Universidad Católica Boliviana de San Pablo

IMT-202 Control II

SELF-BALANCING ROBOT

Docente:

Benjamín Pinaya

Integrante.-

Héctor Yugar Carrasco

Domingo 17 de diciembre del 2017

I.- TABLA DE CONTENIDOS.

II.- TABLA DE FIGURAS…….……3

III.- PROYECTO……..…….……4

1.- RESUMEN…….……..4

2.- INTRODUCCION Y ANTECEDENTES……..……...4

2.1.- INTRODUCCION…….....4

2.2.- ANTECEDENTES……..5

3.- ELABORACION DEL PROYECTO …..……..7

3.1.- LISTA DE MATERIALES……..8

3.2.- MODELADO DEL SISTEMA……..11

OBSERVABILIDAD…….14

CONTROLABILIDAD……..15

METODO DE CONTROL ……...16

4.- CONCLUSIONES YRESULTADO…….....17

5.- CALENDARIO ……..……..18

6.- REFERENCIAS…….……19

II.- TABLA DE FIGURAS.

Figura 3.1.- DISEÑO DE LA IMPRESIÓN DEL CHASIS……**7**

Figura 3.2.- MODELO FINAL DEL CHASIS……**7**

Figura 3.3.- ARDUINO UNO……**8**

Figura 3.4.- MPU6050……**8**

Figura 3.5.- L298N……**8**

Figura 3.6.-KIT DE MOTOR DC ……**9**

Figura 3.7.- CONEXIÓN DEL CIRCUITO……**9**

Figura 3.8.- PRIMER PROTOTIPO……**10**

Figura 3.9.- NUEVA ESTRUCTURA PARA EL CHASIS……**10**

Figura 3.10.-MODELO FINAL DEL ROBOT……**10**

Figura 3.11.-DIAGRAMA DE RUEDAS……**12**

Figura 3.12.-DIAGRAMA DE CUERPO LIBRE……**13**

Figura 3.13.-IMPLEMENTACION DEL PID……**16**

III.- PROYECTO

1. RESUMEN.

En el proyecto narrado a continuación explica y narra la construcción y 6control de un robot balancín, se detallaran: los conocimientos previos necesarios para el proyecto; los materiales utilizados y los métodos de control para el sistema; los resultados obtenidos para concluir exitosamente el proyecto; un calendario con las fechas de la realización del proyecto; finalmente se detallaran las referencias usadas para la redacción del reporte. Con el motivo de la demostración práctica de la teoría vista en control II. Un robot balancín es un sistema mecánico, en el cual a partir de un método de control se mantiene una posición vertical respecto a la horizontal.

Este proyecto consta de una estructura para sostener los circuitos y motores, el cual funcionara como un péndulo. Se implementaron 2 sensores, y un arduino como sistema de operación y a partir de pruebas en el sistema se determinara el método de control que se desenvuelva mejor en el sistema, para este proyecto se utilizara un control PID. Como resultados del proyecto se busca obtener un sistema de péndulo invertido funcional junto a su correcto sistema de control.

2. INTRODUCCION Y ANTECEDENTES.

2.1.- Introducción.-

Durante la última década, el robot balancín ha sido motivo de muchas investigaciones debido a la inestabilidad dinámica del sistema, este robot se caracteriza principalmente por su habilidad de mantenerse vertical, solamente haciendo uso de 2 motores, tienes multiples usos, debido a su maniobrabilidad es posible su uso en el transporte en industrias y su manejo terrenos complicados.

La base de un robot balancín es un péndulo invertido, el cual es bastante conocido como un sistema mecánico usado en control debido a sus usos en el transporte aéreo, el control de un robot bípedo que camina, etc. Existen varios tipos de péndulos invertidos, se pasara a mencionar algunos y se hará un pequeño repaso de su funcionamiento.

El péndulo en carro móvil. Este tipo de vehículo consiste en un carro autónomo con un péndulo giratorio encima, el sistema consiste en controlar el péndulo con el carro en movimiento, si el movimiento del carro es lineal el control no es tan complejo, pero si se mueve en trayectorias circulares se lo considera un péndulo de Furuta.

El péndulo de Furuta[1]**.** Este péndulo fue creado por el doctor Furuta del instituto de tecnología de Tokio, Japón, consiste en un sistema de dos grados de libertad que son el móvil que se mueve en trayectorias circulares, el brazo para y el péndulo como tal. El brazo gira alrededor de su eje y el péndulo (ubicado al extremo del brazo) gira de forma colineal y perpendicular al brazo.

El péndulo invertido clásico. Este tipo de péndulo consiste en un riel sobre el cual se desliza un móvil en forma lineal, el péndulo (ubicado en el móvil) gira libremente mientras se mueve el carro.

Para el control se utilizara el lenguaje de programación C++, y se programara un PID, se vio más conveniente utilizar un PID.

Para la operación del sistema se utilizara un arduino, o algún micro controlador que este a disposición, esto para facilitar la programación y la conexión de los sensores y el motor,

2.2.- Antecedentes.-

Antes de realizar el proyecto se debe entender distintos conceptos, los cuales son la base de la procedencia del proyecto, dichos conceptos se tomaran en cuenta durante la realización del péndulo invertido, dichos conceptos serán mencionados y brevemente explicados a continuación:

Movimiento Oscilatorio[2]**.-** Como su nombre indica es un movimiento que se caracteriza por las oscilaciones en el mismo. Con oscilación se refiere a un objeto que se mueve alrededor de su punto de equilibrio, dicho movimiento presente un periodo en el que su comportamiento se desempeña de la misma manera anteriormente realizada, dicho de otra manera el objeto presenta en su movimiento un desplazamiento repetitivo.

Péndulo[3]**:** La palabra péndulo se refiere a un sistema que presenta una masa que cuelga a través de, ya sea un hilo, una barra, una estructura, etc., de un extremo y en el otro extremo se encuentra sujetada en un punto fijo. Dicho sistema puede presentar un movimiento oscilatorio debido a distintas circunstancias, como ejemplo la acción de la gravedad o la implementación de una fuerza.

Las aplicaciones de un péndulo son variadas, llegando a ser utilizado en metrónomos; relojes; determinar la intensidad de la gravedad; etc. Al referirse a un péndulo se debe considerar dos conceptos del mismo.

- **Péndulo Físico**[4]**.-** El termino péndulo físico es referido a cualquier sistema que presente una masa u objeto colgante, sujeto a un eje fijo, es también llamado péndulo compuesto, dicho sistema se encuentra en un movimiento oscilatorio, el movimiento que presenta dicho péndulo se genera cuando la masa es separada de su posición de equilibrio y se la suelta para retornar a dicha posición, en dicho movimiento la masa se desplaza libremente alrededor de su propio eje. Es llamada péndulo físico debido a la razón de ser un sistema real, posible de construir.

- **Péndulo Simple**[5]**:** en contra parte al péndulo físico, el péndulo simple o también llamado péndulo matemático se refiere a un sistema de una masa suspendida sujeta a un punto fijo, en el cual le movimiento que presenta es lineal y simétrico con respecto a su eje, dicho movimiento no presenta desviaciones y en el cual el peso de la masa colgante es despreciable, que se

puede suponer que es cero. Debido a estas condiciones se considera un sistema ideal e inexistente en la realidad.

Péndulo Invertido[6]**:** El péndulo invertido como su nombre indica busca funcionar de manera inversa a la de un péndulo, consta inicialmente de una barra sujeta de un eje fijo, a su vez dicho eje se encuentra en un objeto móvil. Al aplicar movimiento en el objeto móvil la barra debe terminar en una posición perpendicular al movimiento aplicado, con este fin al sistema se le aplica un método de control, con el cual se busca mantener dicha posición perpendicular a pesar de distintas interferencias, dichas interferencias se las considera como ruido de medición, el cual se aplica como fuerzas externas a tomar en cuenta, como la fuerza de la gravedad, el viento, interferencias manuales, etc.

Control PID[8]**:** Al referirse al control PID o controlador PID se habla de un método en de control, el cual permite controlar distintos tipos de variables de salida, ya sea temperatura, presión, posición, etc., dicho control se genera a partir de un lazo de retroalimentación en el sistema.

Las siglas de PID provienen de los nombres de los parámetros que intervienen en el control, siendo estos respectivamente la ganancia proporcional, la ganancia integral y la ganancia derivativa, donde se obtienen las abreviaciones P. I. D. dichos parámetros en el controlador cumplen distintas funciones cada uno; la ganancia proporcional determina la diferencia entre el error de salida actual y el error de salida anterior; la ganancia integral determina la acción correctiva en el error y el tiempo en el que la misma es aplicada, dicha acción puede causar en el sistema inestabilidad, debido a que mientras más pequeño sea el error su ajuste es más rápido; la ganancia derivativa suaviza la velocidad de la corrección del error del integral, y produce una corrección al error antes que este tome una magnitud porcentualmente alta. Se puede utilizar los parámetros del PID de distintas maneras, dichas maneras son las distintas combinaciones de los parámetros, existiendo las siguientes combinaciones: proporcional (P): proporcional integral (PI); proporcional derivativo (PD); y proporcional integral derivativo (PID). Como conocimiento básico se muestra la ecuación básica en un sistema PID:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\partial t + K_d \frac{d}{dt} e(t) \quad (1)$$

Donde: K_p,K_i y K_d son los coeficientes de ganancias proporcional, integral y derivativo respectivamente, t es el tiempo y e(t) es el error en función del tiempo.

Arduino[9][10]**:** Un Arduino es un microcontrolador de uso simple, el cual será usado como controlador del movimiento en el sistema del sistema del péndulo invertido.

Tiempo de asentamiento[12]**.-** El tiempo de asentamiento en un sistema de control es la respuesta en el tiempo que provee el método de control utilizado en el sistema. Dicha respuesta es la corrección a cualquier desviación del punto de referencia o a la posición deseada en el sistema.

MPU6050 [13]**:** el MPU6050 es un chip que contiene los sensores acelerómetro, giroscopio y magnetómetro adaptado para su uso en arduino, usa 16 bits que se convierten de analógico a digital gracias a la librería I2C proveida por Arduino.

3. Elaboración del proyecto.

Primeramente se hizo el armado mecánico del sistema, empezando por un listado de los materiales y una breve explicación de por qué el uso de cada uno de ellos y luego un modelo de cómo se ve el sistema terminado.

3.1. Lista de materiales.

Chasis: Comenzando por el chasis del robot, el cual fue realizado en SolidWorks, pieza por pieza como se muestra a continuación:

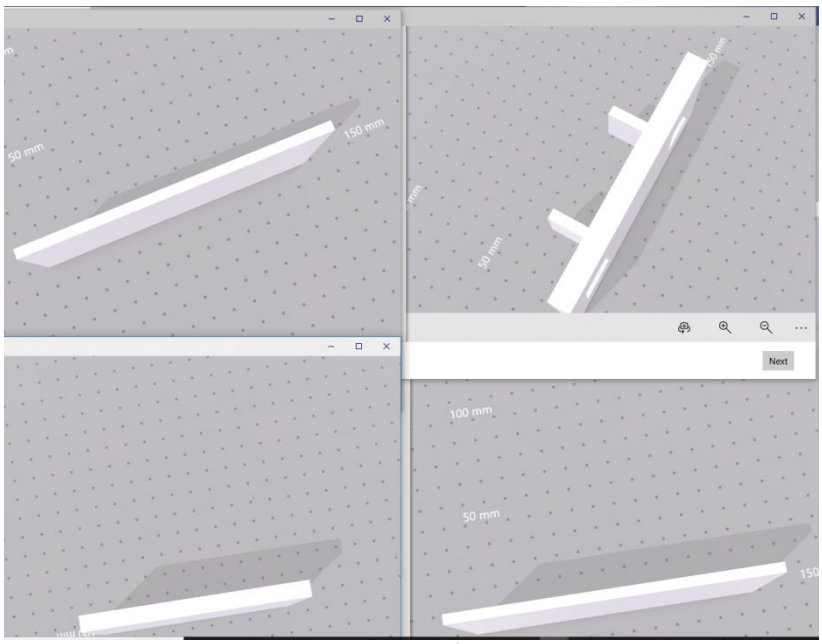

Figura 3.1. Diseño para impresión del chasis (figura propia)

Se eligió el plástico de impresión 3D debido a que es ligero y resistente, pero caro, este se ensamblo para obtener una estructura clásica de un robot balancín, cada cavidad es para colocar el resto de los materiales, cada parte mide aproximadamente 15x1 cm de anchura, a excepción de la parte de abajo que sostiene a los motores la cual es más gruesa con 2 cm. de grosor obteniendo una estructura final como la siguiente:

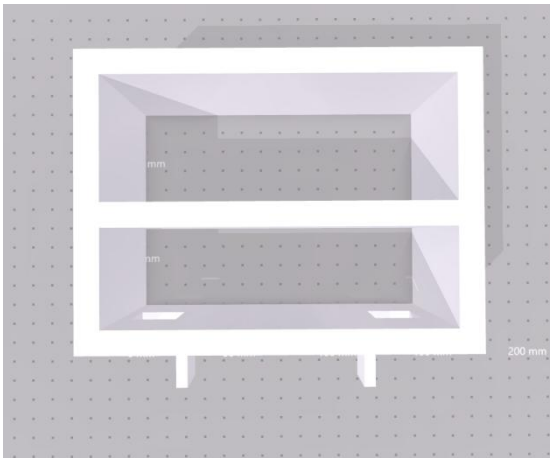

Figura 3.2. Modelo final del chasis (diseño propio)

Arduino: como se menciono antes, el Arduino es el microcontrolador de elección debido a su fácil programación, sus múltiples usos y librerías, se eligió el arduino uno debido a que es pequeño y cabe perfectamente en el chasis, se puede utilizar también un Arduino nano o mega.

Figura 3.3. Arduino uno [14]

MPU6050: el MPU6050 fue el sensor de elección debido a que su conexión a arduino es muy sencilla, su precisión, aun sin hacer uso de un estimador como un filtro, es muy buena y se obtienen lecturas bastante precisas, es necesario obtener una lectura del ángulo del chasis y de la velocidad de los motores, es por todo esto que se utilizo este sensor para la construcción del robot.

Figura 3.4. MPU6050 [15]

L298N: este shield es esencialmente un puente H que se comunica con Arduino y es usado para el comunicar 2 motores DC al Arduino de manera fácil, haciendo uso de librerías proveidas por Arduino.

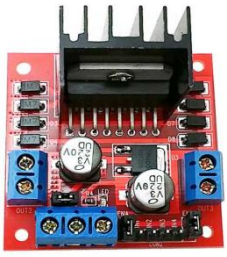

Figura 3.5. L298N [16]

Motores DC: debido a su fácil conexión con el L298N, su alimentación es estándar (9V) y que no es necesario un motor especial para este tipo de proyecto, fueron los motores elegidos para este trabajo.

Figura 3.6. kit de motor DC[17].

Las conexiones realizadas para el circuito son las siguientes, la imagen fue diseñada en fritzing:

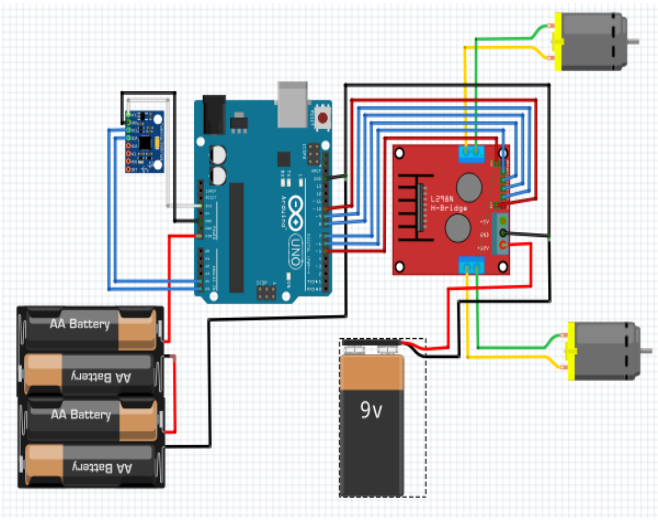

Figura 3.7. Conexión del circuito. (figura propia)

Una vez realizada la conexión del circuito y de colocarla dentro del chasis se obtuvo el primer prototipo del robot balancín:

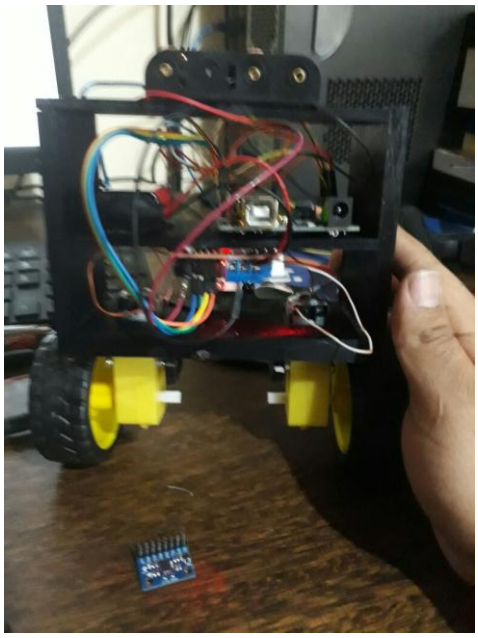

Figura 3.8. Primer prototipo del robot (figura propia)

Los problemas que se encontraron con este primer prototipo, fue el mal cálculo de la distancia de los motores hacia el chasis, como se observa en la figura existía gran cantidad de rozamiento entre las ruedas y el chasis, eso provocaba que haya pérdida de fuerza en los motores dificultando su control y manejo.

Luego se rehízo la estructura del chasis corrigiendo los problemas mencionados arriba obteniendo la siguiente estructura, igualmente impresa en 3D.

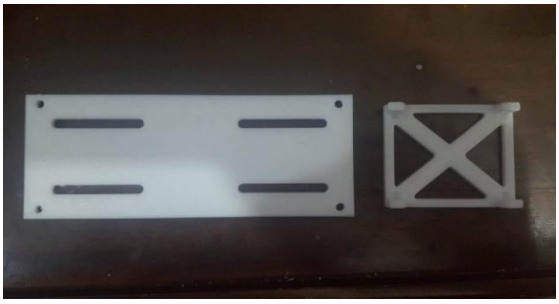

Figura 3.9. Nueva estructura para el chasis (figura propia)

Con esto se arreglo el problema de rozamiento con los motores y se obtuvo una estructura más pequeña y por ende más fácil de controlar, siendo esta la siguiente:

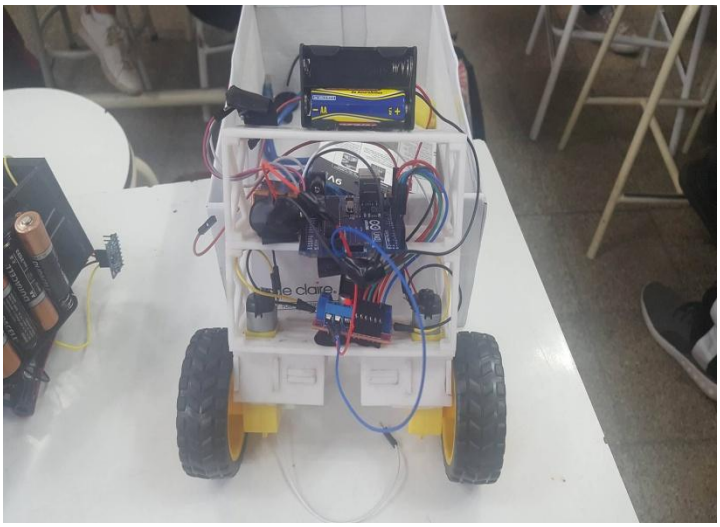

Figura 3.10. Modelo final del robot. (figura propia)

3.2. Modelado del sistema

Modelado dinámico

Primeramente se obtuvo el modelado dinámico del sistema empezando por la obtención de datos del robot, siendo los siguientes:

Radio de las ruedas r: 5.1 cm.

Masa del chasis M_b: 1130 gr.

Masa de las ruedas M_w: 30 gr.

Longitud hacia el centro de masa L: 7 cm.

Gravedad de La Paz: 9.81 $\frac{m}{s^2}$.

Datos teóricos de los motores [18]:

Torque del motor K_m: 0.006123 $\frac{Nm}{A}$.

Voltaje de alimentación Va: 9v.

Resistencia nominal: 3Ω.

Constante electromagnética (emf) ke: 0.006087 $\frac{Vs}{rad}$.

Para las inercias de nuestro sistema se utilizo la siguiente ecuación

$M_w r^2 = I_w$ Representa la inercia de las ruedas (1)

$M_b L^2 = I_b$ Representa la inercia del cuerpo (2)

$I_b = 762.6 \frac{gr}{cm^2}$

$$I_w = \ 340 \ \frac{gr}{cm^2}$$

Primero se modeló la física en las ruedas siendo la siguiente:

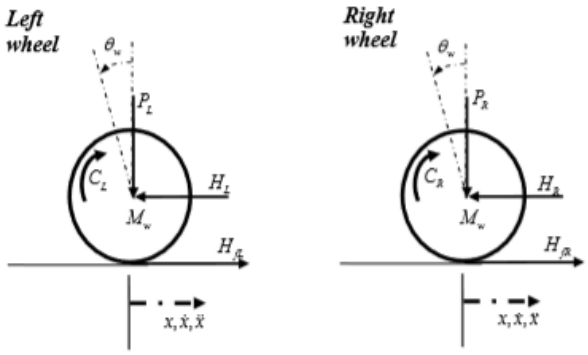

Figura 3.11. Diagrama de las ruedas [18] pag. 26.

De este diagrama y haciendo uso de las leyes de Newton, se obtiene la sumatoria de fuerzas en el eje horizontal siendo las mismas:

$$\sum F_x = Ma \ (3)$$

$$M_w \ddot{x} = H_{fR} - \ H_R \ (4)$$

Donde:

$$H_{fR} = \frac{-K_m K_e}{Rr} \dot{\emptyset} w + \frac{K_m}{Rr} Va - \frac{I_w}{r} \ \ddot{\emptyset} w \ (5)$$

Reemplazando (5) en (4) para ambas ruedas se obtiene:

Para la rueda izquierda:

$$M_w \ddot{x} = \frac{-K_m K_e}{R} \dot{x} + \frac{K_m}{Rr} Va - \frac{I_w}{r} \ \ddot{x} - H_l \ (6)$$

Para la rueda derecha:

$$M_w \ddot{x} = \frac{-K_m K_e}{R} \dot{x} + \frac{K_m}{Rr} Va - \frac{I_w}{r} \ \ddot{x} - H_R (7)$$

Se reemplaza $\dot{x}$ debido a que la velocidad angular se anula con el desplazamiento de r entonces $\dot{x} = r\dot{\emptyset} w$ entonces reemplazando eso en (5) se obtienen las ecuaciónes (6) y (7).

Para el diagrama del cuerpo libre del chasis, se lo considera como si fuera un péndulo invertido, pero aplicando los diagramas de ruedas se obtiene lo siguiente:

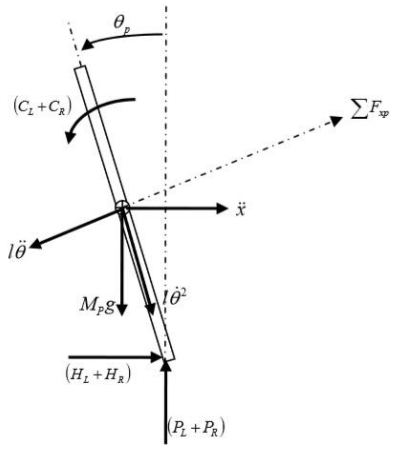

Figura 3.12. Diagrama de cuerpo libre [18] pag. 27.

De este diagrama y haciendo uso de las leyes de Newton, se obtiene la sumatoria de fuerzas en el eje horizontal siendo las mismas:

$$\sum F_x = Ma \ (3)$$

$$(H_L + H_R) - M_b l\ddot{\emptyset}\cos(\emptyset) + M_b l\dot{\emptyset}^2 \sin\emptyset = M_b\ddot{x} \ (8)$$

Con esta ecuación, se reemplaza con 6 y 7, luego se despeja $\ddot{x}$ y $\ddot{\emptyset}$ para obtener las ecuaciones de movimiento final que son:

$$\ddot{\emptyset} = \frac{M_b L}{I_b + M_b L^2}\ddot{x} + \frac{2K_m K_e}{Rr(I_b + M_b L^2)}\dot{x} - \frac{2K_m}{R(I_b + M_b L^2)}Va + \frac{M_b g L}{I_b + M_b L^2}\emptyset \ (\mathbf{9})$$

$$\ddot{x} = \frac{2K_m}{2M_w + \frac{2I_w}{r^2} + M_b}Va - \frac{2K_m K_e}{Rr^2(2M_w + \frac{2I_w}{r^2} + M_b)}\dot{x} + \frac{M_b L}{2M_w + \frac{2I_w}{r^2} + M_b}\ddot{\emptyset} \quad (\mathbf{10})$$

A partir de estas ecuaciones se hace uso del jacobiano para obtener las ecuaciones de estado, que son las siguientes:

Primeramente el orden de las variables es el siguiente.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\emptyset} \\ \ddot{\emptyset} \end{bmatrix} (11)$$

Donde:

$\dot{x}$= velocidad lineal en x.

$\ddot{x}$= aceleración lineal en x.

$\dot{\emptyset}$= velocidad angular.

$\ddot{\emptyset}$= aceleración angular.

Luego se obtiene la matriz A, como ya se explico esta se obtuvo haciendo uso de derivadas parciales con jacobianos.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{2K_m K_e (M_b L r - I_b - M_b L^2)}{Rr^2 * \left[I_b 2M_w + \frac{2I_w}{r^2} + M_b + 2M_b L^2 \left(M_w + \frac{I_w}{r^2}\right)\right]} & \frac{M_b^2 g L^2}{2M_w + \frac{2I_w}{r^2} + M_b} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{2K_m K_e (rD - M_b L)}{Rr^2 * I_b 2\left[M_w + \frac{2I_w}{r^2} + M_b + 2M_b L^2 \left(M_w + \frac{I_w}{r^2}\right)\right]} & \frac{M_b g L * D}{2M_w + \frac{2I_w}{r^2} + M_b} & 0 \end{bmatrix} (12)$$

Como se observa en la matriz A la primera columna desaparece completamente, esto debido a que el desplazamiento en x es irrelevante para el sistema, ya que, no importa cuánto se mueva el robot en x mientras se mantenga vertical.

Reemplazando los datos del robot en la matriz A se obtiene:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.009731 & 11.16 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.02932 & 172.1 & 0 \end{bmatrix} (13)$$

Como se observa todos los datos son reales, lo cual lleva a pensar que el modelado del sistema podría correctamente planteado, uno de los más grandes problemas que se tuvo al hacer esta parte del proyecto fue que, con el primer modelado que se obtuvo (el cual estaba incorrectamente planteado) obteníamos una matriz A con números imaginarios, con una controlabilidad nula, esto llevo a buscar y demostrar otro modelado matemático mejor, llegando a este modelado que nos permitió seguir adelante.

Controlabilidad: Observando los resultados de la matriz A se propone una matriz B la cual es:

$$B = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} (14)$$

La matriz B representa los actuadores en nuestro sistema, nuestros actuadores físicos son los motores DC principalmente y como se tiene lectura del ángulo y la velocidad

(gracias al MPU) se dice que tenemos datos como se marcan en la matriz ya mostrada.

La razón de obtener la matriz B es para sacar la controlabilidad de nuestro sistema, esto nos permitirá ver si el sistema es controlable o no, de no ser controlable habría que recalibrar nuestros actuadores aumentando variables para obtener un sistema controlable. La matriz de controlabilidad obtenida gracias a operadores en matlab, es la siguiente:

$$\mathrm{Ctrb(A,B)} = \begin{bmatrix} 0 & 0 & 0.0011 & 0 \\ 0 & 0.0011 & 0 & 0.1921 \\ 0 & 0 & 0.017 & 0 \\ 0 & 0.0172 & 0 & 2.9624 \end{bmatrix} (15)$$

Se obtienen estos datos debido a que se están reemplazando con la matriz A ya resuelta, ahora con el rango o determinante de esta matriz se puede determinar si nuestro sistema es controlable o no, para saber eso se tiene que obtener un rango igual a 4, el rango que se obtuvo fue:

$$\mathrm{Rank(ctrb(A,B)= 4\ (16)}$$

Entonces se puede decir con seguridad que nuestro sistema es controlable, ahora para determinar que dato en nuestro sistema aporta mayor grado de controlabilidad se usa el comando de matlab svd, obteniendo:

$$svd = \begin{bmatrix} 2.9687 \\ 0.0172 \\ 0.0001 \\ 0.0000 \end{bmatrix} (17)$$

Como se observa, el dato que aporta mayor controlabilidad al sistema es el desplazamiento en X, lo cual es completamente lógico, pero no del todo cierto, debido a que para controlar un robot balancín la lectura del ángulo es vital para controlar un robot sin ese dato no es posible controlarlo.

Observabilidad: Observando los resultados de la matriz A, se puede proponer una matriz C, en base a los sensores que se están utilizando para controlar el sistema, en el caso de un robot balancín se está utilizando un acelerómetro y un giroscopio. La matriz C propuesta es la siguiente:

$$C = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} (18)$$

Se propone esta matriz ya que podemos controlar la velocidad y aceleración de los motores, y se tiene datos del ángulo del robot, para determinar la observabilidad de un sistema la determinante de la matriz de observabilidad que se mostrara a continuación deber ser real y diferente a 0, el resultado que se obtuvo es el siguiente:

$$\text{Obsv(A, C)} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -0.0390 & 183.2754 & 0 \\ 0 & 0.0004 & -0.4358 & 183.2754 \end{bmatrix} (19)$$

Otro grande problema que se vio usando el modelado erróneo, fue que esta matriz salía con números imaginarios y se obtenía una determinante imaginaria, lo cual no se podía determinar si era observable o no.

$$\det\left(obsv(A, C) = 3.3590e + 04\right) (20)$$

Como se observa la determinante del sistema es real y diferente a 0 lo cual nos lleva a concluir que el sistema es observable, para determinar que variable es la que aporta mayor información al sistema se hace uso del gramiano de nuestro sistema, y para eso se propone una matriz D que representa el ruido al sistema, se planteara un escenario teórico en el que no se tiene ningún tipo de perturbación ni ruido en el sistema siendo D=0, entonces para obtener el gramiano del sistema de usa el comando gram en matlab de nuestro sistema, lo que se obtuvo es que nuestro sistema es dinámicamente inestable y por eso no es posible obtener un gramiano, pero teóricamente se sabe que la variable que aporta mayor información al sistema es el ángulo.

Método de control

El método de control que se eligió para el robot balancín fue un PID, debido a su facilidad de implementación, ahora para la obtención de los valores de Kp, Ki y Kd, se hizo utilizando el algoritmo de cálculo de un PID con feedback, y con en matlab haciendo uso de un PID tune, primeramente se obtuvo la función de transferencia del sistema, siendo la misma:

$$\text{tf} = \frac{2s^3 + 0.01946s^2 - 161s - 1.348}{s^4 + 0.009731s^3 - 172.1s^2 - 1.348s} \quad (21)$$

Con esta función de transferencia se hizo uso de un PID en matlab obteniendo la siguiente grafica:

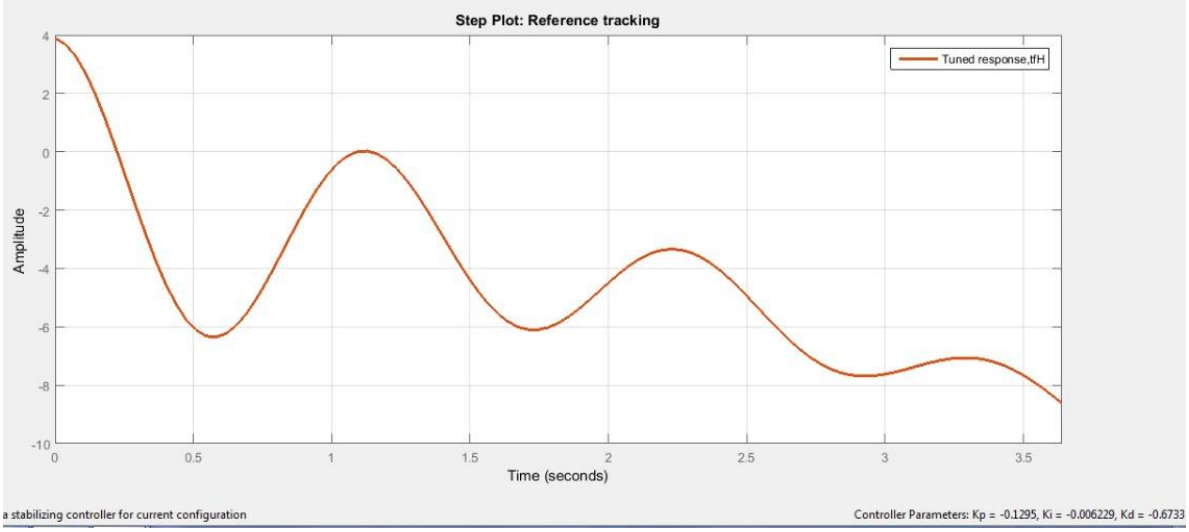

Figura 3.13. Implementación de pid (figura propia)

Los datos teoricos de pid serian:

Kp= -0.1295

Ki= -0.006229

Kd= -0.6733

Como se observa el tiempo de asentamiento es bastante grande, pero fue el mejor dato obtenido de PID, obviamente el sistema no se controlaba óptimamente con estos datos, pero la lectura del sensor era bastante limpia, los resultados finales de pid, después de cambiar los resultados teóricos a base de prueba y error son los siguientes:

Kp= 3

Ki= -10.5

Kd= 0.15

Con estos resultados se logro controlar el robot balancín.

4. Resultados y conclusiones

El resultado final fue un robot correctamente controlado, aunque se esperaba que sea más resistente a perturbaciones externas, también se puede concluir que la estructura mecánica fue vital en el control, la primera estructura que se obtuvo era demasiado débil y los motores no estaban seguros, haciendo imposible su control, la segunda era demasiado robusta, y debido al mal cálculo había demasiado rozamiento en los motores, luego se llego a tener una estructura buena con la cual se realizo un buen control, el manejo del MPU6050 pese a que es un sensor muy preciso y útil, llega a ser un poco tedioso debido a que a veces falla en su comunicación o deja de funcionar sin razón aparente, la conversión de las medidas de pitch a grados es necesaria para su funcionamiento, las mejoras que se pueden hacer a este robot, son:

- debido a que los motores consumen demasiado rápido la batería de 9v, se deberían poner 2 baterías en paralelo o ver la forma de alimenta continuamente el motor sin quitarle la independencia al sistema.

- mejorar la implementación del PID, debido a que el robot balancín se controla, pero no se es resistente a perturbaciones externas muy bruscas, se buscara mejorar el PID que se propone en este proyecto.

- probar un controlador LQR y/o LQG para ver si el sistema se controla mejor que con un PID.

- utilizar un filtro Kalman para el sensor o cualquier tipo de estimador para mejorar la lectura de nuestro sensor (que ya es bastante buena en si), mientras mas limpios sean los datos leidos, mejor funcionara el robot.

Se concluye que el proyecto fue realizado correctamente debido a que se logro controlar el sistema propuesto y se espera lograr implementar las mejoras planteadas para el futuro.

5.- CALENDARIO.-

En esta parte se describirá las fechas del avance del proyecto. Cabe aclarar que estas son fechas aproximadas es posible que no sean precisas del todo debido a que todas las fechas descritas, excepto las revisiones en clase, por distintas causas pueden haber sido otras o son las correctas.

Tabla.- Calendario Tentativo.

Fechas	Avance del proyecto
2/9/17	Primera investigación de conocimientos necesarios
10/9/17	Adquisición de materiales necesarios
15/9/17	Cálculos preliminares de movimiento y de control
1/10/17	Modelado del sistema mecánico
2/10/17	Inicio del armado del sistema mecánico
5/10/17	Primera revisión de avance en clase
17/10/17	Segunda etapa de investigación.
20/10/17	Arreglos pertinentes a partir de la primera revisión
25/10/17	Finalización del sistema mecánico
28/10/2017	Primera prueba del sistema
29/10/2017	Cambios en sistema basados en la primera prueba
1/11/2017	Segunda prueba del sistema
29/10/17 - 1/11/17	Cambios en sistema basados en la segunda prueba
16/11/2017	Segunda revisión de avance en clase
20/11/2017	Tercera etapa de investigación

21/11/2017	Inicio de pruebas con el modelado correcto
22/11/2017	Prueba de PID con el modelado correcto
27/11/2017	Arreglos en la estructura mecanica
30/11/2017	Reconstrucción de la estructura mecánica (debido a accidente)
5/12/2017	Entrega del proyecto terminado

En esta tabla se muestran las fechas del avance del proyecto, en la columna de la izquierda se muestran las fechas y en la columna de la derecha se muestra una breve descripción del avance del proyecto.

6 ANEXOS

En esta parte se incluirá el código usado para controlar el robot

```
#include "I2Cdev.h"

#include "MPU6050.h"

#include "Wire.h"

#include "Follower.h"

MPU6050 accelgyro;

Motors robot(5, 8, 7, 6, 10,9);


// para el pid

uint16_t ts = 5;

 long lastTime;

float setpoint = 0;

float input;

float output;


// parametros

const float kp =  10;

const float ki = 50;

const float kd = 0;


//----------

pid controladorInclinacion(&input, &output, &setpoint, kp, ki, kd);
```

```
///----------------



int16_t ax, ay, az;
int16_t gx, gy, gz;

float inclinacion;
#define LED_PIN 13
bool blinkState = false;

void setup() {
// para el mpu
    Wire.begin();

    Serial.begin(38400);

    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");

    pinMode(LED_PIN, OUTPUT);

    // set sample time
    controladorInclinacion.SetSampleTime(ts);
    lastTime = millis();
}

void loop() {
  int accum=0;
  for(int i = 0; i < 10; i++)
  {
    // lectura del sensor
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
    accum += 28.64789 * atan2(az, ay) * 2 - 90.0;
```

```
  }
  inclinacion = accum / 10;

  if((millis() - lastTime) > ts)
  {
    lastTime = millis();
    // calculo de la inclinacion en grados
    //inclinacion = 28.64789 * atan2(az, ay) * 2 - 90.0;

    input = inclinacion;

    int salida = controladorInclinacion.Compute();

   // Serial.println(input);
    Serial.print("--");
    Serial.println(salida);

    robot.drive(salida);
    // blink LED to indicate activity
    blinkState = !blinkState;
    digitalWrite(LED_PIN, blinkState);

  }
 }
```

Para el PID

```
#include "Follower.h"

pid::pid(float * input, float * output, float * setpoint,
            float inKp, float inKi, float inKd)
{
            userInput = input;
            userOutput = output;
            userSetpoint = setpoint;
            SetLimits(-255, 255);
```

```
		sampleTime = 100;
		SetParameters(inKp, inKi, inKd);
		Init();
}
int16_t pid::Compute()
{
		float input = *userInput;
		float error = *userSetpoint - input;
		ITerm  += (ki * error);
		if(ITerm > max_out) ITerm = max_out;
		else if(ITerm < min_out) ITerm = min_out;
		float dInput = (input - lastInput);

		float output = kp * error + ITerm - kd * dInput;
		if(output > max_out) output = max_out;
		else if(output < min_out) output = min_out;
		*userOutput = output;
		lastInput = input;
		return (int16_t)output;
}
void pid::SetLimits(float nMin, float nMax)
{
		min_out = nMin;
		max_out = nMax;
}
void pid::SetParameters(float nKp, float nKi, float nKd)
{
		kp = nKp;
		ki = nKi * (sampleTime / 1000.0);
		kd = nKd / (sampleTime / 1000.0);
}

void pid::SetSampleTime(uint16_t ntime)
{
```

```
        if(ntime == 0) return;
        float ratio = (float)ntime / (float)sampleTime;
        ki *= ratio;
        kd /= ratio;
        sampleTime = ntime;
}

void pid::Init()
{
        ITerm = *userOutput;
        lastInput = *userInput;
}
```

6.- REFERENCIAS

[1]. Wikipedia, Furuta pendulum https://en.wikipedia.org/wiki/Furuta_pendulum

[2]. Wikipedia, Movimiento oscilatorio https://es.wikipedia.org/wiki/Movimiento_oscilatorio

[3]. Wikipedia, Péndulo https://es.wikipedia.org/wiki/P%C3%A9ndulo

[4]. Física con ordenador, Péndulo Físico http://www.sc.ehu.es/sbweb/fisica/solido/pendulo/pendulo.htm

[5]. Física con ordenador, Péndulo simple http://www.sc.ehu.es/sbweb/fisica/dinamica/trabajo/pendulo/pendulo.htm

[6]. Levantamiento y estabilización del péndulo invertido http://www.ctrl.cinvestav.mx/~fcastanos/mios/bachelorCastanos.pdf febrero 3, 2003

[7]. LBA, Que es un encoder http://www.abm-industrial.com/2013/02/07/que-es-un-encoder/

[8]. Franklin Electronic, Que es el control PID https://franklinelinkmx.wordpress.com/2013/09/05/que-es-el-control-pid/ septiembre 5, 2013

[9]. Javier Louriero, como usar un encoder con arduino https://javierloureiro.wordpress.com/2014/04/24/como-usar-un-encoder-con-un-arduino/ abril 24, 2014

[10]. Arduino, Que es el Arduino http://arduino.cl/que-es-arduino/

[11]. Wikipedia, Linear-quadratic regulator
https://en.wikipedia.org/wiki/Linear%E2%80%93quadratic_regulator

[12]. Agarcia, Respuesta en el tiempo de un sistemas de control
http://agarcia.fime.uanl.mx/materias/ingco/apclas/05%20-%20Respuesta%20en%20el%20Tiempo%20de%20un%20Sistema%20de%20Control.pdf

[13]. MPU6050
https://playground.arduino.cc/Main/MPU-6050

[14]. Figura Arduino uno.
https://store.arduino.cc/usa/arduino-uno-rev3

[15]. Figura mpu6050
http://www.dx.com/p/gy-521-mpu6050-3-axis-acceleration-gyroscope-6dof-module-blue-154602#.Wd-BgGiCzIU

[16]. Figura l298n
http://www.dx.com/p/l298n-dual-h-bridge-stepper-motor-driver-module-w-heat-dissipation-for-arduino-red-white-408436#.Wd-BYWiCzIU

[17]. Figura motor DC
http://www.baudaeletronica.com.br/kit-motor-dc-roda-para-robo.html

[18]. Obtención de datos del motor
http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=6E03B73A00D44253C14622F7C7DC05C9?doi=10.1.1.186.7096&rep=rep1&type=pdf