón, 2008)

Los manipuladores paralelos que poseen un elemento terminal cuya movilidad es inferior a 6 grados de libertad, se denominan manipuladores paralelos de baja movilidad, como es nuestro caso ya que nuestra plataforma contara con 3 grados de libertad.

Nuestra plataforma es un manipulador completamente paralelo ya que el número de cadenas cinemáticas que posee es igual al de los grados de libertad de su elemento terminal y presenta dichas cadenas con la misma morfología.

Todo el desarrollo se ha realizado siguiendo la metodología de diseño en V y aplicando diseños virtuales para verificar y analizar el sistema.

4. METODOLOGÍA DE TRABAJO

En el presente proyecto, se realizara el análisis y la posterior puesta en marcha de una plataforma triaxial para la estabilización de una pelota colocada en la zona superior y cuya posición está monitorizada por una cámara. Para ello se utilizara la metodología de trabajo denominada diseño en V, que consiste primero en la verificación del diseño de cada parte del conjunto utilizando cuando sea necesario la simulación, y posteriormente la verificación experimental de cada componente, seguida del montaje del sistema de forma completa y la validación en conjunto del mismo.

4.1. DISEÑO EN V

El desarrollo en V es un proceso lineal que sigue la forma de una "V" para describir el diseño de sistemas complejo. Así, el proceso comienza con una secuencia descendente en el lado izquierdo de la V, que se corresponde con el diseño de cada componte y el conjunto comenzando desde las especificaciones y los requerimientos del sistema. A continuación, se realiza la validación y verificación que se desarrolla a lo largo de la rama ascendente de la V, como se muestra en la imagen.

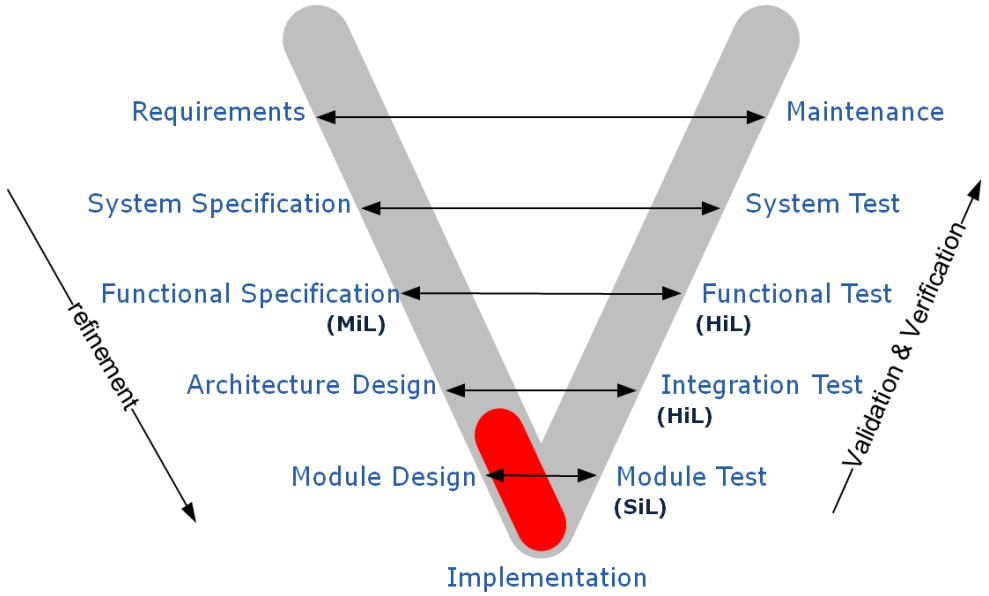


Figura 9. Modelo en V. (Ramaekers, Hermans, De Meulenaere, Denil, & Anthonis, 2011)

5. DESCRIPCIÓN DE LA PLATAFORMA Y TRABAJO REALIZADO

Nuestra plataforma tiene una arquitectura triaxial, es decir puede girar sobre el eje x, el eje y, así como desplazarse en el eje z. Para ello en la base están colocados tres servomotores separados 120º entre ellos y conectados mediante unas articulaciones libres acabadas en una rotulas con la plataforma superior.

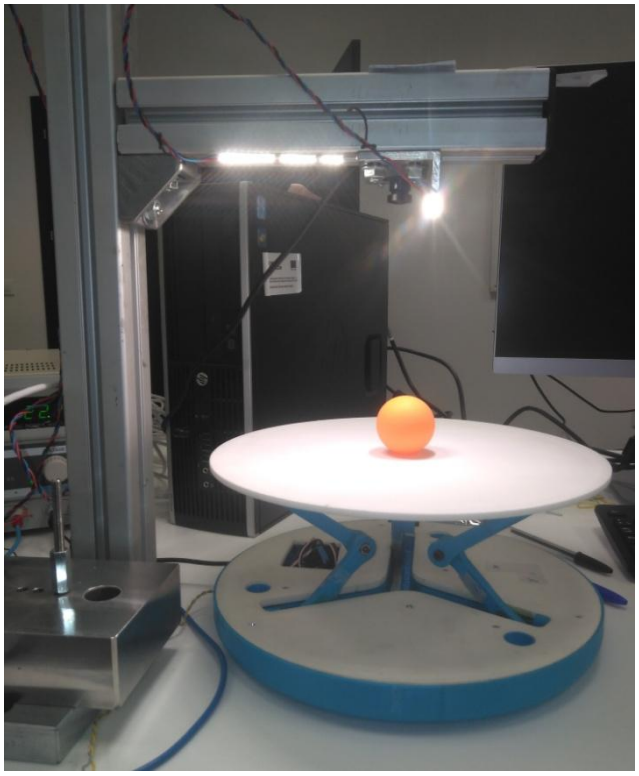


Figura 10. Plataforma triaxial.

Este proyecto parte de un diseño preexistente realizado por Johan Link y disponible en: https://www.instructables.com/id/Ball-Balancing-PID-System/

El material de referencia del cual se parte para la realización de este proyecto, es:

- Algunas de las piezas de la plataforma en formato STL para poder imprimirlas en la impresora 3D.
- El código de detección de objetos
- El control de la plataforma.

Con esos datos de partida, en el presente proyecto se ha desarrollado el trabajo siguiendo la metodología de diseño en V basado en modelos. Partiendo del análisis de los componentes en la rama descendente de la "V":

- En primer lugar se analiza la montabilidad de la estructura, materiales que necesitábamos, cuáles de ellos teníamos disponibles en el laboratorio y cuales deberíamos comprar.
- Posteriormente se realiza el análisis electrónico y de los motores para asegurar la funcionalidad y verificar el cableado.
- Una vez que el montaje de la plataforma estaba acabado, se paso a realizar el análisis cinemático, para ello hemos comparado el modelo cinemático original con un modelo generado en Recurdyn® que es un software de simulación de sistemas, y también lo hemos comparado con un modelo cinemática teórico realizado por nosotros mismos.
- Para acabar la rama descendente del modelo en V, se realiza el análisis de la arquitectura software y la implementación del algoritmo de control, distribuido en sendas plataformas PC/Python® y Arduino®.

En la rama ascendente de la metodología en V, que corresponde a la validación y a la puesta en marcha del sistema, realizamos las siguientes acciones.

- Ajuste de los elementos estructurales:
  - Componentes de la plataforma (brazos, uniones, rótulas).
  - Soporte de la cámara para aumentar su rigidez.
- Preparación del entorno de trabajo con Python® y Arduino®.
- Verificación de la electrónica y funcionamiento de los motores.
- Compilación y resolución de problemas en el software, es decir:
  - La corrección, adaptación y ajuste del código a los elementos disponibles en el entorno de ITAINNOVA.
  - Ajuste del tiempo de muestreo.
  - Ajuste del algoritmo de detección mediante visión artificial.
  - Corrección de código para el correcto movimiento de la plataforma.
  - Análisis de las necesidades de control y ajuste del PID que controla el movimiento de la plataforma.
- Por último, se realizan pruebas del sistema completo para acabar de ajustar la funcionalidad general de la plataforma.
- Revisión y análisis del diseño

En esta sección se describen las actividades de verificación y análisis correspondientes con la rama descendente de la metodología de diseño en "V".

5.1. REQUERIMIENTOS

Los requerimientos de partida del proyecto son:

- Puesta en marcha de una plataforma triaxial siguiendo el concepto de referencia. La evaluación consistirá en:
  - Prueba funcional de los motores.
  - Asegurar la estabilidad de una pelota colocada sobre su superficie y afectada de perturbaciones externas.

- Utilización en lo posible de los medios materiales disponibles en el grupo de mecatrónica del Instituto Tecnológico de Aragón:
    - Máquina de impresión 3D (BQ Prusa i3 Hephestos)
    - PC con Windows 10

6. ANÁLISIS DEL DISEÑO MECÁNICO Y FABRICACIÓN DE LAS PIEZAS

En nuestro caso, como se parte de un diseño ya existente, el primer paso ha sido el ensamblaje virtual de las piezas en el software de diseño 3D inventor para realizar un primer análisis básico. Dos puntos se han evaluado:

- Ausencia de interferencias entre las piezas
- Comprobación de que las medidas y la de las piezas eran las correctas para su posterior impresión y montaje

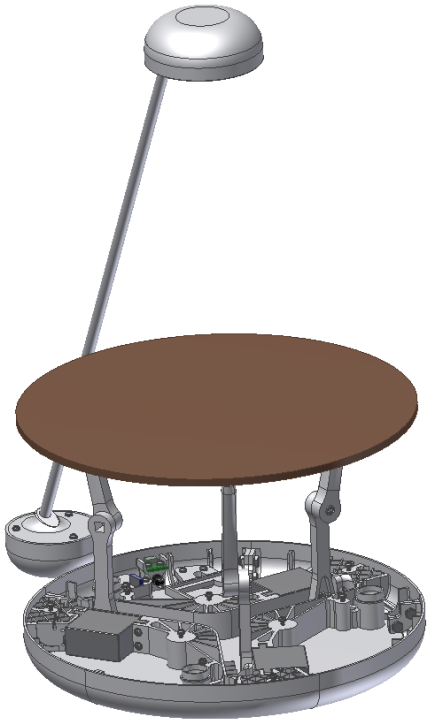


Figura 11. Ensamblaje en inventor de la plataforma.

La fabricación de las piezas, fue en su mayor parte mediante impresión 3D, tan solo la parte superior y la parte inferior de la plataforma, las cuales no cabían en la zona de impresión, no pudieron hacerse mediante impresión y fueron realizadas en prototipado rápido. La pieza superior tuvo que ser modificada debido a que al realizarla por este método salía con un pequeño alabeo el cual afectaba al control final.

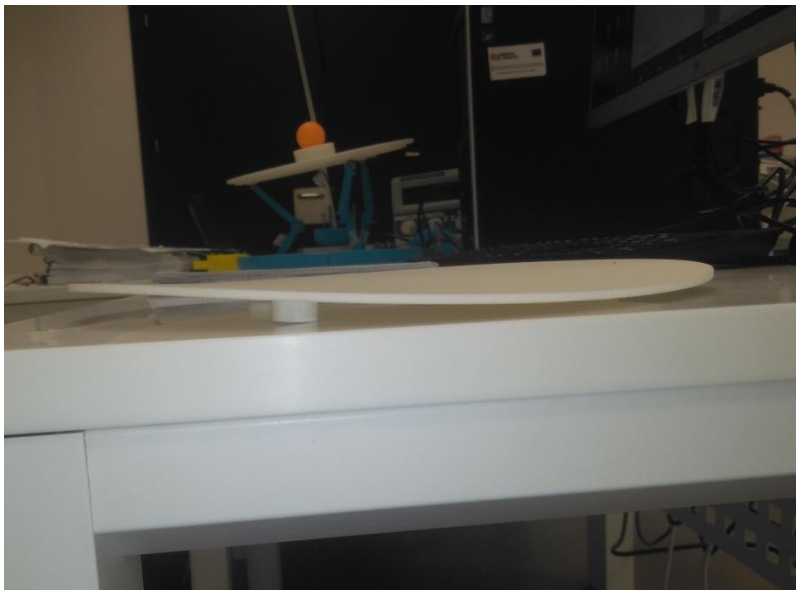


Figura 12. Pieza superior deformada

La modificación de la plataforma superior consistió en añadir los nervios que podemos observar en la imagen y aumentar el grosor de la pieza. Con ello se consiguio que quedara totalmente lisa la parte superior y que fuera válida para el propósito final de control.

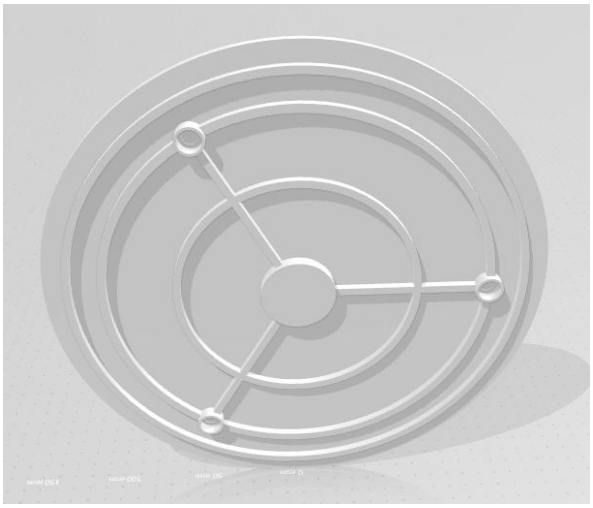


Figura 13. Imagen 3D pieza superior corregida.

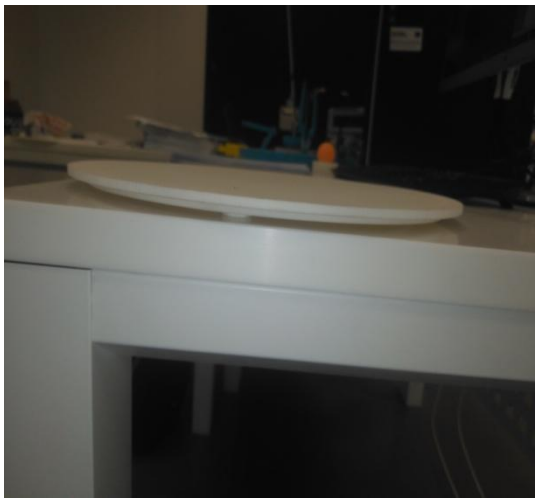


Figura 14. Pieza superior corregida.

Para realizar la impresión 3D, utilizamos la impresora BQ Prusa i3 Hephestos y el software Ultimaker Cura en el cual se pueden modificar diferentes parámetros para poder conseguir la mejor impresión para nuestra pieza. Con este software podemos modificar la velocidad de impresión, temperatura, tipo de base sobre la cual se agarra la pieza y otros parámetros para realizar la impresión adecuada de cada pieza.

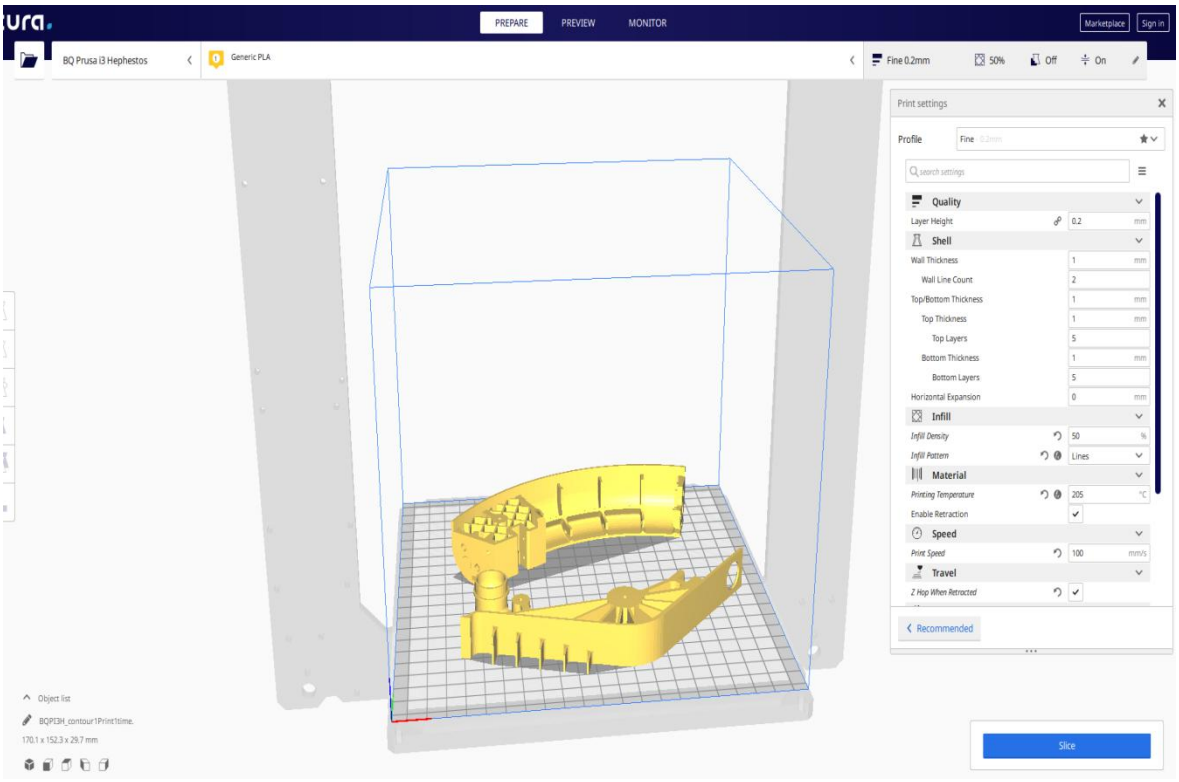


Figura 15. Imagen software de impresión 3d Ultimaker Cura.

7. ANÁLISIS CINEMÁTICO

En este apartado se realiza el análisis cinemático de la plataforma para posteriormente compararlo con el modelo cinemático implementado en el software de control. Este análisis es fundamental para asegurar la funcionalidad de la plataforma, pues es la base del movimiento de la misma para lograr estabilizar la pelota.

En primer lugar realizamos la búsqueda de la relación entre los ángulos de giro de la plataforma superior θ_x y θ_y y el desplazamiento de la misma sobre el eje x y el eje y.

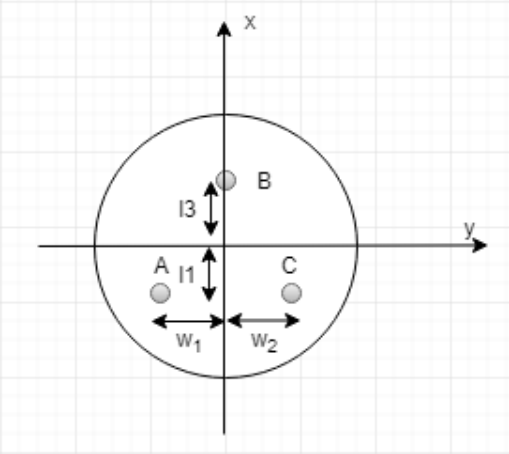


Figura 16. Plataforma superior y posición de motores

El primer paso es calcular la relación entre el desplazamiento vertical de cada línea de actuación y los giros de la plataforma:

- La relación del ángulo de giro sobre el eje x, θ_x, con el desplazamiento de los tres actuadores $(d_i, i = A, B, C)$.
  Se sabe que la variación del desplazamiento cuando gira sobre el eje x, es la siguiente:

$$\Delta d = d_A - d_C$$

  En este caso no aparece d_B debido a que está colocado sobre el eje

  Una vez que conocemos variación del desplazamiento se sabe que esta es igual a la suma de la distancia de los puntos de conexión de las articulaciones A y C con el centro de la plataforma por la tangente del ángulo de giro sobre el eje x, θ_x.

$$\Delta d = d_A - d_C = (w_1 + w_2) * \tan\theta_x$$

$$\theta_x = \tan^{-1}\left(\frac{d_A - d_C}{w_1 + w_2}\right)$$

Autor: **Raúl López Martín**
[424.19.38]

La relación entre el ángulo de giro sobre el eje y, θ_y, y el desplazamiento vertical de los tres actuadores. En este caso, se sabe que la variación del desplazamiento cuando giramos sobre el eje y, es la siguiente:

$$\Delta d = d_B - \frac{d_A + d_C}{2}$$

Ahora conociendo la variación del desplazamiento, podemos establecer la relación con el ángulo de giro, igualando esta variación con el producto entre la distancia de los puntos de conexión de las articulaciones A y C que son paralelas y la distancia de la articulación B con el centro de la plataforma y la tangente del ángulo de giro sobre el eje y, θ_y.

$$\Delta d = d_B - \frac{d_A + d_C}{2} = (l_1 + l_3) * \tan\theta_y$$

$$\theta_y = \tan^{-1}\left(\frac{d_B - \frac{d_A + d_C}{2}}{l_1 + l_3}\right)$$

La relación del desplazamiento en z viene dada por la forma geométrica de la parte superior de la plataforma es la siguiente (aunque no se utiliza para la estabilización de la bola sobre la plataforma):

$$z = \frac{1}{3}(d_A + d_B + d_C)$$

El siguiente paso es la obtención de las ecuaciones que definen el desplazamiento, d_A, d_B y d_C.

$$T = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{w_1 + w_2} & 0 & -\frac{1}{w_1 + w_2} \\ -\frac{1}{2 * (l_1 + l_3)} & \frac{1}{l_1 + l_3} & -\frac{1}{2 * (l_1 + l_3)} \end{pmatrix}$$

Ahora se expresan las relaciones anteriores en forma matricial para poder despejar d_A, d_B y d_C y obtener así las ecuaciones que los controlan. Para mayor comodidad en el cálculo y no encontrar el arco tangente de los ángulos de giro de la plataforma, se usa la tangente de estos mismos.

$$\begin{pmatrix} z \\ \tan\theta_x \\ \tan\theta_y \end{pmatrix} = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{w_1+w_2} & 0 & -\frac{1}{w_1+w_2} \\ -\frac{1}{2*(l_1+l_3)} & \frac{1}{l_1+l_3} & -\frac{1}{2*(l_1+l_3)} \end{pmatrix} * \begin{pmatrix} d_A \\ d_B \\ d_C \end{pmatrix}$$

Para despejar d_A, d_B y d_C primero se calcula la inversa de la matriz de transformación.

$$T^{-1} = \begin{pmatrix} 1 & \frac{w_1+w_2}{2} & -\frac{l_1+l_3}{3} \\ 1 & 0 & \frac{2*(l_1+l_3)}{3} \\ 1 & -\frac{w_1+w_2}{2} & -\frac{l_1+l_3}{3} \end{pmatrix}$$

El siguiente paso será multiplicar por la izquierda en ambos términos de la ecuación por la matriz T^{-1}.

$$\begin{pmatrix} 1 & \frac{w_1+w_2}{2} & -\frac{l_1+l_3}{3} \\ 1 & 0 & \frac{2*(l_1+l_3)}{3} \\ 1 & -\frac{w_1+w_2}{2} & -\frac{l_1+l_3}{3} \end{pmatrix} * \begin{pmatrix} z \\ \tan\theta_x \\ \tan\theta_y \end{pmatrix} = \begin{pmatrix} d_A \\ d_B \\ d_C \end{pmatrix}$$

Y con esto el resultado queda:

$$\begin{pmatrix} z + \frac{(w_1+w_2)*\tan\theta_x}{2} - \frac{(l_1+l_3)*\tan\theta_y}{3} \\ z + \frac{2*(l_1+l_3)*\tan\theta_y}{3} \\ z - \frac{(w_1+w_2)*\tan\theta_x}{2} - \frac{(l_1+l_3)*\tan\theta_y}{3} \end{pmatrix} = \begin{pmatrix} d_A \\ d_B \\ d_C \end{pmatrix}$$

$$d_A = z + \frac{(w_1+w_2)*\tan\theta_x}{2} - \frac{(l_1+l_3)*\tan\theta_y}{3}$$

$$d_B = z + \frac{2*(l_1+l_3)*\tan\theta_y}{3}$$

$$d_C = z - \frac{(w_1+w_2)*\tan\theta_x}{2} - \frac{(l_1+l_3)*\tan\theta_y}{3}$$

Una vez se conoce la relación entre el desplazamiento vertical de los actuadores y los giros de la plataforma, se pasa a calcular la relación entre el desplazamiento del punto donde está conectada la articulación con la plataforma, P_2 y el ángulo de giro de los motores θ_A, θ_B y θ_C.

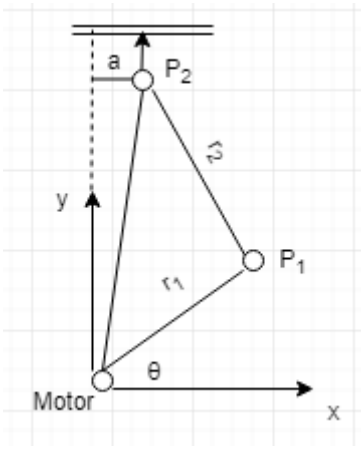


Figura 17. Cadena cinemática plataforma

Para poder definir esta relación, primero estableceremos las coordenadas del punto P_1que son las siguientes:

$$P_1 = \begin{pmatrix} r_1 * \cos\theta \\ r_1 * \sin\theta \end{pmatrix}$$

Posteriormente definimos el punto P_2:

$$P_2 = \begin{pmatrix} a \\ y_2 \end{pmatrix}$$

Para poder calcular el desplazamiento en el punto deseado despejaremos la coordenada y_2, sabiendo que el modulo de la resta de los puntos $|P_2 - P_1|$ es igual al radio de la articulación superior r_2.

$$|P_2 - P_1| = r_2 \rightarrow \sqrt{(r_1 * \cos\theta - a)^2 + (r_1 * \sin\theta - y_2)^2} = r_2$$

Se obtienen dos soluciones que difieren en la posición de la articulación si la articulación está colocada hacia dentro o hacia afuera, para el supuesto estudiado se elegirá la primera de las soluciones.

$$y_2 = \sqrt{-a^2 + 2 * a * r_1 * \cos\theta + r_1^2 * (\sin\theta)^2 - r_1^2 + r_2^2} + r_1 * \sin\theta$$

$$y_2 = r_1 * \sin\theta - \sqrt{-a^2 + 2 * a * r_1 * \cos\theta + r_1^2 * (\sin\theta)^2 - r_1^2 + r_2^2}$$

Para obtener el desplazamiento d_j debemos restar la posición en la que queda la plataforma, coordenada y_2 menos la posición inicial $y_2(\theta = 0)$.

$$d = y_2 - y_2(\theta = 0)$$

$$y_2(\theta = 0) = \sqrt{-a^2 + 2 * a * r_1 - r_1^2 + r_2^2}$$

$$d = y_2 - y_2(\theta = 0) = \sqrt{-a^2 + 2 * a * r_1 * \cos\theta + r_1^2 * (\sin\theta)^2 - r_1^2 + r_2^2} + r_1 * \sin\theta - \sqrt{-a^2 + 2 * a * r_1 - r_1^2 + r_2^2}$$

De esta ecuación se obtienen θ_A, θ_B, y θ_C.

Los cálculos anteriores se comparan con el modelo cinemático original y con un análisis cinemático realizado con el software de simulación Recurdyn®. Recurdyn® es un software para simular mecanismos y analizar su comportamiento dinámico. En el proyecto, el software se ha utilizado para simular el movimiento de la plataforma en función de los ángulos de giro de los motores. El resultado ha servido para validar sendos modelos teóricos: el descrito arriba y el original proporcionado con el prediseño.

Lo primero que hicimos en Recurdyn® es introducir el modelo de la plataforma, previamente ensamblado en inventor. Una vez se dispone el modelo, el primer paso es unir los cuerpos rígidos de la plataforma para que el manejo del programa sea más simple y para dejar aisladas las uniones de los elementos móviles de la plataforma.

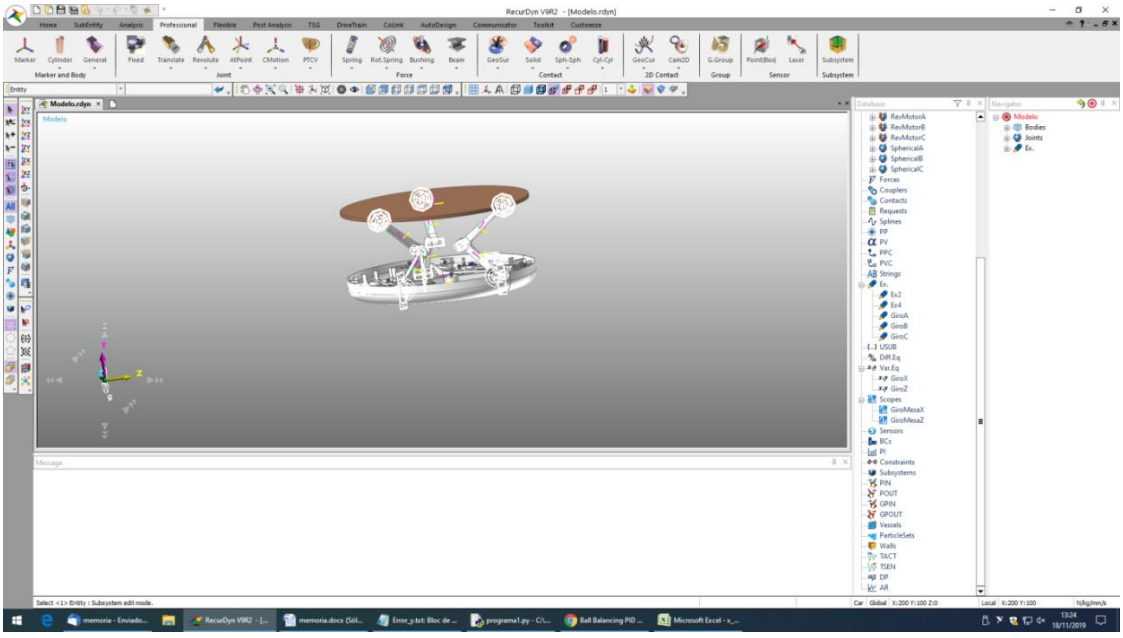


Figura 18. Imagen modelo en RecurDyn®.

Autor: **Raúl López Martín**
[424.19.38]

El siguiente paso es definir las uniones correctamente para que el movimiento de cada articulación sea el correcto. Al hacerlo hay que orientar el movimiento de las mismas con los marcadores (*markers*) que son los ejes de coordenadas de cada elemento de la plataforma.

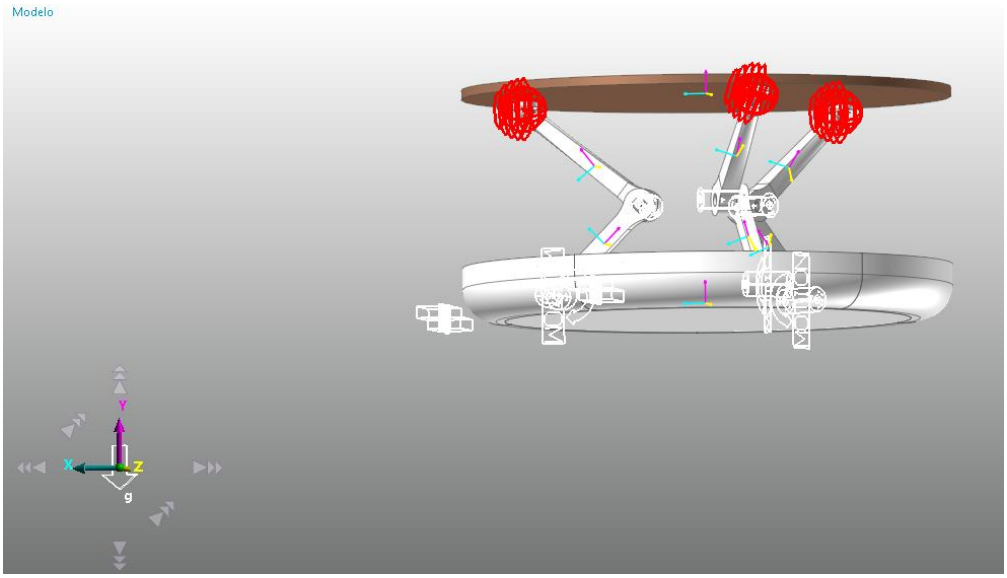


Figura 19. Imagen unión superior esférica.

La imagen 19 muestra las uniones esféricas que están conectadas con la plataforma superior.

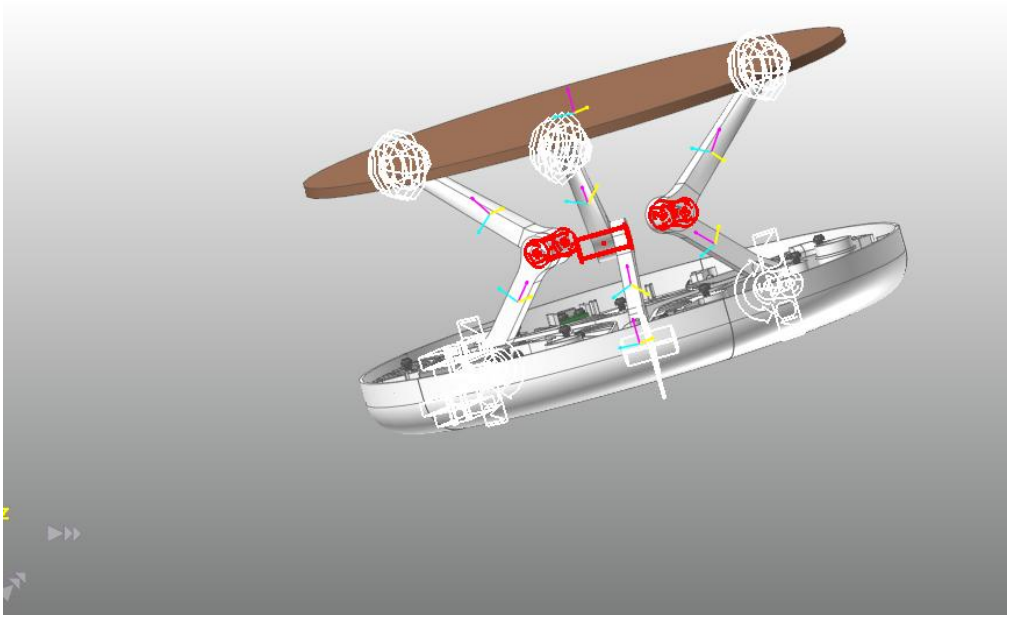


Figura 20. Muestra las uniones móviles con los *markers* de los cuerpos de la plataforma, o de alguno creado para dicho propósito.

Y por último se coloca mediante la instrucción *motion* el movimiento en cada uno de los motores para posteriormente poder introducir los ángulos deseados en la comprobación. Para ello se generan 3 expresiones haciendo la transformación de grados a radianes y se generan 2 variables, GiroX y GiroY donde se monitorizaran los ángulos de giro de la plataforma. Los resultados se representan en una grafica.

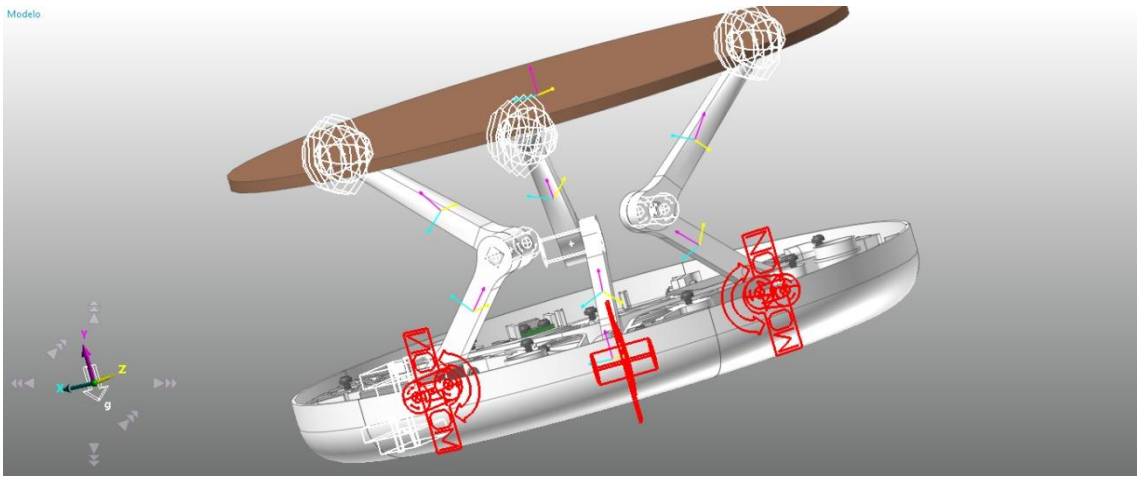


Figura 21. Es esta imagen se muestra donde se coloca la instrucción *motion* para el movimiento de los motores.

Con objeto de comparar los resultados anteriores con el modelo original, es necesario cambiar la referencia α y β en nuestra θ_x y θ_y. α y β es la representación utilizada dentro del código y su definición aparece en la imagen inferior. α es el ángulo de giro y β el ángulo para posicionar en 2D el eje de giro.

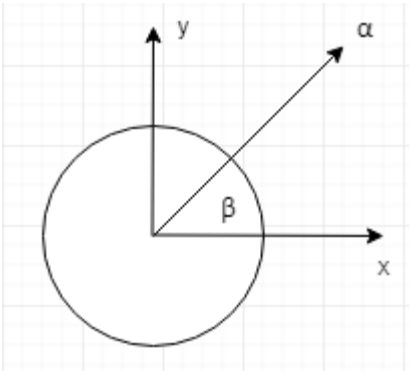


Figura 22. Representación de los ángulos de giro.

Autor: **Raúl López Martín**
[424.19.38]

$$\theta_x = \alpha * \cos\beta$$

$$\theta_y = \alpha * \sin\beta$$

En primer lugar se comparará la relación entre θ_x , θ_y y las tres aproximaciones para θ_A:

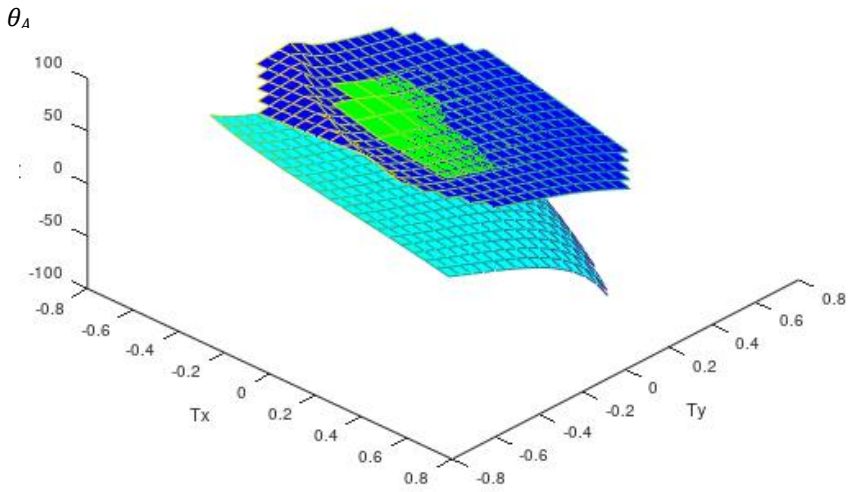


Figura 23. Grafico de comparación de las θ_A.

Se pueden observar dos planos pegados, el azul oscuro y el verde que se corresponden con su modelo y la simulación en Recurdyn® respectivamente, y el tercer plano, de color azul claro, que está relacionado con el modelo anterior. Se puede ver claramente que tienen las mismas tendencias y la única diferencia es que el plano está separado debido a que la posición inicial tomada es diferente de las de los otros dos modelos.

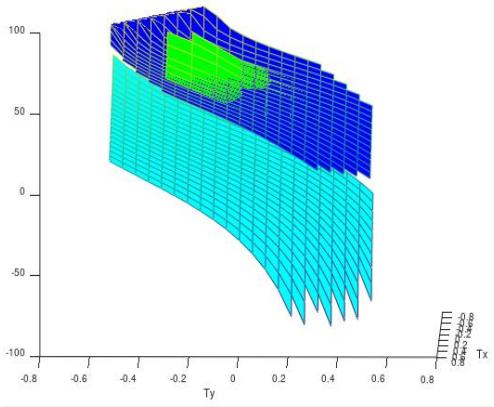


Figura 24. Grafico de comparación de las θ_A.

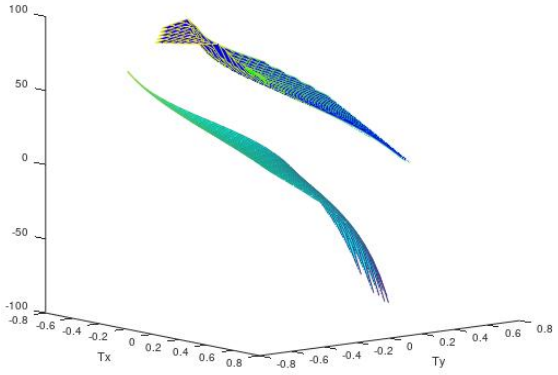


Figura 25. Grafico de comparación de las θ_A.

Las figuras 24 y 25 muestran en detalle el modelo anterior.

Ahora se pasa a realizar la comparación entre las tres aproximaciones para θ_B:

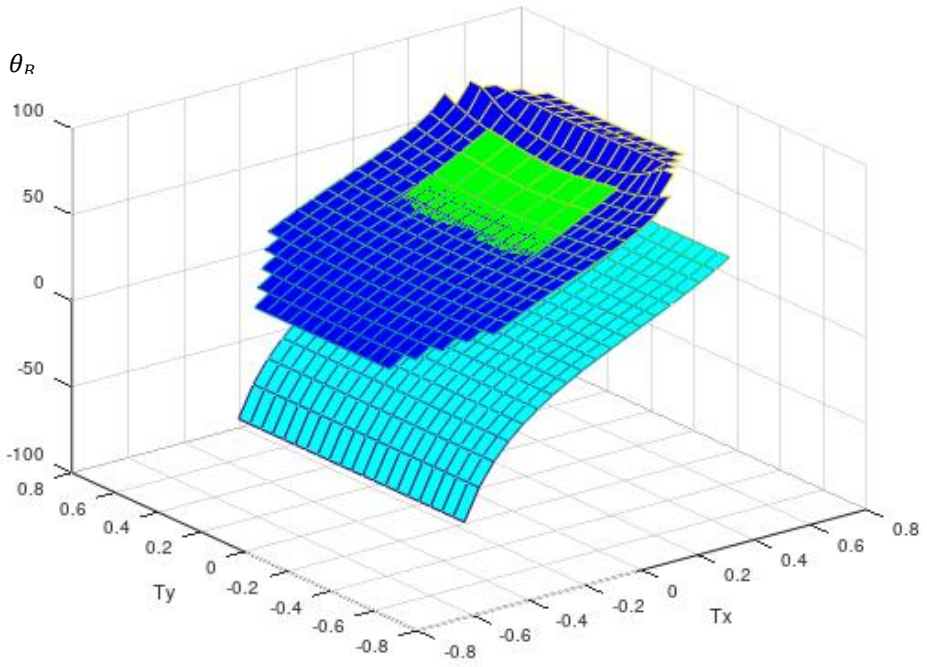


Figura 26. Grafico de comparación de las θ_B.

Se observa la misma situación que anteriormente, los planos tienen las mismas tendencias, dos coincidentes correspondientes al modelo original y a Recurdyn® y el modelo planteado separado por ese offset debido a la posición inicial escogida.

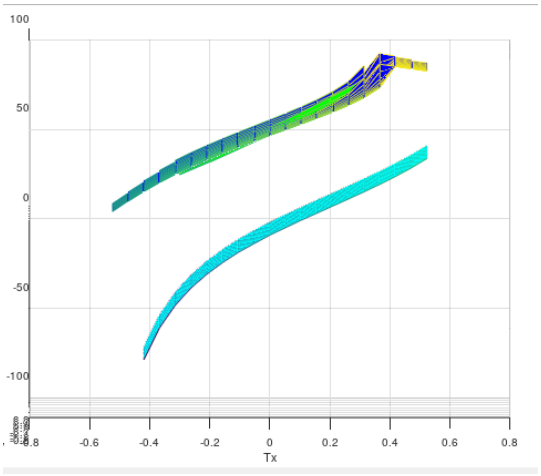


Figura 27. Grafico de comparación de las θ_B.

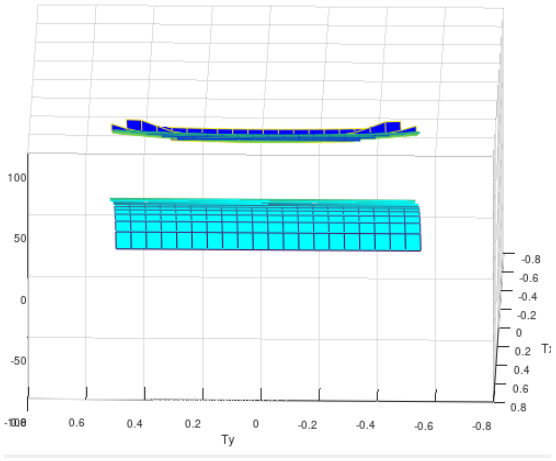


Figura 28. Grafico de comparación de las θ_B.

Aquí se ve como la dirección en θ_x y θ_y coincide en los planos de los 3 modelos.

Y por último se comparará las tres aproximaciones para θ_C

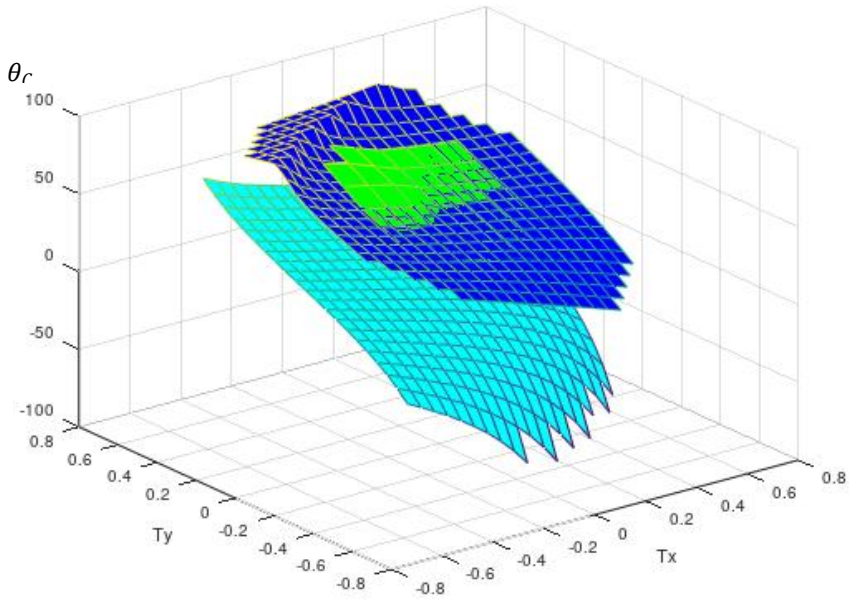


Figura 29. Grafico de comparación de las θ_C.

Y de nuevo vuelve a suceder la misma situación, coincidiendo las tendencias y difiriendo en la distancia tomada como posición inicial.

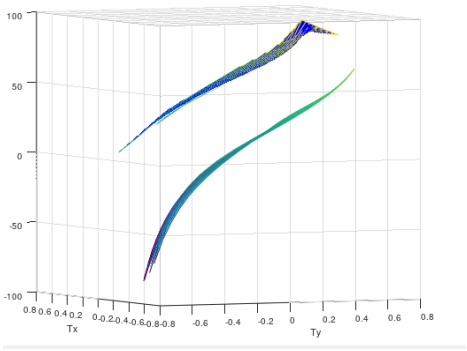


Figura 30. Grafico de comparación de las θ_C.

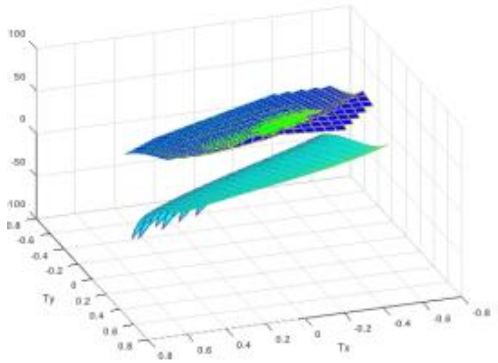


Figura 31. Grafico de comparación de las θ_C.

En estas imágenes se puede observar como las direcciones en θ_x y θ_y coinciden en los tres planos.

Como se ha observado en los gráficos anteriores, el modelo original y el de Recurdyn® son coincidentes y prácticamente paralelos al modelo anteriormente planteado. Estas diferencias son debidas a que el punto de equilibrio es diferente en ambos casos.

Una vez realizada esta comparación se puede decir que ambos modelos son correctos, ya que han sido verificados por Recurdyn® y se tendrá que limitar el ángulo α a menos de 35 grados.

8. ANÁLISIS DE SOFTWARE

8.1. FLUJO DEL PROGRAMA.

Para la descripción de la arquitectura software se han utilizado diagramas UML que, por su capacidad descriptiva, permiten fácilmente mostrar el flujo de información y el detalle de las variables y parámetros del sistema. A pesar de no haberse realizado una codificación en objetos, se ha hecho una aproximación basada en las unidades software existentes: control, cámara motor e interface.

La imagen inferior muestra un diagrama con el funcionamiento general del programa.

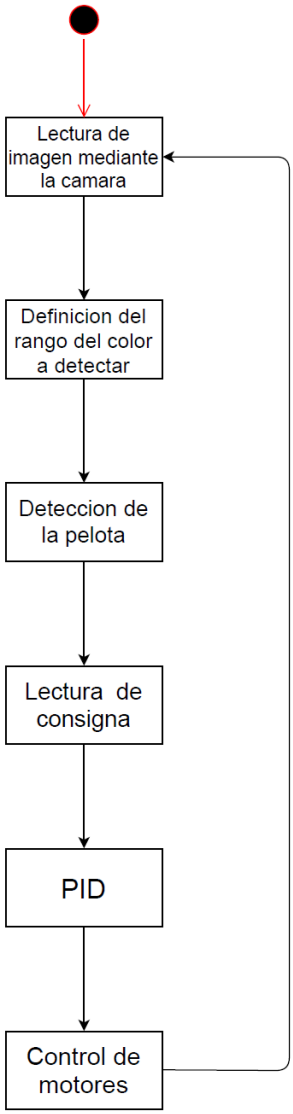


Figura 32. Diagrama general

En el siguiente diagrama, se puede ver la inicialización del programa.

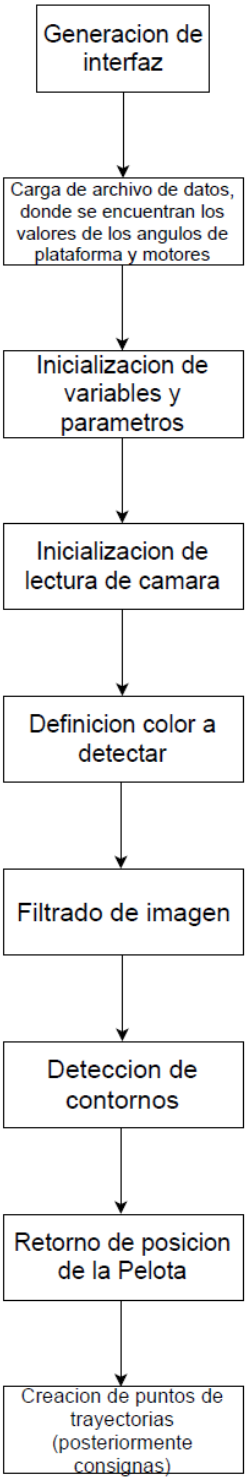


Figura 33. Diagrama de inicialización

A continuación se observa un diagrama de clases en el cual aparecen los objetos, sus funciones y las variables que afectan a cada uno de ellos.

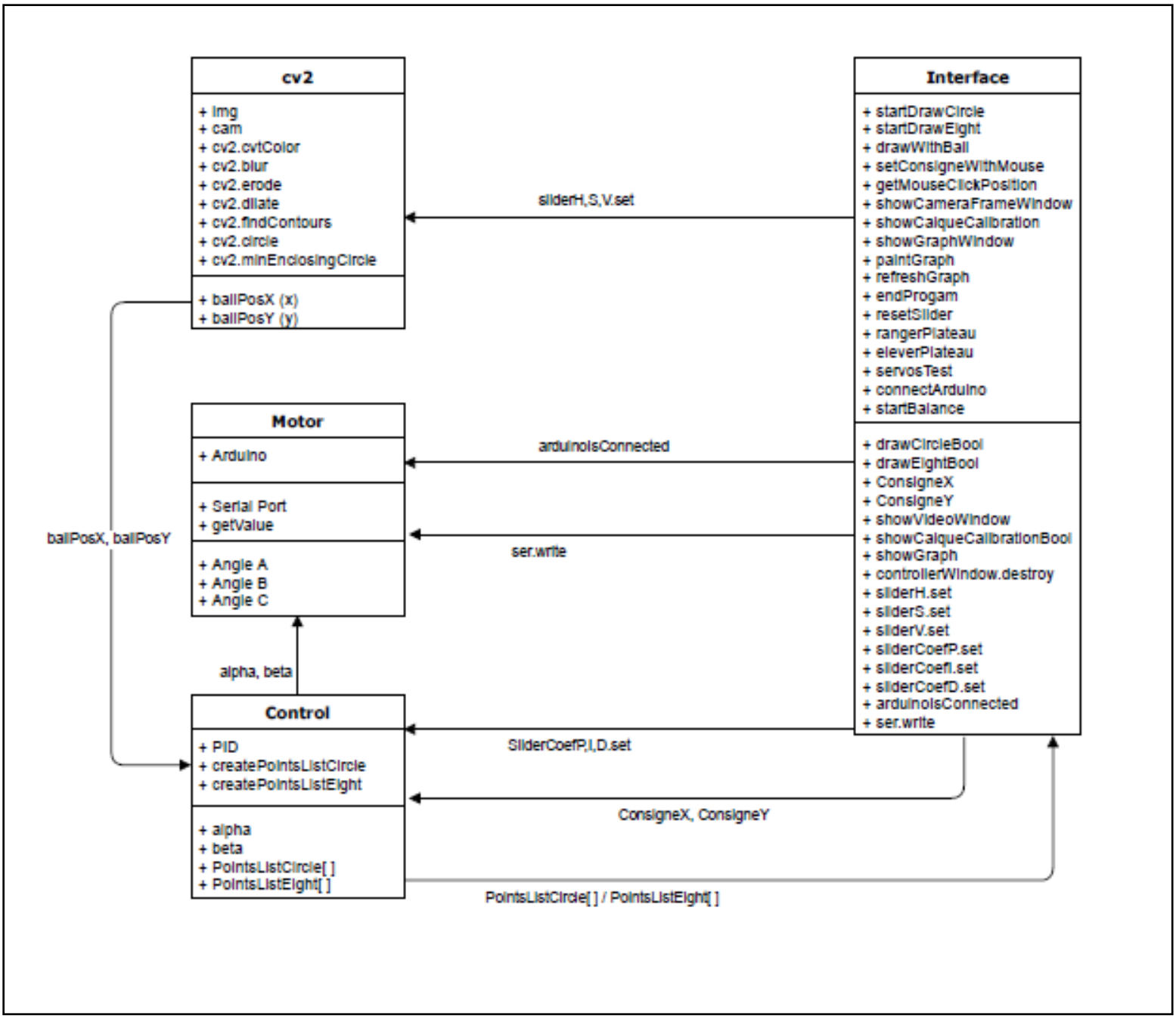


Figura 35. Diagrama de clases

En este diagrama se pueden observar las diferentes clases en las que se han clasificado las partes del programa:

- Cv2 es la parte de procesamiento de imágenes y detección de objetos, en esta clase se encuentran las funciones que determinan el color, filtran la imagen y detectan la posición de la pelota. Como se puede observar sólo se les pasa el valor de los coeficientes que definen el color y nos devuelve la posición de la pelota, las coordenadas x e y, las cuales serán utilizadas en el control.
- En la clase interface se encuentran las funciones que permiten activar diferentes partes del programa, como la conexión con el Arduino®, algunas órdenes para controlar la plataforma, la consigna y la configuración de los valores del PID.
- Otra de las clases que se pueden observar, es la clase control. El elemento principal que compone dicha clase es el PID, en el se introduce la consigna, la posición de la pelota, los coeficientes k_p, k_d y k_i y la variable que devuelve son los ángulos de control de la plataforma.
- Y por último se encuentra la clase motor, sobre la cual trabaja Arduino®, y que recibe las variables de conexión y escritura en el puerto serie de Arduino® y este comanda los motores para realizar el movimiento oportuno.

Y por último, la imagen inferior muestra el diagrama de dispositivos:

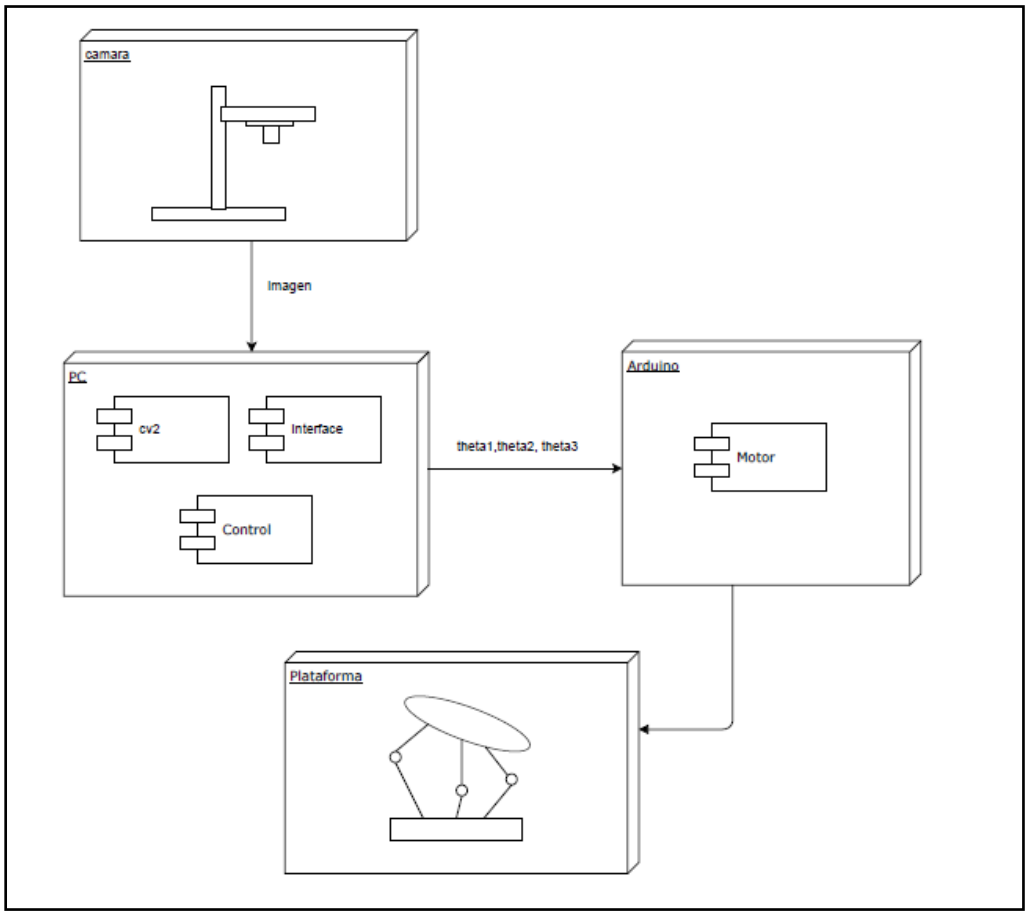


Figura 36. Diagrama de dispositivos.

8.2. EL CONTROLADOR

A continuación se presentan las opciones para el ajuste del PID. Para ello, partimos del modelo de referencia que aparece en la siguiente imagen:

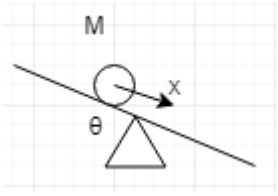


Figura 37. Modelo teórico de referencia

La representación simplificada del sistema anterior permite obtener la función de transferencia.

$$M * \ddot{x} = M * g * \theta$$

$$\frac{x}{\theta} = \frac{g}{s^2}$$

Se realiza a continuación el análisis en bucle abierto mediante el método del lugar de las raíces, con lo que la función que controla el PID quedara de la siguiente manera.

$$G = \frac{9.8 * K_p}{s^2} + \frac{9.8 * K_d}{s} + \frac{9.8 * K_i}{s^3}$$

A continuación se muestra cómo responden las diferentes combinaciones de parámetros:

- Aplicando la configuración P, únicamente la parte proporcional, se obtiene el siguiente resultado:

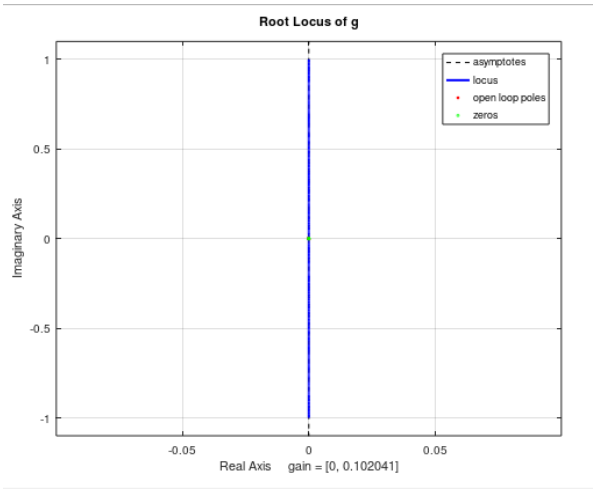


Figura 38. Resultado configuración P

Se observa que no tiene polos en la parte imaginaria por lo que el sistema se considera inestable.

- Aplicando la configuración DI (derivativo e integral), se obtiene el siguiente resultado.

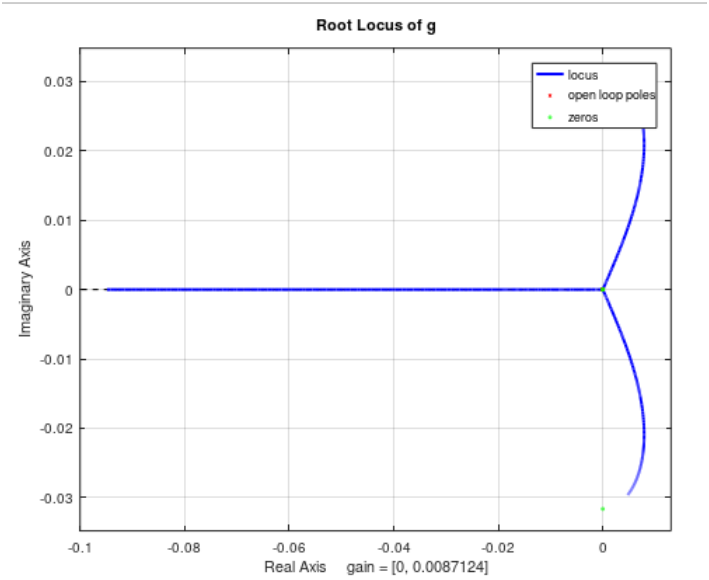


Figura 39. Resultado configuración DI

Observando la figura 38 podemos encontrar polos en la parte derecha, la parte positiva, por lo cual el sistema es de carácter inestable.

- El siguiente caso a analizar, es la configuración PD:

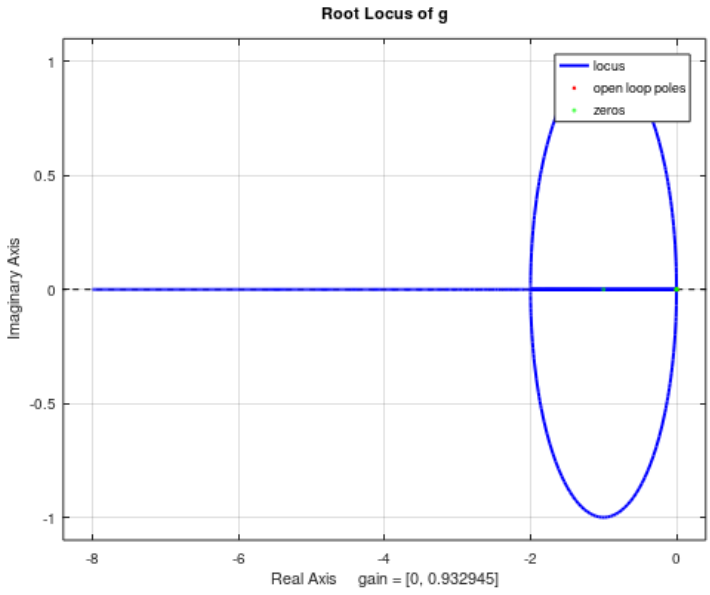


Figura 40. Resultado configuración PD

En el detalle siguiente, se observa que todos los polos están situados en la zona imaginaria, por lo que el sistema es estable.

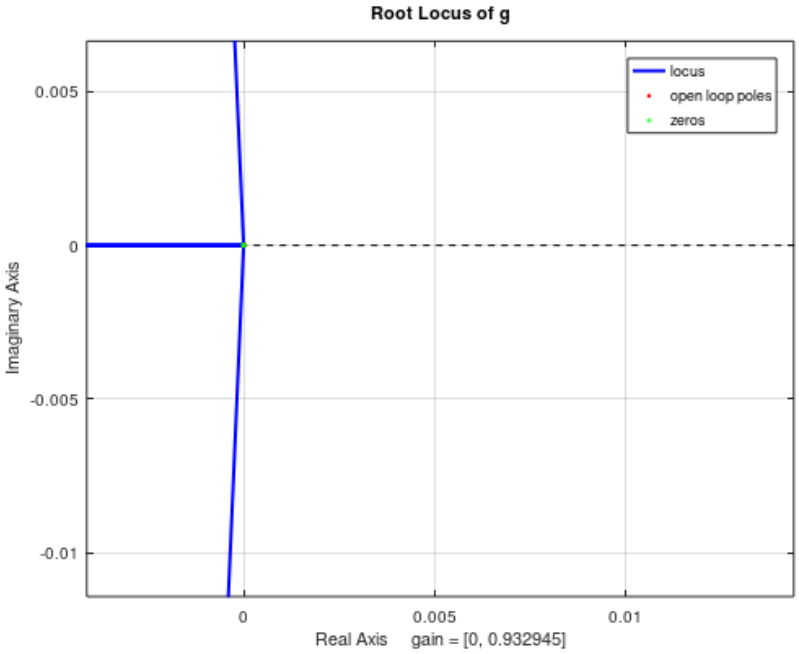


Figura 41. Resultado configuración PD

- En la configuración PI, se introducen en Octave los datos y se obtiene el siguiente resultado:

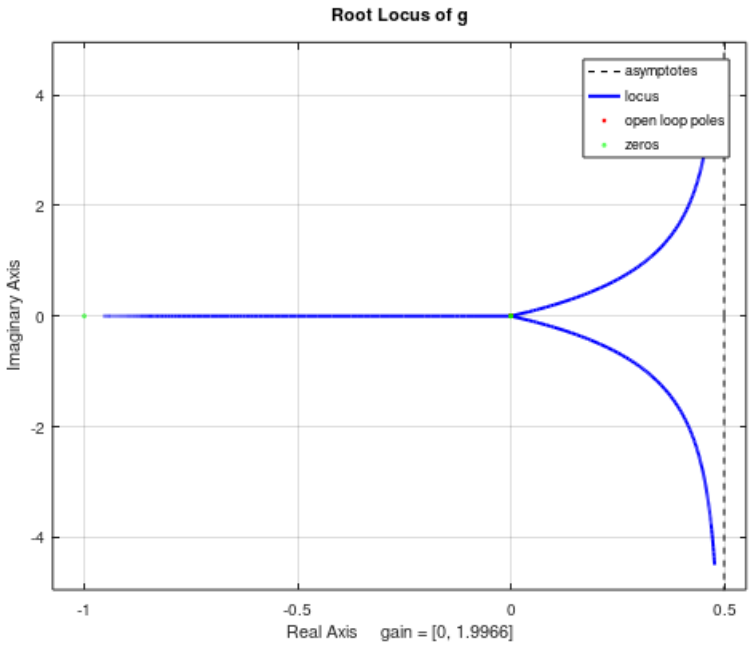


Figura 42. Resultado configuración PI

Viendo la imagen y viendo que hay polos en la parte positiva de la grafica se concluye que el sistema será inestable, por lo que no sería conveniente utilizar esta configuración.

- La siguiente configuración a analizar es un PID, con un valor muy bajo en la parte integral:

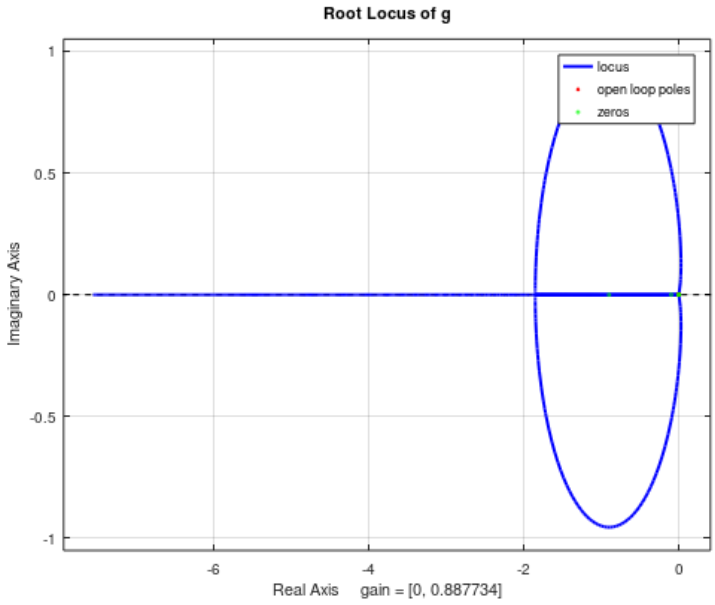


Figura 43. Resultado configuración PID

Parece que es estable pero si nos acercamos al origen de coordenadas, se observa que hay una pequeña zona de inestabilidad.

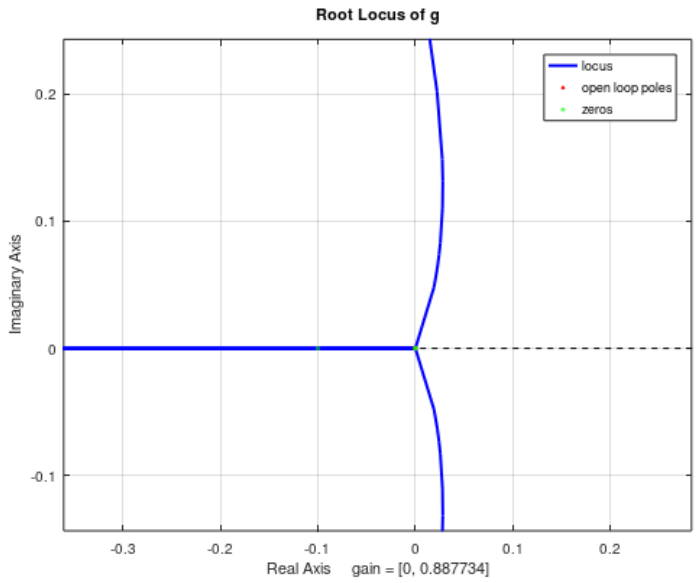


Figura 44. Resultado configuración PID

Esto puede corregirse elevando la parte proporcional y derivativa pero al aumentar mucho estas, el control del sistema se vuelve muy agresivo y es difícil de controlar la pelota.

- Por último se analiza una configuración con únicamente la parte derivativa.

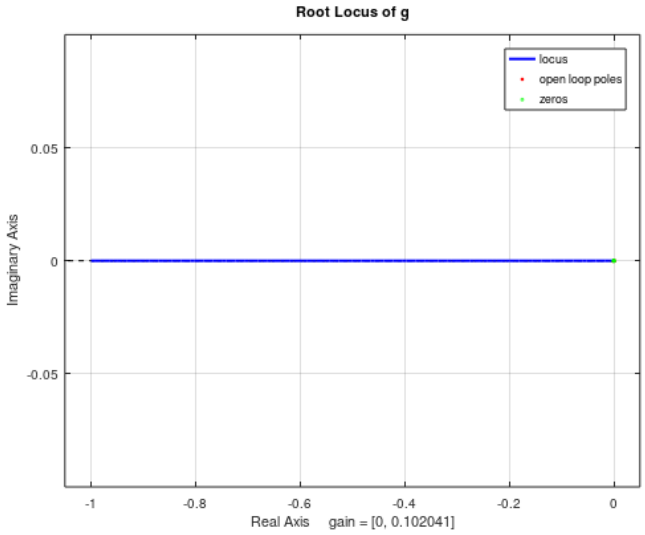


Figura 45. Resultado configuración D

Se puede ver que es estable pero se necesita ajustar más el control.

Como conclusión la configuración ideal teóricamente sería un PD, aunque debido a la sencillez del modelo en la práctica se necesita una componente pequeña de integración para que la pelota no se quede parada en zonas alejadas, sino que tienda a rodar hacia el punto de referencia.

9. ANÁLISIS ELECTRÓNICO

A continuación, se procede a la revisión de los planos de la PCB (Printed Circuit Board) que gobierna los motores que mueven la plataforma.

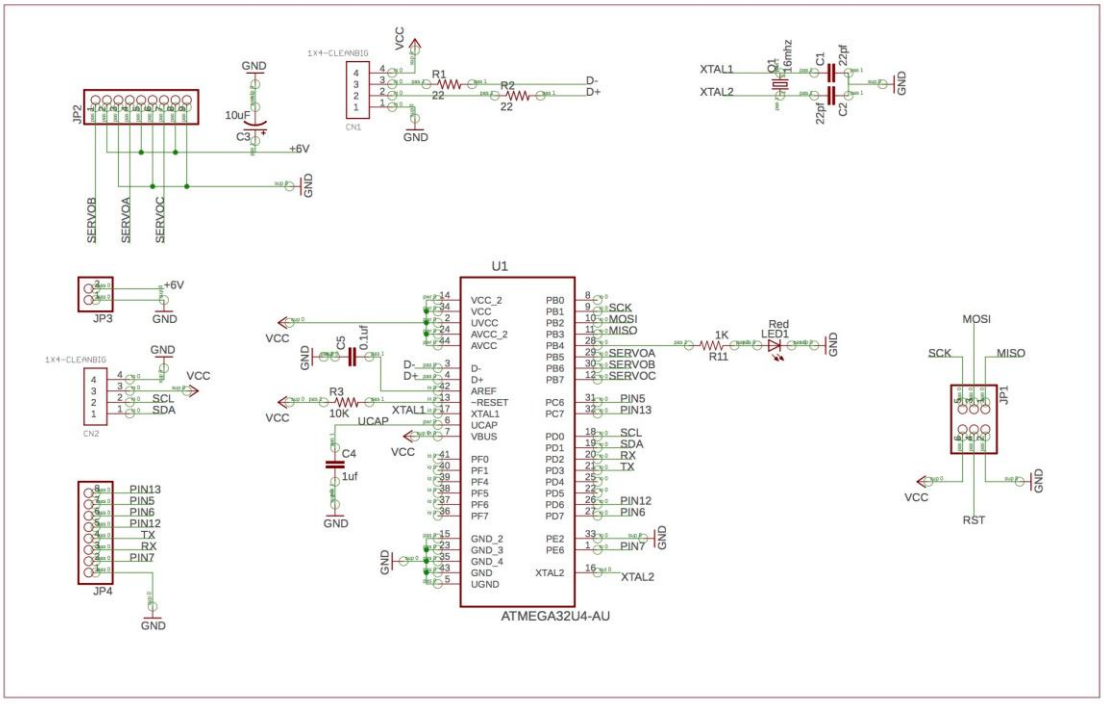


Figura46. Plano eléctrico PCB

La PCB se fabricó en un proveedor externo:

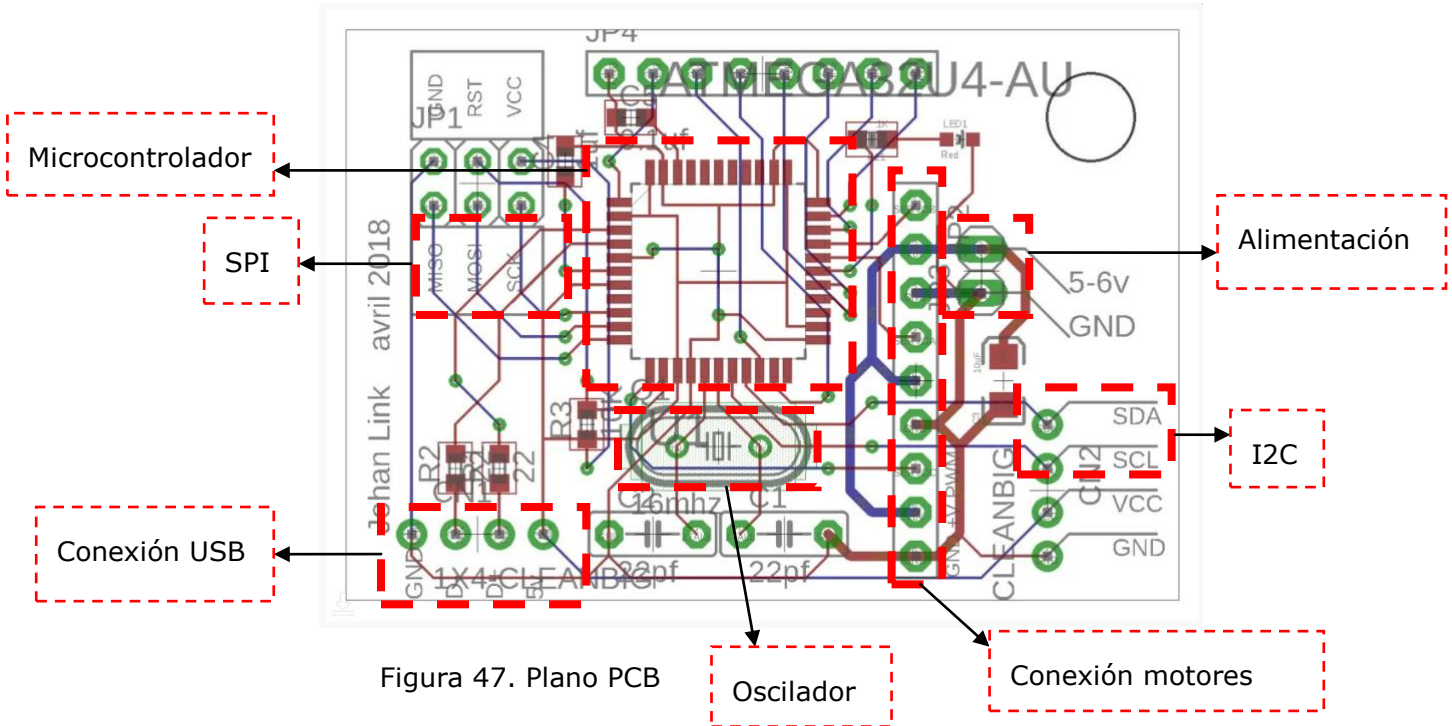


Figura 47. Plano PCB

Autor: **Raúl López Martín**
[424.19.38]

En el plano anterior se puede observarla distribución de los componentes de la PCB. La alimentación se encuentra en la parte derecha de la placa, en nuestro caso funciona con 5 voltios. Se encarga principalmente de alimentar los motores.

Otro de los componentes que se puede observar es un conector con nueve pines hembra, donde se conectaran los motores. Continuando con la descripción se puede encontrar el oscilador, el cual es necesario para el funcionamiento del reloj del microcontrolador.

En la parte central de la placa se encuentra el microcontrolador, el caso analizado es un ATMEGA32U4-AU, que es un microcontrolador CMOS de 8 bits de bajo consumo de potencia basado en la arquitectura AVR RISC.

En la parte inferior izquierda se encuentra la conexión USB que es la encargada de conectar la placa con el PC, esta se conforma de 4 pines, dos para transmitir la información, denominados D+ y D- y otros dos que son la masa y la alimentación.

Y por último tenemos los buses de datos SPI e I2S que permiten la comunicación entre el microcontrolador y otros dispositivos.

La placa superior es la que controla los servomotores. El control se realiza mediante un programa de Arduino que convierte la señal del ángulo en el motor, en consignas de voltaje:

$$posicion\ servo\ A = ((-2380\ +\ 1450) * (anguloA)\ /\ 90\ +\ 2380);$$

$$posicion\ servo\ B = ((-2330\ +\ 1480) * (anguloB)\ /\ 90\ +\ 2330);$$

$$posicion\ servo\ C = ((-2270\ +\ 1430) * (anguloC)\ /\ 90\ +\ 2270);$$

Para trabajar en Arduino, primero hay que quemar el gestor de arranque (*bootloader*), mediante el cual se configurará el microcontrolador.

Para esta plataforma se han elegido como actuadores, 3 servomotores Futaba S3003 para aplicaciones de robóticas. Estos motores, tienen un rango de giro de 180º lo cual es suficiente para nuestra aplicación, y son controlados mediante una señal PWM proporcionada por el microcontrolador. La anchura del pulso determina la posición del servo, si la anchura es de 2.3ms el servo se situara en un extremo y si es de 0.3ms se situara en el otro, con cualquier otra anchura comprendida entre estas dos controlaremos todas las demás posiciones. Ver figura (49).

Figura 48. Motor Futaba S3003.

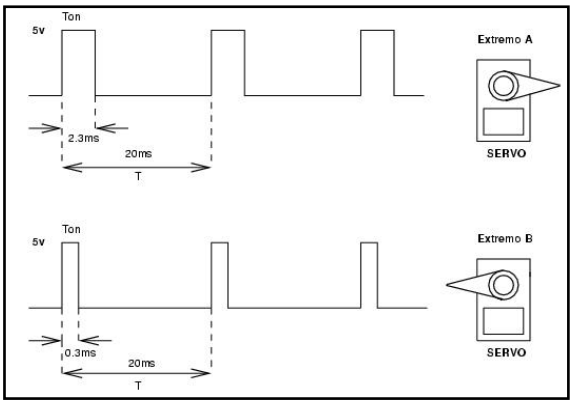


Figura 49. Control motor.

9.1. ANÁLISIS DEL PAR MOTOR

Como parte del proceso de análisis, se comprueba la capacidad del par motor para mover la plataforma superior. Para ello, se utiliza la representación simplificada inferior:

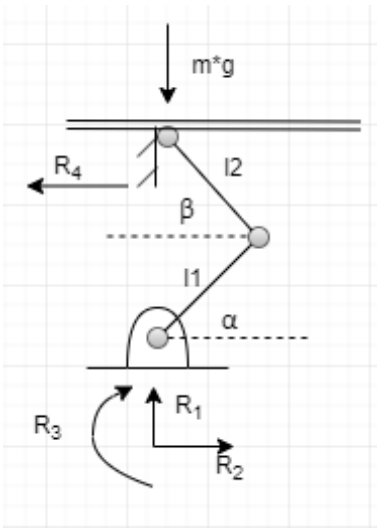


Figura 50. Cadena cinemática plataforma.

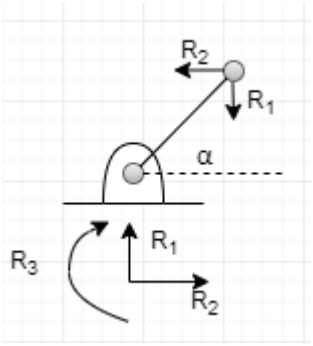
Las ecuaciones de equilibrio en fuerza y momento respecto al eje intermedio son:

$$\sum Fx = 0 \rightarrow R_2 - R_4 = 0$$

$$\sum Fy = 0 \rightarrow R_1 - m * g = 0$$

$$\sum Mz = 0 \rightarrow R_3 - R_1 * \cos \propto * l1 - R_2 * \sin \propto * l1 + m * g * \cos \beta * l2 - R_4 * \sin \beta * l2 = 0$$

Para resolver cuánta masa pueden mover los motores, se necesita una ecuación más. Para ello se pasan los esfuerzos desde la base a la posición de la primera rotula.



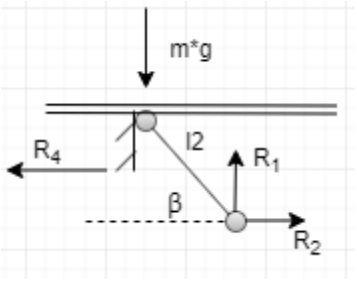


Figura 51. Cadena cinemática inferior Figura 52. Cadena cinemática superior

$$E_r = R_1 * \cos\beta * l2 - R_2 * \sin\beta * l2 = 0$$

El resultado del sistema de ecuaciones es:

$$R_1 = m * g$$

$$R_2 = R_4$$

$$R_2 = \frac{m * g}{\tan\beta}$$

$$R_3 = m * g * \cos\alpha * l1 + \frac{m * g}{\tan\beta} * \sin\alpha * l1 - m * g * \cos\beta * l2 + \frac{m * g}{\tan\beta} * \sin\beta * l2$$

9.1.1. Calculo masa máxima

Como se aprecia en la ecuación anterior, para calcular la masa se necesita conocer los valores de α y β. Como β depende de la posición de α, es necesario calcular primero la relación que hay entre ambas.

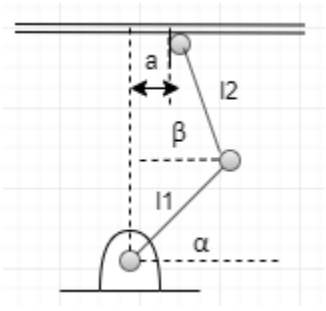


Figura 53. Cadena cinemática plataforma.

$$l_1 * \cos\alpha - l_2 * \cos\beta = a$$

$$\beta = \cos^{-1}\left(\frac{-a + l_1 * \cos\alpha}{l_2}\right)$$

Una vez conocida esta relación, se despeja la masa de la ecuación:

$$m = \frac{R_3}{\left(g * \cos\alpha * l1 + \frac{g}{\tan\beta} * \sin\alpha * l1 - g * \cos\beta * l2 + \frac{g}{\tan\beta} * \sin\beta * l2\right)}$$

Por último, se introducen en Octave los datos, ya que se conoce el torque que es el del motor, y se generará un vector con todos los posibles ángulos α y sus correspondientes β.

A continuación en la grafica inferior se observa el posible valor de masa que puede mover la plataforma para un α dado.

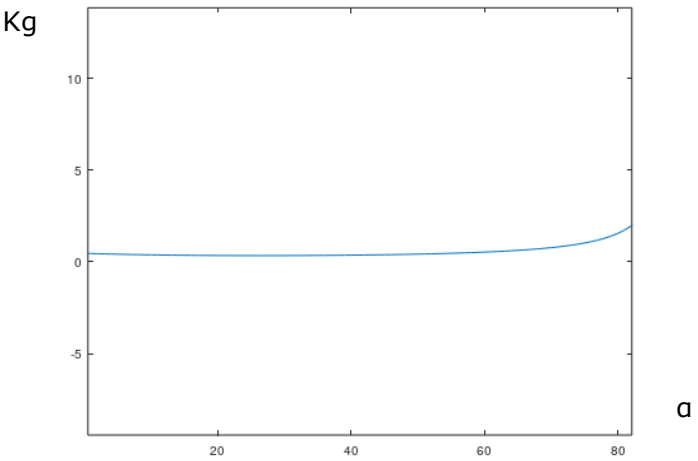


Figura 54. Grafica relación α y masa

En el grafico anterior se observa que es un valor pequeño la masa que puede mover cada motor. Ahora, en el siguiente grafico se verá al detalle el valor al que se aproxima.

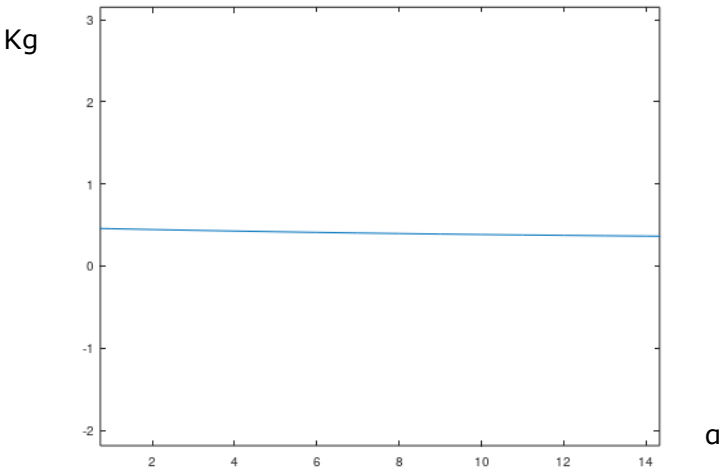


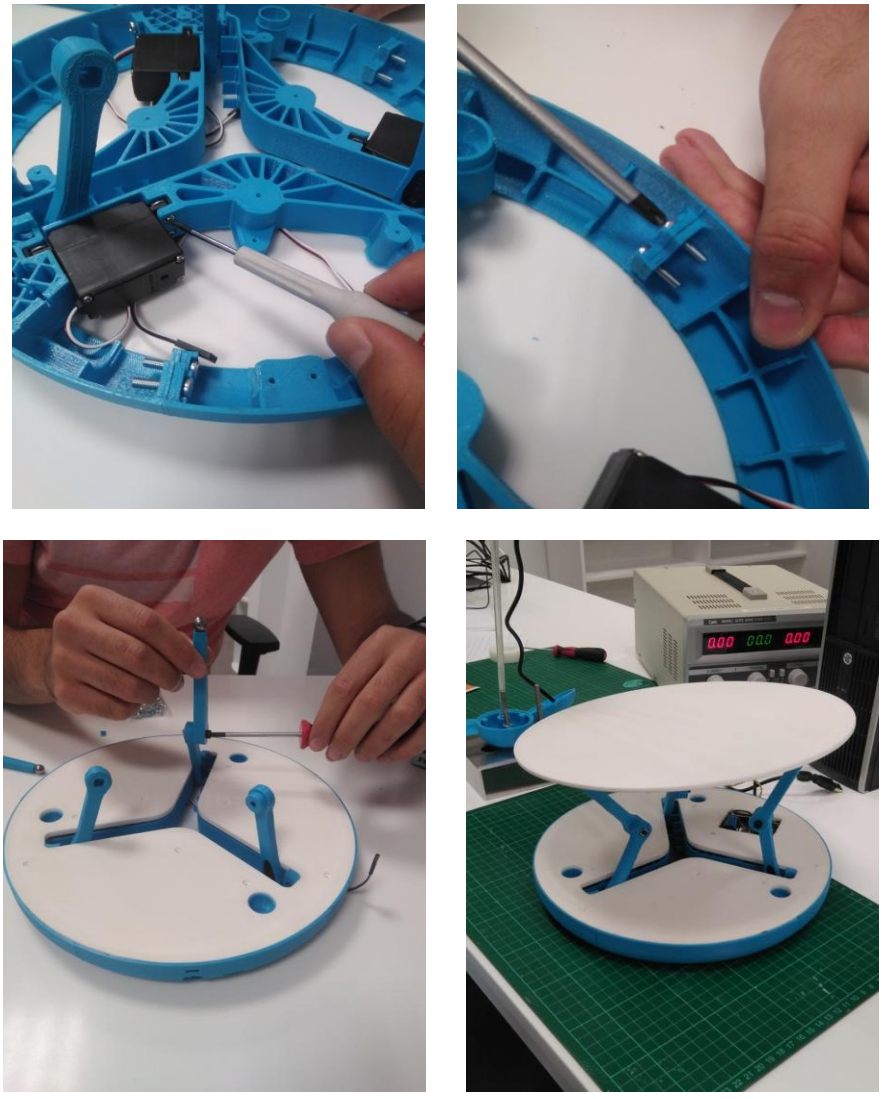
Figura 55. Grafica relación α y masa

Como se puede observar en el grafico superior, la masa que puede mover cada motor en estacionario esta cercana a 0,5 Kg. Esto hace que al tener tres motores, la masa máxima que puede mover la plataforma sean unos 1,5 kilos.

10. VALIDACIÓN

10.1. MONTAJE

Al finalizar la validación de forma satisfactoria, se procedió a realizar el montaje completo de la plataforma.



Figuras 56, 57, 58, 59. Montaje de plataforma.

Al realizar este montaje se observo que el soporte de la cámara no tenía la suficiente masa para mantener el equilibrio. En consecuencia, ha sido necesario modificar el soporte, tal y como se describe más adelante.

10.2. VERIFICACIÓN DE LA ELECTRÓNICA DE LOS MOTORES

El siguiente paso que se realizo fue la comprobación de los motores para asegurar su correcto funcionamiento, para ello se realizo un pequeño programa en Arduino® que su único funcionamiento era llevar los motores a una determinada posición cada segundo.

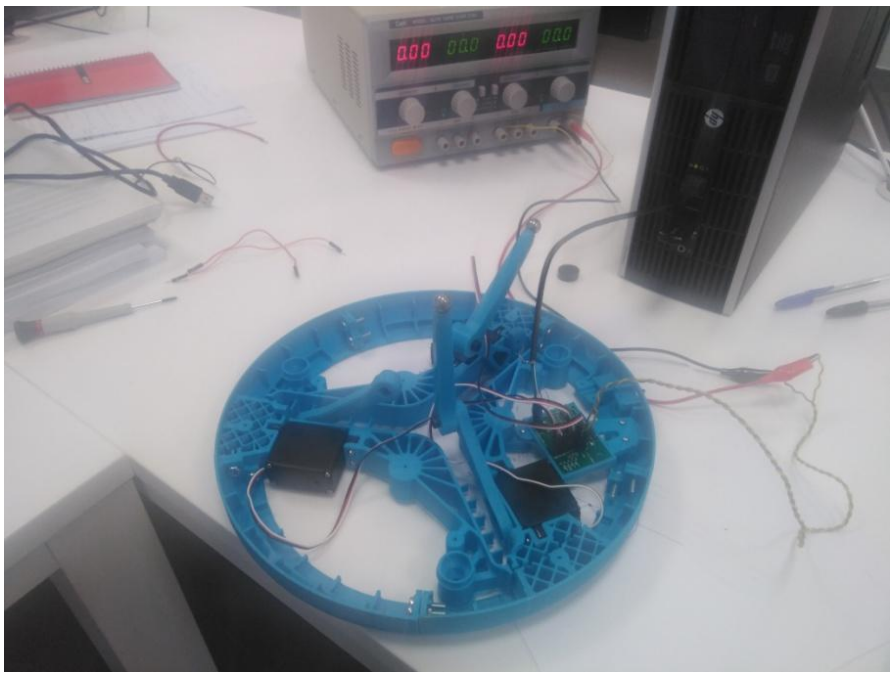


Figura 60. Verificación de motores.

10.3. VERIFICACIÓN DE LA CÁMARA

Una vez confirmado que los motores se movían de manera correcta, se comprobó la cámara mediante la cual se recogerá la imagen que nos dé la posición actual de la pelota y cierre el bucle de control. Esto se realizo descargando el propio software de la cámara y comprobando que aparecía imagen en la pantalla de nuestro ordenador.

10.4. VALIDACIÓN Y CORRECCIÓN DE LOS ELEMENTOS FÍSICOS DE LA PLATAFORMA

Una vez terminado el montaje de todos los elementos se observó que el soporte que sujetaba la cámara era demasiado endeble. La parte superior donde se sujetaba la cámara pesa mucho y el tubo y el soporte inferior que la aguantaban no eran capaces de sostener el montaje de manera estable.

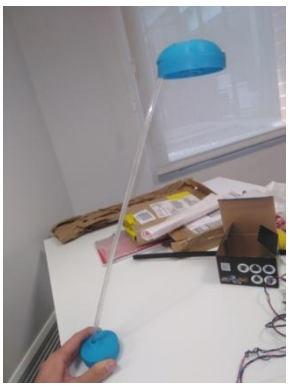


Figura 61. Soporte cámara antiguo.

Para resolver este problema se realizó un nuevo diseño para un soporte nuevo con dos perfiles Bosch en forma de L y una base metálica con suficiente masa como para evitar el vuelco de la estructura. Además el soporte y el perfil en los cuales se sitúa la cámara, están unidos al otro perfil mediante una escuadras, que permiten el movimiento de la cámara en el eje x y el eje z, así se logra un mejor ajuste de la posición de la cámara, lo cual es muy importante a la hora de realizar el control.

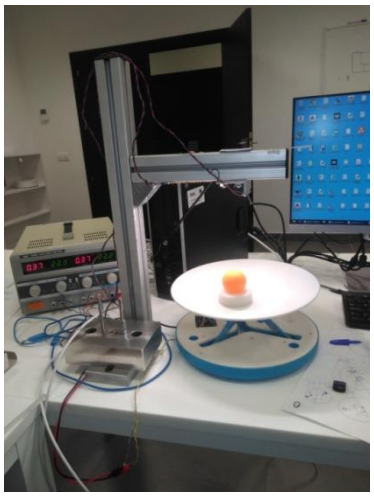


Figura 62. Soporte cámara nuevo.

Otros ajustes necesarios, fueron el pulido y engrasado de cada una de las zonas de la plataforma en las cuales hay movimiento, dado que provocan fenómenos de pegado y deslizamiento.

Por último se colocaron unas tiras LED en el perfil superior que sujeta la cámara para mejorar la visibilidad, eliminando algunas sombras que en determinados momentos dificultaban la detección de la pelota.

10.5. COMPILACIÓN Y RESOLUCIÓN DE PROBLEMAS Y DIFICULTADES EN EL SOFTWARE

10.5.1. Detección mediante visión artificial

El primer problema surgió cuando se realizó la comprobación de la parte de visión y detección de la pelota. En primer lugar se tuvo que modificar la parte en donde se definía el color a detectar, ya que no era capaz de identificar el tono de nuestra pelota. Para ello se modifico el código cambiando el límite superior e inferior que definía el color. Este era modificable mediante unos sliders en la interface de usuario, pero hacia demasiado complicado encontrar la configuración adecuada para nuestro caso y se decidió sustituirlo por unos límites fijos.

```
#lowerBound=np.array([H*(1-sliderH.get()/100),S*(1-sliderS.get()/100),V*(1-sliderV.get()/100)])
#upperBound=np.array([H*(1+sliderH.get()/100),S*(1+sliderS.get()/100),V*(1+sliderV.get()/100)])
```

```
lowerBound=np.array([10,100,20])
upperBound=np.array([25,255,255])
```

Corrección

El siguiente problema a corregir fue la detección de contornos. Este proceso era inestable cuando se realizó la primera prueba debido a que no detectaba o lo hacía de manera intermitente. Para solucionar este problema primero se añadió un argumento más a la función cv2.findContours ya que esta devuelve los contornos y la jerarquía entre ellos e inicialmente solo se almacenaban los contornos y daba problemas al ejecutar el código.

```
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

```
cnts, hierarchy = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
```

Corrección

Posteriormente se tuvo que modificar la línea de código que cargaba los contornos en la variable *cnts*, debido a que estaba fuera de la condición que implicaba detección de contorno y hacia que el código fuese poco robusto e incluso dejara de ejecutarse el programa en algún momento. En esta parte del programa realizamos diversas pruebas para ver que contornos estaba detectando y vimos que el primero que detectaba era el que se requería por lo que, se cambio el código para que fuera el primero el que se almacenara en la variable *cnts*.

```
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if imutils.is_cv2() else cnts[1]
center = None

if len(cnts) > 0:
    c = max(cnts, key=cv2.contourArea)
    timeInterval = time.time() - start_time
    (x, y), radius = cv2.minEnclosingCircle(c)
```

```
if len(cnts) > 0:
    cnts = cnts[0]
    center = None
    c = cnts
    timeInterval = time.time() - start_time
    (x, y), radius = cv2.minEnclosingCircle(c)
```

Corrección

El siguiente cambio que se realizo fue la modificación de la parte del código en la que se cargaba el contorno encontrado en la variable *c* que posteriormente se le pasaba a la función *cv2.minEnclosingCircle*. Originalmente se cargaba el máximo de la variable *cnts* y la variable *key* que almacenaba el área del contorno, pero esto no funcionaba y se cambio para que se almacenara directamente la variable *cnts* con el contorno requerido en *c*.

```
#c = max(cnts, key = cv2.contourArea)
```

```
c = cnts
```

Corrección

Y el último cambio relacionado con la detección que se realizo fue en la parte del código en la que si no se detectaba pelota lo único que se hacía era inicializar el error. En esta parte se añadió que cuando no se detectara pelota se actualizara la posición x e y con la última posición detectada. Este cambio se realizo debido a que en determinados momentos se perdía la referencia de la posición y se producía un comportamiento brusco de la plataforma.

```
    if radius > 10:
        cv2.putText(img,str(int(x)) + ";" + str(int(y)).format(0, 0),(int(x)-50, int(y)-50), cv2.FONT_HERSHEY_SIMPLEX,1, (255, 255, 255
        cv2.circle(img, (int(x), int(y)), int(radius),(0, 255, 255), 2)
        PIDcontrol(int(x),int(y),prevX,prevY,consigneX,consigneY)
        start_time = time.time()
else:
    sommeErreurX, sommeErreurY = 0,0
```

```
if radius > 10:
    cv2.putText(img,str(int(x)) + ";" + str(int(y)).format(0, 0),(int(x)-50, int(y)-50), cv2.FONT_HERSHEY_SIMPLEX,1, (255, 255, 255), 2)
    cv2.circle(img, (int(x), int(y)), int(radius),(0, 255, 255), 2)
    PIDcontrol(int(x),int(y),prevX,prevY,consigneX,consigneY)
    start_time = time.time()
else:
    sommeErreurX, sommeErreurY = 0,0
    x, y = prevX, prevY
```

Corrección

10.5.2. Control

Una vez que se solucionaron los problemas en la detección de pelota, el siguiente problema que apareció, fue en la parte de control. Cuando se compiló y probó la movilidad de los motores, se vio que el giro se producía en sentido contrario al que debería. Por lo que se cambio el ángulo beta por uno modificado que es su opuesto para que la plataforma girara en el sentido deseado.

```
if alpha <= 35 and beta <= 360 and arduinoIsConnected == True and startBalanceBall == True:
    ser.write((str(dataDict[(alpha,beta)])+"\n").encode())
```

```
if beta_m >= 0:
    beta_m = beta_m
    beta_m = round(round(beta_m / 0.2) * 0.2, -int(floor(log10(0.2))))
else:
    beta_m = 360 - abs(beta_m)
    beta_m= round(round(beta_m / 0.2) * 0.2, -int(floor(log10(0.2))))

if alpha <= 30 and beta <= 360 and arduinoIsConnected == True and startBalanceBall == True:
    ser.write((str(dataDict[(alpha,beta_m)])+"\n").encode())

alpha = prevAlpha * omega + (1-omega) * alpha
beta_m = prevBeta * omega + (1-omega) * beta_m
```

Corrección

Otro de los cambios que se introdujeron en el control, aunque viene producido al realizarse el análisis cinemático, fue la limitación del rango de los motores, en lugar de limitar el ángulo alpha a 35 grados como estaba en el código original, se cambió a 30 grados. Este cambio se realizó para que la plataforma no intentara llegar a posiciones no alcanzables.

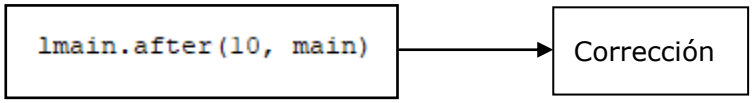
```
if alpha > 35:
    alpha = 35
```

```
if alpha > 30:
    alpha = 30
```

Corrección

Para terminar con los cambios en el control se cambió el sample time del programa debido a que con el que estaba configurado no le daba tiempo a generar la interface cuando se iniciaba el programa.

```
lmain.after(5, main)
```



10.5.3. Otros

Por último aunque no es un cambio que afecte a la funcionalidad de la plataforma se ha introducido en el código unas matrices inicialmente vacías para poder obtener datos de posición y ángulos en función del tiempo que nos aporten información del comportamiento de la plataforma a lo largo del programa.

```
x_a = np.zeros((12000, 1))
y_a = np.zeros((12000, 1))
Ix_a = np.zeros((12000, 1))
Iy_a = np.zeros((12000, 1))
consigneX_a = np.zeros((12000, 1))
consigneY_a = np.zeros((12000, 1))
alpha_a = np.zeros((12000, 1))
beta_m_a = np.zeros((12000, 1))
errorX_a = np.zeros((12000, 1))
errorY_a = np.zeros((12000, 1))
preX_a = np.zeros((12000, 1))
preY_a = np.zeros((12000, 1))
time_a = np.zeros((12000, 1))
f = 0
c = 0
write_index=0
```


Autor: **Raúl López Martín**
[424.19.38]

```
write_index=write_index+1
if write_index<12000:
        x_a[write_index] = ballPosX

if write_index<12000:
        y_a[write_index] = ballPosY

if write_index<12000:
        consigneX_a[write_index] = consigneX

if write_index<12000:
        consigneY_a[write_index] = consigneY

if write_index<12000:
        Ix_a[write_index] = Ix

if write_index<12000:
       Iy_a[write_index] = Iy

if write_index<12000:
       alpha_a[write_index] = alpha

if write_index<12000:
       beta_m_a[write_index] = beta_m

if write_index<12000:
       errorX_a[write_index] = sommeErreurX

if write_index<12000:
       errorY_a[write_index] = sommeErreurY

if write_index<12000:
       preX_a[write_index] = prevX

if write_index<12000:
       preY_a[write_index] = prevY


if write_index<12000:
       time_a[write_index] = time_interval
```

```
def endProgam():
    global x_a, y_a, consigneX_a, consigneY_a, Ix_a, Iy_a, alpha_a, beta_m_a, errorX_a, errorY_a, preX_a, preY_a, time_a
    fic = open("archivoX.txt","w")
    fic.writelines("%f\n" % f for f in x_a)
    fic.close()
    fic1 = open("archivoY.txt","w")
    fic1.writelines("%f\n" % f for f in y_a)
    fic1.close()
    fic2 = open("archivoCX.txt","w")
    fic2.writelines("%f\n" % f for f in consigneX_a)
    fic2.close()
    fic3 = open("archivoCY.txt","w")
    fic3.writelines("%f\n" % f for f in consigneY_a)
    fic3.close()
    fic4 = open("archivoIx.txt","w")
    fic4.writelines("%f\n" % f for f in Ix_a)
    fic4.close()
    fic5 = open("archivoIy.txt","w")
    fic5.writelines("%f\n" % f for f in Iy_a)
    fic5.close()
    fic6 = open("archivoalpha.txt","w")
    fic6.writelines("%f\n" % f for f in alpha_a)
    fic6.close()
    fic7 = open("archivobeta_m.txt","w")
    fic7.writelines("%f\n" % f for f in beta_m_a)
    fic7.close()
    fic8 = open("Error_x.txt","w")
    fic8.writelines("%f\n" % f for f in errorX_a)
    fic8.close()
    fic9 = open("Error_y.txt","w")
    fic9.writelines("%f\n" % f for f in errorY_a)
    fic9.close()
    fic10 = open("Prev_x.txt","w")
    fic10.writelines("%f\n" % f for f in preX_a)
    fic10.close()
    fic11 = open("Prev_y.txt","w")
    fic11.writelines("%f\n" % f for f in preY_a)
    fic11.close()
    fic12 = open("time.txt","w")
    fic12.writelines("%f\n" % f for f in time_a)
    fic12.close()
    controllerWindow.destroy()
```

11. EXPERIMENTACIÓN COMPLETA

A continuación, con los datos obtenidos anteriormente se va a realizar el estudio del comportamiento de la plataforma de forma completa. Con esto lo que se quiere obtener, es el resultado del funcionamiento completo de la plataforma de forma grafica.

En el grafico que se observa a continuación, se muestra la posición de la pelota en función del tiempo.

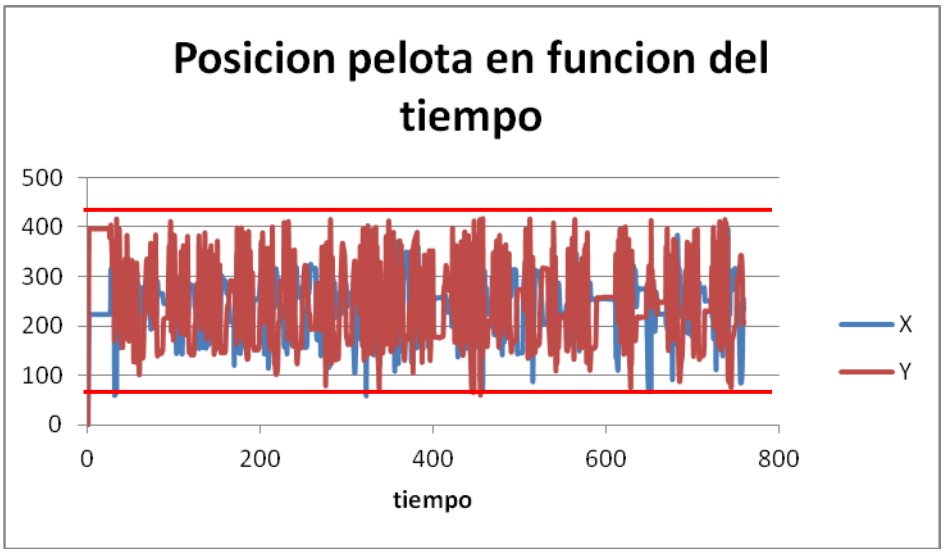


Figura 63. Grafico de posición de pelota respecto al tiempo. Perturbación constante.

En el grafico anterior, se observa que la pelota está sometida a una perturbación constante. En líneas generales se puede ver como la plataforma intenta estabilizar la pelota para que no se salga de la plataforma, manteniéndose entre los límites marcados, que corresponden con los extremos de la plataforma e intentando llevarla al centro que es la consigna previamente establecida en el punto central (240,240).

En el siguiente grafico que se muestra, también se puede observar la posición de la pelota en función del tiempo. En este caso se ha dejado que la plataforma intente estabilizar la pelota en centro de la plataforma que es punto (240,240), y se ha sometido a perturbaciones puntuales.

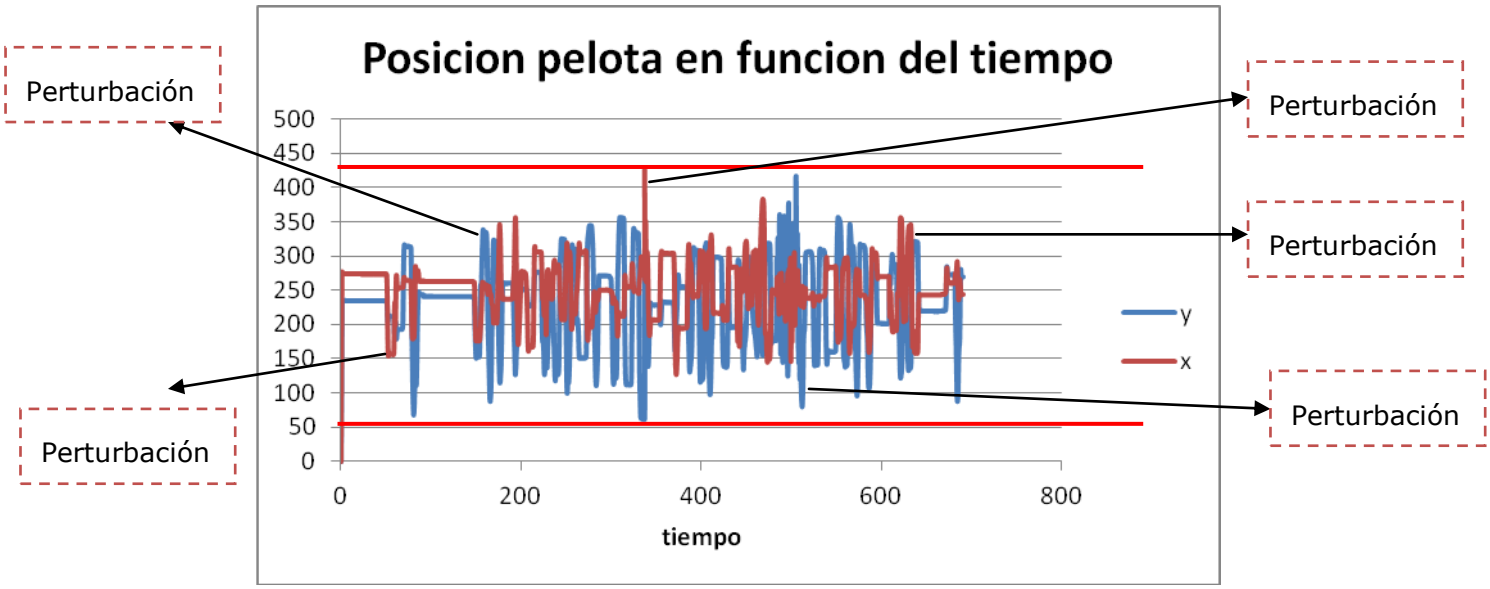


Figura 64. Grafico de posición de pelota respecto al tiempo. Perturbaciones puntuales.

En el grafico anterior se pueden observar las perturbaciones puntuales anteriormente comentadas. Estas perturbaciones han sido provocadas manualmente dándole con una varilla a la pelota y alejándola de su posición de equilibrio.

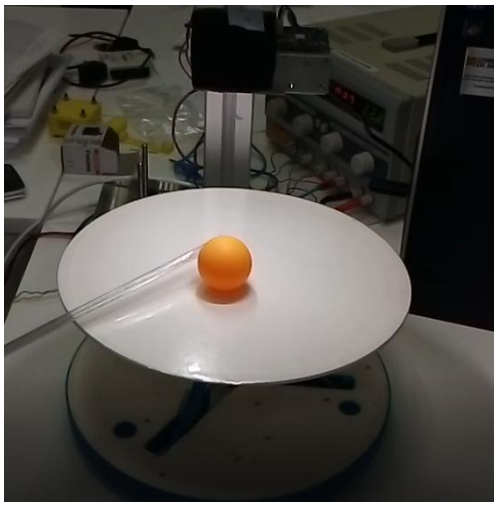


Figura 65. Provocación de la perturbación.

12. TRABAJOS FUTUROS

En el futuro, las líneas de trabajo a seguir son:

- Mejora de la parte mecánica.
    - Eliminación de holguras, rigideces y fricciones.
    - Posible cambio de motores para dar más potencia a la plataforma.
    - Mejora de piezas internas para facilitar su montaje, debido a que actualmente hay zonas con poco espacio para introducir herramientas.

En el apartado relacionado con el software, se podría trabajar en:

- La realización de diversas trayectorias.
- Mejora del tiempo de respuesta.
- Mejorar el control para identificar la dinámica de la pelota y adaptar los parámetros.
- Combinación de IMU y giróscopo para cerrar el lazo de control.

A pesar de ser concebido como un demostrador de capacidades mecatrónicas en un futuro la tecnología utilizada se puede implantar para estabilizar cámaras en movimiento, herramientas, entre otras.

13. BIBLIOGRAFÍA

Meyer, M. A. (14 de marzo de 2014). *Obtención del Espacio de Trabajo de la Plataforma de Gough-Stewart mediante técnicas CAD.* Obtenido de dspace.umh.es/bitstream/11000/1610/7/TD%2BMiguel%2BAngel%2BOliva%2Bi%2BMeyer.pdf

Picón, O. S. (enero de 2008). *Síntesis, Análisis y Diseño de Manipuladores Paralelos de Baja Movilidad .* Obtenido de www.ehu.eus/compmech/welcome/doc/dissertation_Salgado.pdf

Ramaekers, P., Hermans, T., De Meulenaere, P., Denil, J., & Anthonis, J. (agosto de 2011). *Incorporation of AUTOSAR in an Embedded Systems Development Process: A Case Study.* Obtenido de https://www.researchgate.net/publication/221593494

wikipedia. (s.f.). Obtenido de https://en.wikipedia.org/wiki/Stewart_platform

Relación de documentos

(X) Memoria ..... NN páginas

(_) Anexos ..... NN páginas

La Almunia, a 27 de 11 de 2019

Firmado: Raúl López Martín

Etiquetas para CD/DV

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

Nº TFG:

[424.19.38]

Director: Juan Diego Jaria Gazol /José Manuel Rodríguez Fortún

Fdo:
¡Error! No hay texto con el estilo especificado en el documento.

Título TFG:

Analisis y puesta en marcha de una plataforma triaxial como demostrador tecnologico

Autor:

Raúl López Martín

27/11/2019

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

Nº TFG:

[424.19.38]

Director: Juan Diego Jaria Gazol /José Manuel Rodríguez Fortún

Fdo:
Raúl López Martín

Título TFG:

Analisis y puesta en marcha de una plataforma triaxial como demostrador tecnologico

Autor: Raúl López Martín

27/11/2019

Escuela Universitaria Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA

DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

Análisis y puesta en marcha de una plataforma triaxial como demostrador tecnológico

[424.19.38]

Autor: Raúl López Martín

Director: Juan Diego Jaria Gazol / José Manuel Rodriguez Fortún

Fecha: 27/11/2019