ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"IMPLEMENTACIÓN DE UN ESTIMADOR LINEAL PARA EL CONTROL DE UN ROBOT AUTOBALANCEADO"

TRABAJO DE TITULACIÓN

Previo a la obtención del Título de:

MAGISTER EN AUTOMATIZACIÓN Y CONTROL INDUSTRIAL

MARÍA BELÉN GUANO CARRILLO

PAUL GEOVANNY AMÉN MORA

GUAYAQUIL – ECUADOR

AÑO: 2020

AGRADECIMIENTO

Mi agradecimiento a Dios por sus bendiciones y amor.

Agradezco a la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación por la beca otorgada, gracias la cual fue posible realizar esta maestría.

Agradezco a todos mis familiares quienes me han apoyado en este proceso tanto con sus ánimos, cariño, cuidado de mis hijos mientras estuve estudiando y viajando. A mi madre, hermana y esposo.

Mi agradecimiento a los excelentes profesores de la MACI por su calidad humana y profesional. A nuestro director de proyecto MSc. Carlos Salazar.

Belén

AGRADECIMIENTO

Agradezco a la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación por la beca otorgada.
Agradezco a mi mamá y a todas las personas que me han ayudado en este proceso.

Paúl

DEDICATORIA

El presente proyecto está dedicado a mi ángel en el cielo mi padre Felipe Guano quien siempre me alentó al camino de la ciencia haciéndome sentir lo orgulloso que estaba de mí.

Belén

TRIBUNAL DE EVALUACIÓN

Ph.D. Juan Avilés Castillo
SUBDECANO DE LA FIEC

Ms.C. Carlos Salazar López
DIRECTOR DEL TRABAJO DE TITULACIÓN

Ph.D. Douglas Plaza Guingla
MIEMBRO PRINCIPAL DEL TRIBUNAL

DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, nos corresponde exclusivamente; y damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

.....

María Belén Guano Carrillo

.....

Paúl Geovanny Amén Mora

RESUMEN

El presente trabajo trata del proceso desarrollado para implementar un estimador lineal el cual permitirá controlar un robot autobalanceado. El observador con el que se trabaja es el Filtro de Kalman, encargado de eliminar el ruido presente en el acelerómetro y giroscopio, los cuales son encargados de medir el valor del ángulo de inclinación necesario para el control de equilibrio en el robot autobalanceado péndulo invertido. Este tipo de robot es de amplio uso tanto en la investigación como industria, siendo más conocido el vehículo SEGWAY por su uso para transportar a una persona parada en su plataforma mientras se desplaza a la vez que controla el equilibrio.

En el primer capítulo se presenta los motivos para desarrollar este proyecto, el mecanismo de trabajo a aplicar y el alcance el mismo.

En el segundo capítulo se lleva a cabo una introducción al fundamento teórico de los dispositivos usados para el robot autobalanceado, así como la teoría detrás del Filtro de Kalman.

En el tercer capítulo se explica el proceso llevado a cabo para ensamblar el robot, implementar el estimador lineal en el sistema embebido y así lograr el control del robot.

En el capítulo cuatro el funcionamiento y desempeño del robot son presentados.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

CAPÍTULO 1

1. PLANTEAMIENTO DEL PROBLEMA.

1.1 Identificación del problema.

El robot autobalanceado ha sido objeto de amplio estudio tanto en el campo de la robótica, modelamiento de sistemas, sistemas de control. Debido a la dinámica inestable del sistema. La característica principal de este robot es la capacidad de mantenerse equilibrado sobre dos ruedas, incluso cuando cuenta con un chasis pesado o alto que lo convierte en una aplicación del péndulo invertido. Existen múltiples aplicaciones del péndulo invertido móvil como en el campo de la robótica donde basados en la semejanza del movimiento de los pies al caminar con el péndulo invertido permitió modelar y diseñar caminatas en robots humanoides [1]. La aplicación más conocida de este robot es el vehículo eléctrico SEGWAY, se trata de un péndulo invertido de gran tamaño que se mantiene equilibrado mientras traslada a personas.

La investigación del robot autobalanceado en el campo del control se debe a lo inestable del sistema, haciéndolo útil e interesante para el diseño de controladores, modelamiento. Para su funcionamiento este robot hace el uso de sensores como el giroscopio y acelerómetro, los cuales presentan ruido y desviación en la medida, convirtiéndolo en una plataforma para la aplicación del filtro de Kalman, necesario para la fusión de los sensores indicados, este filtro permite compensar los errores de ruido y deriva para proporcionar información de movimiento más completa y precisa.

1.2 Justificación.

La ejecución de esta investigación permitirá implementar un robot autobalanceado, iniciando por determinar los dispositivos necesarios para su funcionamiento, seguido de la ejecución de un programa computacional en la tarjeta escogida como controlador principal donde se llevará a cabo el control del autómata.

Para el diseño del control de equilibrio será necesario adquirir un buen conocimiento del sistema, para lo cual el modelo del robot será simulado, con la finalidad de conocer su respuesta en el tiempo, lugar geométrico de las raíces. Esto servirá como base para la implementación del control y determinación de las ganancias de este.

En el caso de los sensores necesarios para el manejo del robot como el giroscopio el cual mide la orientación en el espacio y el acelerómetro encargado de medir aceleraciones asociadas con el fenómeno de peso experimentado por una masa localizada en el marco de referencia del dispositivo, el ruido presente en el acelerómetro y deriva del giroscopio (desviación de la medida real que se acumula con el paso del tiempo) representan un problema para obtener buenas lecturas de los ángulos como el de inclinación tan importante para balancear el sistema. Con el fin de sortear esta dificultad se diseñará el estimador lineal denominado filtro de Kalman, donde el modelamiento de sistemas para la obtención de las ecuaciones que definen el modelo de medición del sistema será aplicado. Desarrollando estas actividades se obtendrá nuevo conocimiento y destreza en el campo de modelamiento y estimación de estados, pues el filtro de Kalman es de mucha importancia en aplicaciones tecnológicas como la guía, navegación y control de vehículos, especialmente naves espaciales, procesamiento de señales, sistemas de comunicaciones. Como el control se ha extendido no solo a sistemas industriales o tecnológicos, se puede observar que es posible el uso de estos recursos de estimación en campos como la economía, biología, abriendo un abanico de opciones en el cual podemos trabajar y aportar.

1.3 Solución propuesta.

El presente trabajo propone desarrollar el algoritmo del filtro de Kalman para obtener el valor del ángulo de inclinación del robot medido por el giroscopio y acelerómetro fusionados en un mismo dispositivo, libre de ruido y desviación. Para la puesta en marcha de este estimador el modelo matemático de medición será encontrado, mediante un proceso iterativo aplicado a las ecuaciones del filtro de Kalman se eliminará el error cuadrático de las mediciones [2].

Una vez obtenido el ángulo de inclinación libre de ruido proveniente de los sensores, el cual es la variable para supervisar por el control de equilibrio, un compensador Proporcional Integral Derivativo PID será implementado.

El autómata contará con un sistema de comunicación vía bluetooth con el software Visual Basic permitiendo adquirir datos de sensores para supervisar, graficar y enviar datos al robot. Recibirá mediante esta comunicación direcciones enviadas mediante Visual Basic para seguir una trayectoria determinada la cual puede ser de avance en línea recta, retroceso o giros derecha e izquierda, lo cual constituye un reto pues se deberá hacer un buen control para coordinar los compensadores de equilibrio y trayectoria.

El control de trayectoria que permite el avance o retroceso del robot hará uso de los encoder de las llantas para supervisar la velocidad angular y el control de giro que permita avanzar a la derecha o izquierda toma como entrada el ángulo del giroscopio en el eje z. Estos dos controles estarán basados en un PID

1.4 Objetivo de la tesis.

1.4.1 Objetivos generales.

Implementar el estimador lineal Filtro de Kalman para controlar un robot autobalanceado.

1.4.2 Objetivos específicos.

- Investigar las ecuaciones, funcionamiento del algoritmo del filtro de Kalman y proponer un esquema de implementación.
- Desarrollar el algoritmo mencionado en una tarjeta Arduino Uno, el cual sea capaz de filtrar, identificar el valor preciso del giroscopio y acelerómetro fusionados, para obtener la posición.
- Diseñar e implementar un control PID aplicado al robot para alcanzar el punto de equilibrio deseado y seguimiento de trayectoria.
- Establecer comunicación con el robot por medio de bluetooth con el software Visual Basic, mismo que permitirá adquirir datos de sensores para supervisar, graficar y enviar datos al robot para seguir una

trayectoria determinada que actuará como perturbación en la tarea principal de control de equilibrio.

1.5 Metodología.

Previo al desarrollo del experimento se realizará una investigación a fondo de la bibliografía respectiva para fortalecer conocimientos, adquirir nuevos y poder establecer el procedimiento a seguir.

En el avance del proyecto se aplicarán varias metodologías aprendidas en las materias a lo largo de la maestría. Para la implementación del filtro de Kalman el modelamiento de sistemas será de vital importancia pues así se logrará entender y encontrar las ecuaciones de estado que definen el sistema de medición.

Se propondrá un esquema de control basado en un PID, el mismo que será de ayuda para lograr la estabilidad del robot empleado.

Para el seguimiento de trayectoria y giro que actúan como perturbación al robot se desarrollará una comunicación bluetooth con el software Visual Basic, donde se pondrá en práctica conocimientos de programación avanzada.

1.6 Alcance.

El alcance de este proyecto queda delimitado hasta obtener el objetivo marcado, equilibrar un robot autobalanceado (ángulo de inclinación cero) mediante un control PID, que tiene como entrada la lectura de los sensores en este caso el valor de la posición angular del robot con menor error, calculado a partir de la aplicación del filtro de Kalman. El desempeño del controlador y funcionamiento serán probados mediante la aplicación de perturbaciones y gráficas de los datos obtenidos.

CAPÍTULO 2

2. FUNDAMENTO TEÓRICO.

2.1 Robots

Los robots han llamado la atención del mundo desde la aparición de los modelos más básicos, con el pasar de los años el desarrollo de investigación ha avanzado de tal manera que ahora se cuentan con robots aplicados en los más diversos campos desde la medicina, exploración espacial, ayuda a personas con capacidades diferentes o en la industria. Todos estos cambios se dan debido al gran avance en el campo computacional, electrónica, sensores, actuadores entre otros.

El término robot aparece por primera vez en 1921 en una obra teatral del novelista y autor Karel Capek en cuyo idioma la palabra “robota” significa fuerza de trabajo o servidumbre [3]. En ese tiempo esta denominación tuvo aceptación y así es como se relacionó a la aparición de estos dispositivos que empezaban a ser ensamblados cumpliendo funciones básicas. Es necesario destacar que los robots fueron desde sus inicios asociados al trabajo y producción. Con el avance del tiempo esta idea no ha cambiado, pues el enfoque en la mejora de sus prestaciones es mejorar la producción en empresas, disminuir la mano de obra humana.

El uso de estos autómatas en diferentes sectores actualmente se ha ampliado, así como la clasificación de acuerdo con la arquitectura, pese a esto el esquema de un robot en general se puede definir de la siguiente manera:

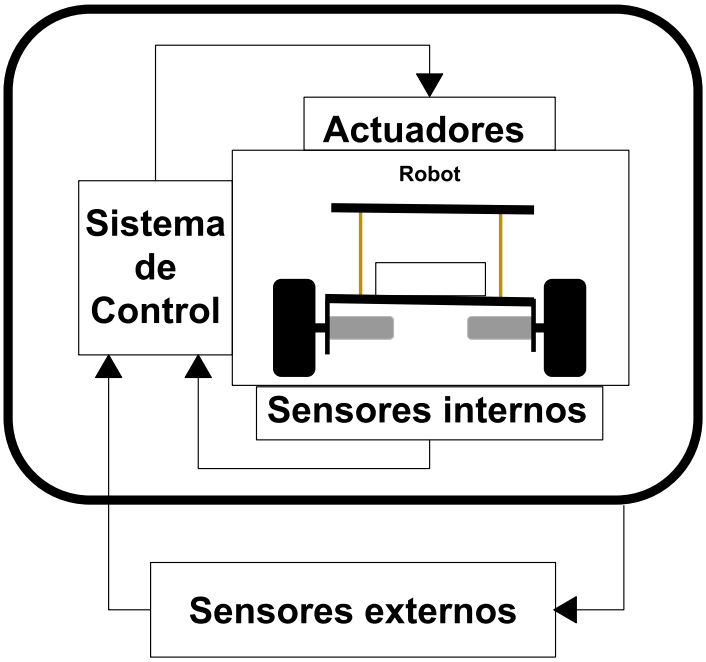


Figura 2.1: Esquema básico de un robot.

Los sensores internos permiten supervisar giros, desplazamientos de articulaciones, velocidades, torque, etc. Hacen posible también cerrar lazos de control en la estructura mecánica.

Los sensores externos dotan al robot de la capacidad de percibir, aprender del entorno, enfocado en la toma de decisiones generando automáticamente acciones en función de comparar la información sensorial con patrones de referencia. La mejora de la percepción sensorial ha sido posible gracias al avance de la tecnología en los sensores.

Los sistemas de control también han presentado mejoras notables gracias a la mejora en la tecnología, como las altas velocidades de procesamiento de datos, alta capacidad de memoria, nuevos algoritmos como las redes neuronales. El control se encuentra enfocado a la acción de compensación en respuesta a la entrada de los sensores.

Los actuadores son los encargados de reflejar las acciones tomadas por el sistema de control, pueden ser brazos mecánicos, motores en el caso de los robots móviles.

Existen múltiples clasificaciones de los robots, por su estructura y función son los más usados en la literatura. Por su función pueden ser robots industriales, médicos, de servicios domésticos, militares, educativos, de servicios de exploración.

En la clasificación por arquitectura constan los robots manipuladores y móviles. Los manipuladores hacen referencia especialmente a los brazos articulados más usados en la industria, capaces de trasladar objetos, llegar a coordenadas determinadas de manera muy precisa. Los robots móviles son aquellos dotados de sensores para controlar el movimiento, velocidad, orientación y desplazamiento, el cual idealmente debería ser autónomo, entre estos también existe variedad de tipos como exploradores, seguidores de trayectoria determinada, entre otros.

Robot Autobalanceado

Dentro de las organizaciones de los robots mencionadas, se encuentra el autobalanceado, el cual es un robot móvil de péndulo invertido de dos ruedas.

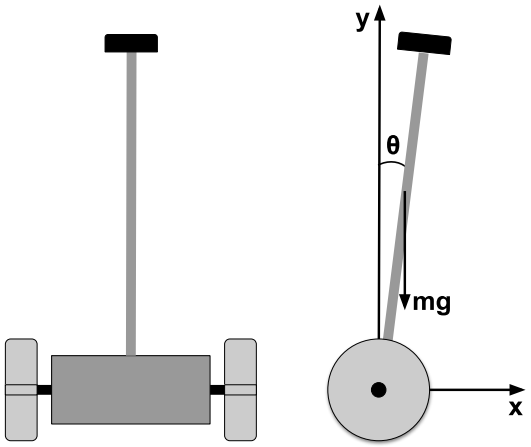


Figura 2.2: Vista frontal y lateral de un robot péndulo invertido de dos ruedas [19].

Para controlar este autómata se requiere de un control muy preciso, pues es necesario equilibrar el péndulo en un ángulo de inclinación cero (θ=0°) como ilustra la figura 2.2, en caso de que el vehículo se mueva en alguna dirección lo debe hacer con cierta velocidad y manteniendo el equilibrio.

Esta es una plataforma móvil de tracción diferencial pues cuenta con dos ruedas acopladas a motores DC, mismas que al moverse en un mismo sentido harán al vehículo desplazarse en línea recta, al hacer un movimiento independiente de las ruedas se produce la rotación sobre el mismo eje.

El chasis sostiene todos los dispositivos electrónicos y mecánicos del sistema como los motores, sensores, baterías y circuitos.

La parte del sistema que causa inestabilidad es el péndulo que debe ser estabilizado, generalmente forma parte de la estructura del chasis.

2.2 Sensores

Los sensores son fundamentales para el desarrollo de la automatización y robótica, pues para realizar una acción es necesario conocer el valor de una variable física.

Para su funcionamiento los sensores se basan en el llamado principio de traducción, el cual cambia la variable física medida a una señal eléctrica, presión, movimiento, etc, para ser utilizada por un sistema de procesamiento de la información.

Para llevar a cabo su función un transductor se vale de algún principio físico de transformación de energía al que se denomina principio de transducción, pueden ser por ejemplo piezoresistivo, capacitivo, ultrasónico, magnético, resistivo entre otros [4].

Las variables físicas que pueden ser medidas son innumerables, en este caso nos enfocaremos en la velocidad y aceleración angular las cuales pueden ser medidas con la ayuda de los siguientes sensores:

2.2.1 Acelerómetro

Este tipo de sensores es ampliamente utilizado para determinar tanto la inclinación de un objeto como su vibración, cuando se utiliza en configuración estática. Para el caso de aplicaciones dinámicas, estos sensores se usan para determinar la aceleración traslacional. Este tipo de sensores pueden estar basados en el principio de transducción piezoresistivo, piezoeléctrico o capacitivo; sin importar qué tipo de principio de transducción se utilice, el sensor tendrá una salida lineal, esto quiere decir que al impulso o inclinación que reciban a la entrada, se observará una respuesta proporcional a la salida.

Un acelerómetro lineal es un sensor inercial que mide la diferencia total entre la aceleración traslacional y la componente de la aceleración gravitatoria a lo largo de su eje de acción.

Como se había mencionado en condiciones estáticas estos sensores pueden ser usados como inclinómetros, son capaces de medir la aceleración en los ejes (x,y,z). La inclinación del sistema se mide a partir de relacionar la aceleración debida a la gravedad de la Tierra y el desplazamiento de la masa móvil. Para obtener el valor de inclinación en estado de reposo se utilizan las expresiones (2.1), (2.2) sobre cada eje, basadas en los ángulos de Euler representados en la Figura 2.3.

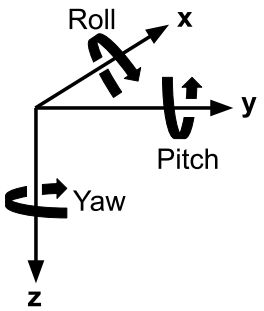


Figura 2.3: Sistemas de coordenadas y ángulos de Euler.

$$Roll = arctan\left(\frac{A_y}{A_z}\right) \quad (2.1)$$

$$Pitch = arctan\left(\frac{-A_x}{\sqrt{A_y^2 + A_z^2}}\right) \quad (2.2)$$

Donde:

Ax, Ay, Az = componentes de la gravedad medidas sobre el eje x, y, z del acelerómetro.

2.2.2 Giroscopio

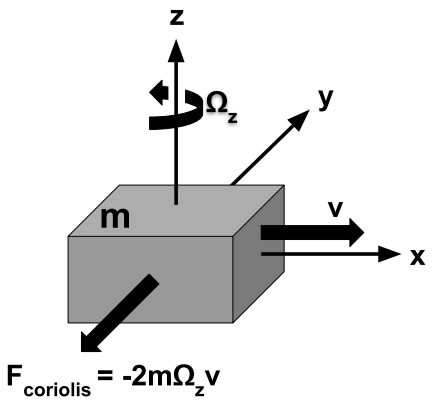


Figura 2.4: Representación de la fuerza de Coriolis [4].

El giroscopio ha sido comúnmente utilizado como un instrumento de medición angular, permite relacionar la rotación relativa con un voltaje. Estos sensores, a pesar de ser excitados por una fuerza inercial, aprovechan los efectos de las fuerzas de Coriolis (figura 2.4) presentes en un movimiento rotacional, para obtener la velocidad angular. La fuerza de Coriolis se muestra en (2.3).

$$\vec{F_c} = -2m\Omega_z v \quad (2.3)$$

Donde:

m: es la masa del cuerpo.

Ω_z: es la velocidad angular respecto al eje que rota.

v: es la velocidad lineal.

La posición angular con respecto al ángulo inicial es calculada integrando en el tiempo la velocidad angular dada por el sensor. Se debe tomar en cuenta la variación de ángulos en intervalos cortos de tiempo como se observa en (2.4).

$$\frac{d\theta}{dt} = \omega \tag{2.4}$$

Donde:

ω: velocidad angular [rad/sg]

θ: ángulo de rotación

dt: tiempo de muestreo

La ecuación (2.4) es válida para instantes definidos de tiempo, no indica el ángulo total de giro sobre un eje en un tiempo "t". Es necesario realizar un algoritmo de carácter acumulativo en tiempo discreto con el fin de conocer el ángulo total alcanzado en un rango de tiempo $t_0 \rightarrow t_f$ como muestra 2.5.

$$\theta_k = \theta_{k-1} + \omega_k * dt \qquad k = 0,1,2 \ldots \tag{2.5}$$

Donde:

θ_k: ángulo actual.

θ_{k-1}: ángulo anterior.

ω_k: velocidad angular actual.

El giroscopio presenta el fenómeno de la precesión que, en conjunto con la presencia de fricción en los ejes y cojinetes, de errores de fabricación, recursividad de la ecuación 2.5 provoca que la desviación del valor real en la medida del ángulo tenga una "deriva", es decir un pequeño error que se va acumulando [5].

Los giroscopios, como casi todos los sensores con sistemas microelectromecánicos calibrados por el fabricante antes de ser

encapsulados; de esa manera, en la hoja de datos del sensor aparecerá un valor de voltaje en estado de reposo y una sensitividad.

2.2.3 MPU 6050

El MPU-6050 es un dispositivo integrado de seguimiento de movimiento de 6 ejes, combina un giroscopio de 3 ejes, un acelerómetro de 3 ejes y un procesador de movimiento digital, todo en un pequeño paquete de 4x4x0.9mm. Con su bus dedicado I2C [6].

La figura 2.5 muestra la dirección de los ejes de la tarjeta MPU 6050, que se debe considerar para su correcto posicionamiento.

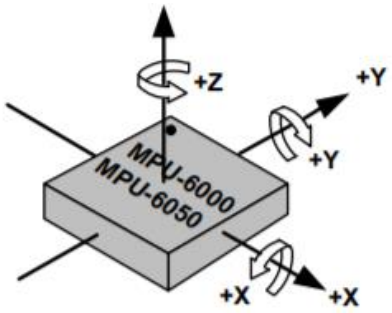


Figura 2.5: Orientación de ejes de sensibilidad y polaridad de rotación.

Características principales:

Voltaje de alimentación: 3-5 V

Rango del giroscopio: 250/500/1000/2000 ◦/s

Rango del acelerómetro: 2/4/8/16 g

2.2.4 Encoder

Son los sensores más utilizados para la medición de sistemas rotacionales, capaces de convertir el movimiento rotacional o lineal en una señal digital equivalente que puede ser usado para realizar control, ser procesada. Son usados para conocer la velocidad angular, posición, dirección de giro.

Existen varias formas como los encoder son implementados, entre ellos:

Encoder Incremental. Hace uso de uso de dos pares de fotodiodo y fototransistor, desfasados 90° entre ellos, de esta manera existe un desfase entre las señales, al girar en sentido horario la señal del primer par esta adelantada 90° con respecto a la segunda, en caso contrario en el sentido antihorario la señal del segundo par estará adelantada 270°. Con este sensor se puede contar el número de vueltas, obtener la posición, cuando se pierde la alimentación no es posible saber la última posición conocida.

Encoder Ópticos. Usa un par fotodiodo y fototransistor, entre ellos un disco ranurado acoplado mecánicamente al eje, este disco está compuesto cierto número de ranuras, al girar el disco a través del emisor/receptor se produce una señal cuya frecuencia determina la velocidad del eje.

Encoder Absoluto. En este tipo de encoder el disco tiene varias bandas situadas de forma concéntrica codificadas en código grey, cada una de estas bandas posee un par emisor/receptor que generará un código correspondiente a la posición. La información de la posición al perder la alimentación puede ser conocida.

Encoder Magnético. Se usan para medir velocidad y posición. Hacen uso de un sensor de efecto hall para la detección o medición de campos magnéticos. Existen varias configuraciones utilizadas para colocar el imán respecto del sensor, el resultado obtenido será el mismo en todos los casos, cada vez que el sensor de efecto hall detecte la presencia del imán se produce un cambio de estado en la señal generada, la frecuencia de esta señal determina la velocidad del eje.

2.2.5 Motor con Encoder JGB37 520B

Este motorreductor ilustrado en la figura 2.6, utiliza un conjunto de engranajes para convertir la alta velocidad original y el bajo torque del motor en baja velocidad y alto torque. Al eje del motor se encuentra acoplado un encoder magnético. Este tipo de motores son de amplio uso

en el campo de la robótica educativa ya que proporcionan potencia, alto torque y bajo ruido.

Figura 2.6: Motor con encoder JGB37 520B.

Características principales:

Torque: 0.1 - 30 Kg/cm

Velocidad: 12 – 1600 RPM

Corriente: 300 – 400 mA

Volteje Nominal: 12V

2.3 Sistemas Embebidos

Es un circuito electrónico digital capaz de realizar operaciones de computación, generalmente en tiempo real, que sirven para cumplir una tarea específica en un producto [7].

Los componentes de un sistema embebido son:

La parte central donde se encuentra el microprocesador capaz de procesar instrucciones a una velocidad dada por el reloj del sistema para poder comunicarse con los periféricos.

Unidad de memoria donde se almacenan programas y datos. Algunos casos pueden tener memoria interna o externa.

Unidades de entrada/salida, soportan el conexionado de los sensores y actuadores del dispositivo a controlar.

Se pueden programar directamente en el lenguaje ensamblador del microcontrolador incorporado, utilizando compiladores específicos, o también, pueden utilizarse lenguajes como C o C++.

Otra característica importante es el bajo consumo de energía, necesario para muchas aplicaciones que hacen uso de baterías.

Tarjeta Arduino Uno

Es una placa basada en el microcontrolador ATmega328P, cuenta con 14 pines digitales de los cuales 6 pueden ser usados como salidas PWM y otros 6 pines analógicos, posee una velocidad de reloj interna de 16 MHz. Para su programación cuenta con su propio entorno de desarrollo integrado que puede ser descargado de forma gratuita o con una contribución voluntaria a través de la página web oficial de Arduino.

Es de hardware libre y debido a su bajo costo y poseer una gran comunidad de desarrolladores se vuelve accesible iniciar cualquier tipo de proyecto desde una simple aplicación hasta un proyecto científico.

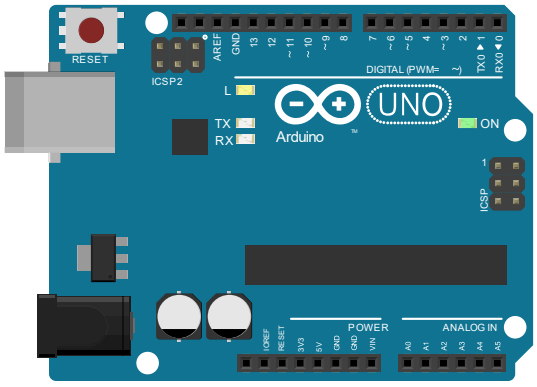


Figura 2.7: Tarjeta Arduino Uno

2.4 Comunicación Inalámbrica con Bluetooth

Es una comunicación para corta y media distancia, caracterizada por bajo costo y menor consumo. De gran uso para redes pequeñas o de captación de información.

La tecnología inalámbrica Bluetooth utiliza la banda de radio ISM (industrial, científica, médica) mundialmente disponible, de 2.4GHz. Las bandas ISM incluyen los rangos de frecuencias entre 902-928 MHz y 2.4 - 2.484GHz. 48 que no requieren una licencia de operador otorgada por las autoridades reguladoras de telecomunicaciones [8].

Tiene una comunicación maestro/esclavo, soporta hasta 8 dispositivos. Es un enlace robusto, no susceptible al deterioro por señales electromagnéticas e interferencias. El estándar bluetooth fue creado para ser usado por un gran número de fabricantes, permitiendo la conexión de todos los dispositivos que utilicen esta tecnología, para esto se crearon perfiles de protocolo y modelos de usuario.

Módulo Bluetooth HC 05

Es un módulo Bluetooth SPP (protocolo de puerto serie), diseñado para configuración de conexión en serie inalámbrica transparente. Cumple con las especificaciones del estándar Bluetooth 2.0. Puede ser usado como maestro o esclavo dependiendo de la configuración. Utiliza CSR Bluecore 04, sistema bluetooth externo de un solo chip con tecnología CMOS.

Figura 2.8: Módulo bluetooth HC 05

Características principales:

Voltaje de operación: 3.6V - 6V DC

Consumo corriente: 50mA

Frecuencia: Banda ISM 2.4GHz

Alcance 10 metros

Velocidad de transmisión: 1200 - 1.3Mbps

Baudrate por defecto: 38400,8,1,n.

2.5 Representación en espacio de estados

En la mayoría de las representaciones de modelos matemáticos el ruido es obviado para simplificar su tratamiento, pero no podemos dejar de lado que existe está presente estático en la naturaleza, de tal manera, que si queremos modelarlo debemos hacer uso de métodos estocásticos.

A continuación, se explica la representación en modelo de espacio de estados de un proceso dinámico con ruido.

La siguiente ecuación diferencial de orden n-ésimo describe un proceso dinámico autoregresivo (AR).

$$\boldsymbol{y}_{i+1} = a_{0,i}\boldsymbol{y}_i + \cdots + a_{n-1,i}\boldsymbol{y}_{i-n+1} + \boldsymbol{w}_i, \qquad i \geq 0 \tag{2.6}$$

Donde w_i es el ruido blanco aleatorio del proceso con media cero y su autocorrelación está representada de la forma:

$$E\left[w_i, w_j{}^*\right] = R_w = Q_i\delta_{ij} \tag{2.7}$$

Donde δ_{ij} es la función delta Kronecker de valor cero, excepto cuando $i = j$.

Los valores iniciales $\{y_0, y_{-1}, \dots, y_{-n+1}\}$ son variables aleatorias con media cero, con una matriz de covarianza P_0 de orden $n\ x\ n$.

$$P_0 = E\left[y_{(-j)}, y_{(-k)}\right], j, k \in 0, n-1 \tag{2.8}$$

Se asume que:

$$E[w_i, y_i{}^*] = 0, \forall -n+1 \leq j \leq 0 \cap i \geq 0 \tag{2.9}$$

Como asegura [9]:

$$E\left[w_i, y_j{}^*\right] = 0, i \geq j \geq 0 \tag{2.10}$$

Lo que quiere decir que el ruido es estadísticamente independiente del proceso a estimar.

La ecuación diferencial 2.6 puede ser reescrita de la siguiente manera

$$\mathbf{x}_{i+1} \triangleq \begin{pmatrix} y_{i+1} \\ y_i \\ y_{i-1} \\ \vdots \\ y_{i-n+2} \end{pmatrix} = \underbrace{\begin{pmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} y_i \\ y_{i-1} \\ y_{i-2} \\ \vdots \\ y_{i-n+1} \end{pmatrix}}_{\mathbf{x}_i} + \underbrace{\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{G} \boldsymbol{w}_i$$

Lo cual describe el espacio de estados:

$$\mathbf{x}_{i+1} = A\mathbf{x}_i + G\boldsymbol{w}_i \tag{2.11}$$

$$\boldsymbol{y}_i = [1 \quad 0 \quad \cdots \quad 0]\boldsymbol{x}_i \tag{2.12}$$

Se puede escribir también el proceso autoregresivo (2.6) en la forma espacio de estados como:

$$\mathbf{x}_{i+1} = A\mathbf{x}_i + G\boldsymbol{w}_i \tag{2.13}$$

$$\boldsymbol{y}_i = H_i\boldsymbol{x}_i \tag{2.14}$$

La ecuación 2.13 describe el nuevo estado $\boldsymbol{x}_{i+1}$ modelado como una combinación lineal del estado previo $\boldsymbol{x}_i$ y el ruido del proceso $\boldsymbol{w}_i$. En la ecuación 2.14 se describe la forma de las mediciones del proceso u observaciones $\boldsymbol{y}_i$, las cuales dependen del estado interno $\boldsymbol{x}_i$. Estas ecuaciones son conocidas como el modelo del proceso y el modelo de medición respectivamente, son la base de todos los métodos de estimación lineal [10].

2.6 Filtro de Kalman

En una gran cantidad de casos del modelado de sistemas prácticos, solo algunos estados están disponibles físicamente para llevar a cabo mediciones sobre ellos. En la mayoría de los casos, solamente la variable de salida se encuentra a disposición y, por lo tanto, con todo lo que contamos es con una relación (a través de las ecuaciones) entre el valor de la salida y su dependencia con respecto a los estados [11].

Se puede deducir un valor estimado del vector a partir de la disponibilidad de los valores de salida y conocimiento del sistema. Para llevar esta tarea a cabo se desarrolló el filtro de Kalman, el cual es un grupo de ecuaciones matemáticas que

implementa un estimador predictor-corrector que es óptimo refiriéndose a que minimiza el error estimado de covarianza.

Debido al avance de la tecnología computacional este filtro ha sido sujeto de mucha investigación e implementación en campos como la navegación, seguimiento en gráficos de computadora interactivos, predicción de movimiento y fusión sensorial.

Este filtro tiene como finalidad estimar el estado desconocido $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{n}}$ en puntos discretos en el tiempo de un proceso estocástico modelado en espacio de estados de la forma:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{2.15}$$

Con las mediciones $\boldsymbol{z} \in \mathbb{R}^{\boldsymbol{m}}$

$$z_k = Hx_k + v_k \tag{2.16}$$

Donde H es la matriz de observación que determina las combinaciones lineales del vector de estados x_k y v_k que es una secuencia de vectores de sin correlación con una distribución normal con media cero y covarianza R_k [12].

$$R_k = E\left[v_k v_k^T\right] \tag{2.17}$$

$$p(V) \sim N(0, R) \tag{2.18}$$

w_k y v_k con variables randómicas que representan ruido blanco de media cero, presente en el proceso y medición respectivamente, se asume que entre ellas no existe correlación $\left(S_k = E\left[v_k w_j^T\right] = 0\right)$, cuando no hay realimentación.

$$E\left[\begin{bmatrix} w_k \\ v_k \end{bmatrix} \begin{bmatrix} w_j{}^* & v_j{}^* \end{bmatrix}\right] = \begin{bmatrix} Q_k & S_k \\ S_k{}^* & R_k \end{bmatrix} \delta_{kj} \tag{2.19}$$

En la práctica las matrices de covarianza del proceso Q_k y covarianza de las medidas R_k pueden cambiar en cada instante de tiempo o medición, para fines de cálculo y explicación se asumen constantes.

Se asume que las matrices A de dimensión $(n\ x\ n)$ en la ecuación 2.15 relaciona el estado previo del sistema $(k-1)$ con el estado actual (k), en ausencia de señal de control o ruido. La matriz B de dimensión $(n\ x\ l)$ asocia la señal de

control ($\boldsymbol{u}$) con el estado ($\boldsymbol{x}$). H de dimensión $(m\ x\ n)$ observada en la ecuación 2.16, determina el estado de la medición $\boldsymbol{z}_k$.

Las matrices A, B, H, $Q_k\ (l\ x\ l)$, $R_k\ (m\ x\ m)$, $S_k\ (l\ x\ m)$ se asumen conocidas.

La ecuación del observador de estados Filtro de Kalman se muestra en la ecuación 2.20

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{2.20}$$

Donde $\hat{x}_k^- \in \mathbb{R}^n$ es la estimación de estado a priori (predicción) en el tiempo k teniendo conocimiento del proceso antes del step k ($\hat{x}_{k-1}$). El estado $\hat{x}_k \in \mathbb{R}^n$ se conoce como estimación del estado posterior, obtenido en el tiempo k dada la medición z_k.

Estos términos se pueden apreciar de mejor manera observando la figura 2.9, donde se explica como el filtro de Kalman se basa en la información de la predicción $\hat{x}_k^-$ y la medición z_k para calcular cuál de ellas posee el menor error cuadrático medio y obtener el estimador óptimo de estado $\hat{x}_k$.

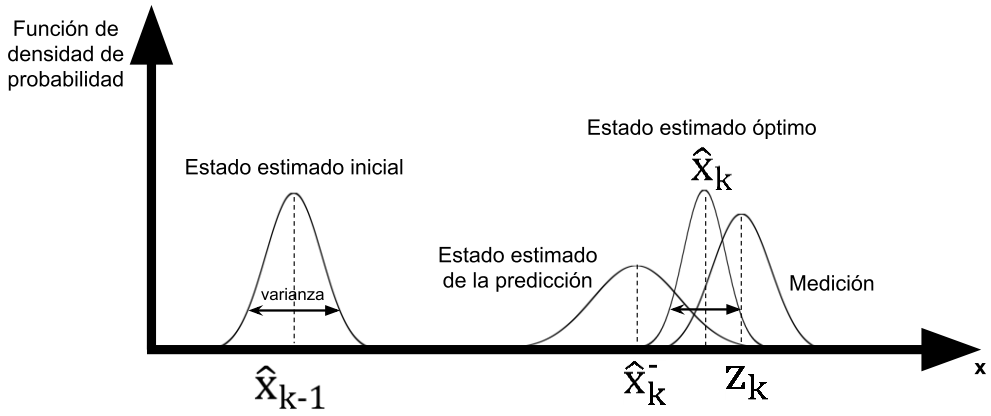


Figura 2.9: Estados de la ecuación del Filtro de Kalman.

Los errores de los estados de predicción y posterior se muestran en las ecuaciones 2.21 y 2.22 respectivamente:

$$e_k^- \equiv x_k - \hat{x}_k^- \tag{2.21}$$

$$e_k \equiv x_k - \hat{x}_k \tag{2.22}$$

Basados en estos errores se define las covarianzas de error de la predicción y del estimador del estado posterior.

$$P_k^- = E[e_k^- e_k^{-\,*}] \tag{2.23}$$

$$P_k = E[e_k e_k{}^*] \tag{2.24}$$

Continuando con los valores necesarios para desarrollar el algoritmo de la ecuación 2.20, la expresión $(z_k - H\hat{x}_k^-)$ se refiere a la innovación de la medida, esta proporciona la diferencia entre la medida predicha $H\hat{x}_k^-$ y la medida actual z_k.

2.6.1 Ganancia de Kalman

La matriz K $(n\ x\ m)$ de la ecuación 2.20 es conocida como la ganancia de Kalman, minimiza el error de la covarianza del estimador del estado posterior, ecuación 2.24.

Reemplazando la ecuación 2.20 en la definición del error e_k, sustituyendo esto en la ecuación 2.24 y resolviendo para despejar K se obtiene:

$$K_k = P_k^- H^* (H P_k^- H^* + R)^{-1}$$

$$K_k = \frac{P_k^- H^*}{H P_k^- H^* + R} \tag{2.25}$$

Haciendo un análisis del denominador de la ecuación 2.25 se puede ver como el peso de la ganancia de Kalman determina como contribuyen las mediciones y el estado estimado de predicción al cálculo del estimador óptimo de estado $\hat{x}_k$.

$$\lim_{\mathrm{R}\to 0} K_k = H^{-1} \tag{2.26}$$

De la ecuación 2.26 se interpreta, si el ruido de la medición es pequeño la medición es más confiable y contribuye a $\hat{x}_k$ más que el estado estimado de predicción.

$$\lim_{P_k^- \to 0} K_k = 0 \tag{2.27}$$

La ecuación 2.27 muestra el caso opuesto a lo antes expuesto, donde el error de la predicción estimada es pequeño, por ende, más confiable; el valor de $\hat{x}_k$ está definido por la estimación.

2.6.2 El Algoritmo del Filtro de Kalman

Este filtro estima un proceso usando una forma de realimentación. El filtro estima el estado del proceso en un determinado tiempo, después obtiene mediciones (ruidosas) como realimentación. Existen dos grupos de ecuaciones de las que este observador hace uso:

- Ecuaciones de actualización en el tiempo, encargadas de predecir en el tiempo el estado actual y error de covarianza estimados para obtener los estados de estimación de predicción para el siguiente paso del tiempo, por ellos también conocidas como ecuaciones de predicción.
- Ecuaciones de actualización de observaciones, responsables de la realimentación para incorporar nueva información de medición en el estado estimado de predicción para obtener una mejora en el estado posterior, por ellos se las conoce como ecuaciones de corrección.

A continuación, se muestran las ecuaciones utilizadas para cada uno de estos dos grupos de igualdades, de predicción y corrección.

Ecuaciones de Actualización en el Tiempo.

Ecuaciones de Predicción.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.28}$$

$$P_k^- = AP_{k-1}A^* + Q \tag{2.29}$$

Ecuaciones de Actualización de las Observaciones.

Ecuaciones de Corrección.

$$K_k = P_k^- H^* (HP_k^- H^* + R)^{-1} \tag{2.30}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.31}$$

$$P_k = (I - K_k H)P_k^- \tag{2.32}$$

La primera tarea para implementar el algoritmo es desarrollar las ecuaciones de corrección de las cuales se debe empezar por cuantificar la ganancia de Kalman K_k, seguido de la actualización de las mediciones del proceso z_k, a fin de generar el estado posterior $\hat{x}_k$. Se finaliza esta parte del cómputo consiguiendo el error de covarianza posterior P_k.

A este procedimiento le precede el desarrollo y deducción de las ecuaciones de predicción en el orden que se puede apreciar en la figura 2.10.

Entre cada paso de las ecuaciones de actualización de las observaciones el nuevo estado posterior estimado es usado como el estado de predicción en el próximo cálculo de las ecuaciones de corrección, lo que hace que este filtro sea recursivo, una de las características más comentadas y atractivas del estimador.

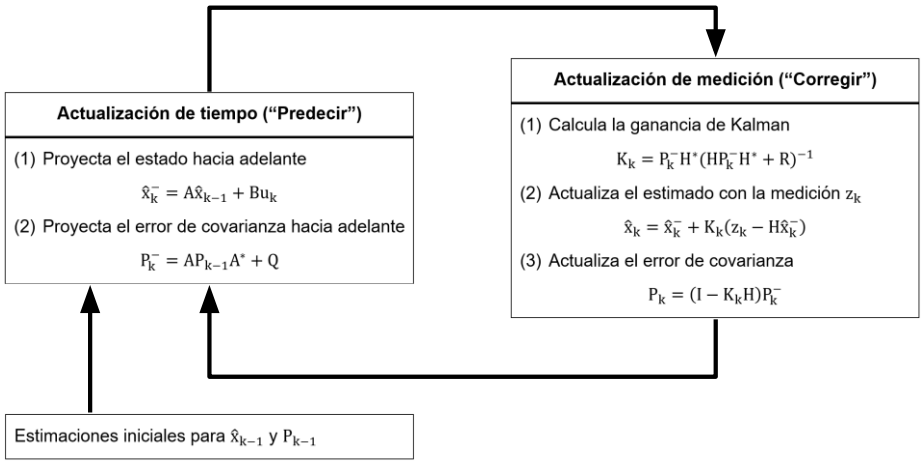


Figura 2.10: Representación del algoritmo recursivo del Filtro de Kalman.

CAPÍTULO 3

3. IMPLEMENTACIÓN.

Este robot autobalanceado tipo péndulo se armó sobre el chasis del robot CKK0001-1 de la marca Cokoino. Los dispositivos electrónicos como sensores, tarjeta de control para motores, tarjeta Arduino, han sido ensamblados a conveniencia para este proyecto. El sistema computacional fue programado en el software disponible para Arduino, donde se realizan acciones como la lectura de sensores, filtro de Kalman, comunicación serial, control de motores.

En este capítulo se explica a detalle los pasos y procesos llevados a cabo para lograr la implementación física como informática del autómata.

3.1 Estructura del Sistema Mecánico

Para la construcción del robot autobalanceado se tomó como base el modelo CKK0001-1 de la marca Cokoino. El chasis está compuesto por dos láminas de acrílico de 120 mm x 85 mm, dos llantas de 65 mm, 4 columnas hexagonales de cobre M3 x 45 mm, dos soportes para motores y dos motores con encoder JGB37-520, los cuales se conectan como se observa en la figura 3.1.

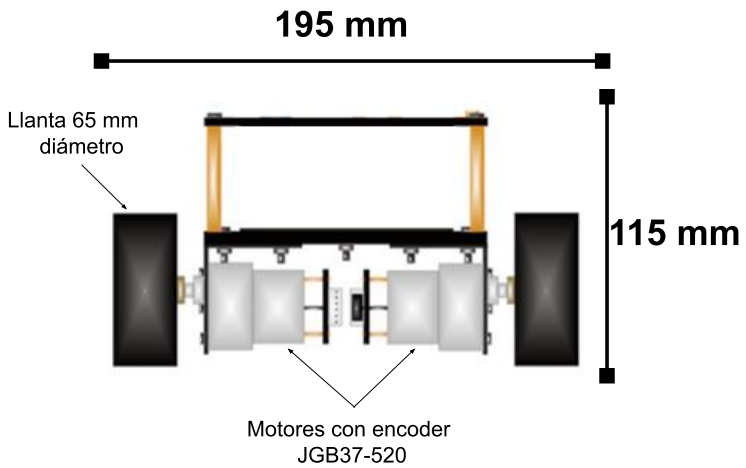


Figura 3.1 Diagrama de estructura y dimensiones del robot.

En el piso superior del robot se ubican la placa Arduino Uno, el módulo HC-05 y el driver para motores l298n. En el piso inferior se encuentran la porta pilas con 3 baterías del tipo 18650 de 3.7 v cada una y el sensor MPU-6050.

Para la orientación del sensor de medición se lo hizo de modo tal que el eje X de este coincida con el eje de las ruedas del motor, lo que significa que la inclinación del vehículo será el Roll del acelerómetro

Las conexiones de los diferentes elementos se muestran a continuación en la figura 3.2:

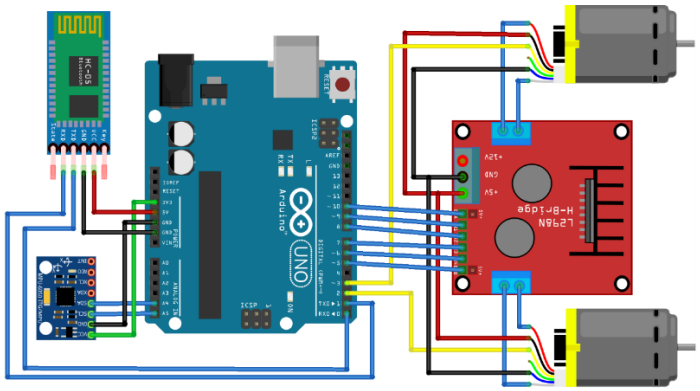


Figura 3.2 Diagrama de conexiones del robot.

La estructura final del robot se aprecia en la figura 3.3.

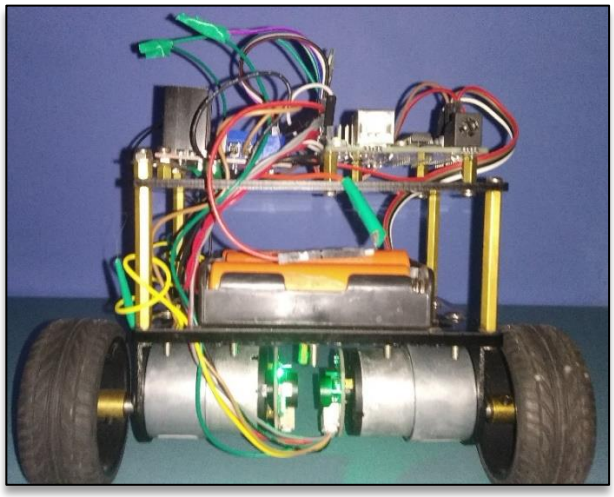


Figura 3.3 Robot ensamblado.

3.2 Estructura del Sistema

El módulo central o módulo de control está compuesto por la placa Arduino y cuyo objetivo principal es mantener el robot en equilibrio, para esto debe recibir y filtrar las mediciones que vienen del módulo medición del ángulo del cual se habla más adelante. En base al ángulo que reciba el control, este decide si aumenta o disminuye la velocidad de las ruedas, esto lo hace a través del puente H. Los encoder magnéticos que se encuentran en los motores son los encargados de indicarle al control si están o no girando a la velocidad solicitada. Por otro lado, el robot se comunica de forma inalámbrica con una computadora en caso de que una persona quiera modificar su trayectoria a través de una aplicación programada en Visual Basic. La estructura del sistema puede ser representada como en la figura 3.4.

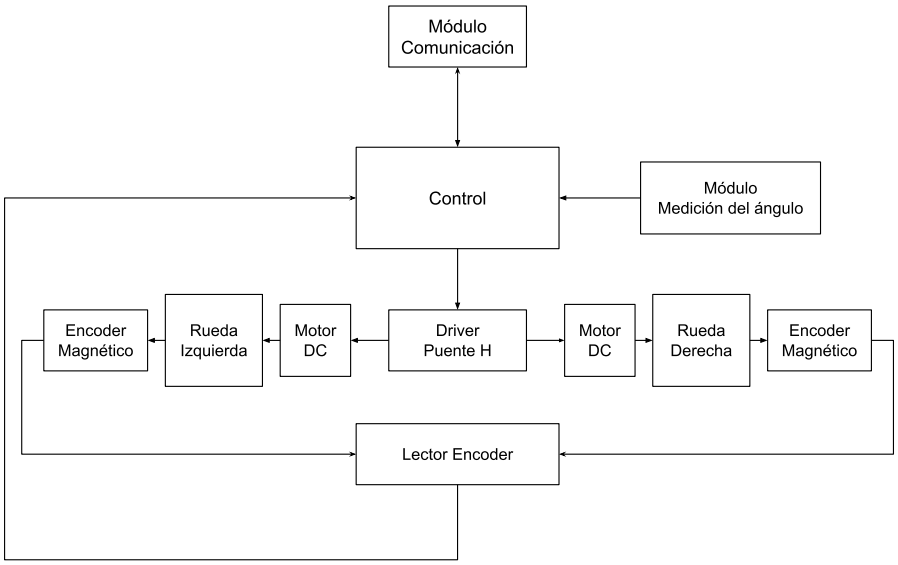


Figura 3.4 Diagrama de flujo de la estructura del sistema.

3.3 Comunicación Bluetooth

La comunicación entre el robot y la computadora se realiza vía bluetooth. Del lado del robot el módulo de comunicación cuenta con el integrado HC-05, mientras que del lado del pc se ejecuta una aplicación programada en Visual Basic, tal como el que se observa en la figura 3.5:

Figura 3.5 Representación del módulo de comunicación.

El circuito HC-05 es configurado en modo esclavo con una velocidad de transmisión de 115200 baudios, por defecto viene con otro valor (9600), de modo que es necesario configurarlo vía comandos AT.

De forma inalámbrica el robot está siempre esperando una variable de tipo ‘char’ que es enviada desde la computadora por una comunicación serial y de acuerdo con el valor que tome esa variable el robot ejecutará las distintas acciones programadas, que pueden ser avanzar hacia adelante, atrás, girar a la izquierda, girar a la derecha, o modificar alguno de las ganancias de los PIDs. Mientras que del lado robot cada 0.5 segundos estará enviando sus mediciones del ángulo filtrado, lectura de los encoders, y valores de los PWM de cada rueda, además de los valores del PIDs en caso de que se necesiten ajustar.

Para la programación del bluetooth del lado de la tarjeta Arduino no hace falta modificar nada más, el módulo bluetooth ya viene listo para realizar la comunicación serial, solo es necesario conectar correctamente los pines Rx y Tx de la del módulo HC-05 con los pines Tx y Rx de la placa Arduino, se deben conectar de forma cruzada tal como se indica en la figura 3.2.

Y del lado del PC para poder comunicarse es necesario que se cuente con un bluetooth que trabaje en la banda de los 2.4 GHz, una vez conectados el computador asigna dos puertos de comunicación, ver figura 3.6.

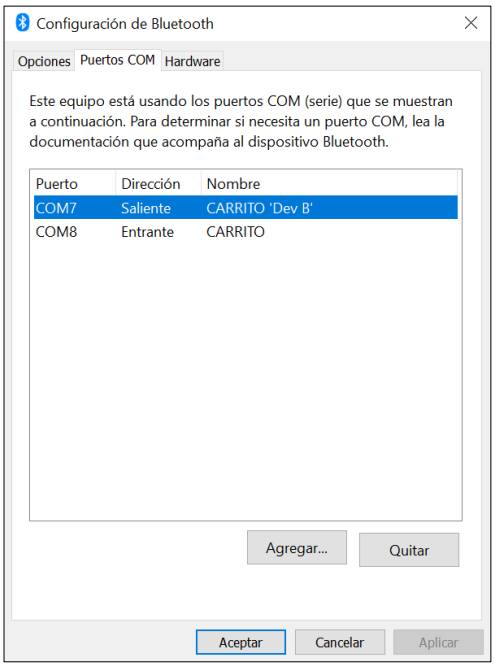


Figura 3.6 Puertos del PC asignados para la comunicación Bluetooth.

Para realizar la programación en Visual Basic se debe tener en cuenta el puerto de dirección saliente, en el computador con el que se realizaron las pruebas fue el COM7. Ese puerto debe ser seleccionado al momento de ejecutar el programa tal como se observa en la figura 3.7.

Figura 3.7 Selección del puerto de comunicación en el programa desarrollado en Visual Basic.

Durante la ejecución del programa el módulo HC-05 hará de esclavo mientras que el bluetooth del computador hará de maestro del modelo de comunicación.

3.4 Fusión Sensorial aplicando el Filtro de Kalman

El módulo de medición del ángulo del robot autobalanceado consiste en un módulo MPU-6050, cuenta con un acelerómetro de 3 ejes y un giroscopio de 3 ejes, ambos pueden ser utilizados de forma independiente para calcular el ángulo de inclinación, sin embargo, presentan errores de medición. En el caso del acelerómetro las mediciones se ven afectadas por el ruido y por los movimientos del sensor, mientras para el giroscopio que mide velocidad angular, si se desea calcular el ángulo es necesario realizar un algoritmo acumulativo para obtener la integral con respecto al tiempo, pues la medición de este sensor acumula errores y ruido a medida que aumenta el ángulo, a esto se lo conoce como deriva, este concepto explicado numéricamente está ilustrado en la ecuación 2.5.

Estos precedentes mencionados reflejan el conflicto que el giroscopio y acelerómetro generan al momento de obtener directamente un valor medido, viéndonos en la obligación de tratar estas señales para obtener valores de medición angular confiables.

De modo que para calcular el ángulo de la mejor manera posible usando el integrado MPU-6050 se fusionan ambos sensores (acelerómetro y giroscopio) usando el filtro de Kalman como se explica a continuación.

El estado del sistema en el tiempo k viene dado por la ecuación 2.15, al relacionarlo con la ecuación 2.5 que describe el sistema de medición del ángulo del giroscopio en tiempo discreto, si el ángulo θ representa la variable de estado a corregir entonces:

$$\theta_k^{-} = A\,\theta_{k-1} + B\dot{\theta}_k + w_k \tag{3.1}$$

$$z_k = H\theta_k^{-} + v_k \tag{3.2}$$

La ecuación 3.1 representa al sistema de medición de la variable de estado θ (ángulo de inclinación), la ecuación 3.2 expresa el observador del sistema. Debido a que el objetivo es la fusión sensorial entre el giroscopio (modelo del sistema) y acelerómetro (observador), el modelo presentado necesita la inclusión

de dos variables de estado. Por lo tanto, la expresión 3.1 tendrá orden matricial el cual se explica a continuación para cada término.

A partir de la ecuación 3.1 se aclaran a continuación cada uno de sus términos y valores correspondientes:

- La matriz de estado $\theta_k{}^-$ viene dada de la siguiente manera:

$$\theta_k{}^- = \begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_k \tag{3.3}$$

$\dot{\theta}_b$ indica cuanto el giroscopio se ha desviado, si se desea calcular la velocidad real se debe restar la desviación $\dot{\theta}_b$ de la medición del giroscopio [13].

- La matriz de transición A de dimensión $(2\ x\ 2)$

$$A = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix} \tag{3.4}$$

- La matriz B $(2\ x\ 1)$ o matriz de control se define como:

$$B = \begin{pmatrix} \Delta t \\ 0 \end{pmatrix} \tag{3.5}$$

- El ruido del proceso w_k, tiene una distribución gaussiana con media cero y matriz de covarianza Q en el tiempo k.

$$w_k \sim N(0, Q_k)$$

Q_k es la matriz de covarianza de ruido del proceso que para esto caso representa la matriz de covarianzas del estado estimado del acelerómetro y giroscopio. Se asume que ambas son variables independientes, no correlacionadas.

$$Q_k = \begin{pmatrix} Q_\theta & o \\ 0 & Q_{\dot{\theta}_b} \end{pmatrix} \Delta t \tag{3.6}$$

La matriz de covarianza Q_k depende del tiempo actual k por lo tanto es necesario multiplicar la varianza del acelerómetro Q_θ y la varianza de la desviación $Q_{\dot{\theta}_b}$ por Δt.

De la ecuación 3.2 que representa las mediciones se explica sus componentes en forma matricial a continuación:

- H de dimensiones $(1\ x\ 2)$ es la matriz de observación, usada para mapear el estado de espacios real dentro del espacio observado.

$$H = (1 \quad 0) \tag{3.7}$$

- El ruido de la medición v_k, tiene una distribución gaussiana con media cero y covarianza R.

$$v_k \sim N(0, R)$$

Donde R representa la covarianza de la medición, en este caso la covarianza de una misma variable da como resultado la varianza de esta.

$$R = E[v_k v_k^*] = var(v_k) \tag{3.8}$$

Se asume que el ruido de la medición es el mismo en todo instante de tiempo k.

$$var(v_k) = var(v)$$

3.4.1 Valores de las Matrices de Covarianza Q_k y R.

Se ha dispuesto un apartado para explicar la manera en que se obtuvo los valores de estas matrices por lo importantes que son en el desarrollo y buen funcionamiento de este filtro.

La matriz de covarianzas de proceso Q_k, representa el ruido asociado al estado estimado tanto del acelerómetro como del giroscopio en cada instante de tiempo, lo que podría calcularse en cada cálculo de estimación, pero al representar valores pequeños asociados en la ecuación 2.29 a la matriz de covarianza de error del estado estimado posterior (P_{k-1}), se hacen despreciables; pues la operación para conocer el error actual presente en el estado estimado de predicción se realiza en cada actualización del algoritmo, debido a esto se propone dar un valor 0 a Q_k, o cantidades pequeñas, realizando una búsqueda de literatura se encontró que en muchos casos como en [14] propone asignar $Q_\theta = 0.001$ y $Q_{\dot{\theta}_b} = 0.003$ o valores cercanos a estos a prueba y error. Para nuestro caso estas cuantías dieron buenos resultados y son las utilizadas para la matriz Q_k.

La matriz de varianza de error en las mediciones R, representan el error asociado al acelerómetro pues como se explicó este es el observador del sistema, en referencias como [15], sugiere calcular para cada instante de tiempo dicho valor de varianza, más, al igual que en el caso de la covarianza Q_k, se recomienda dar valores constantes. Existen múltiples sugerencias para calcular este valor constante como en [16], pero se decidió basarnos en la definición de ruido, entonces se tomó una muestra en un instante determinado de tiempo las lecturas del acelerómetro a x° de inclinación (figura 3.8).

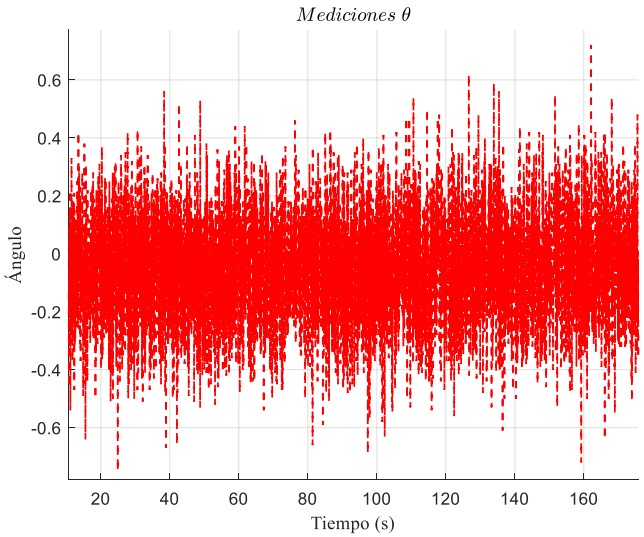


Figura 3.8 Ángulo constante calculado usando el acelerómetro.

Para validar que este ruido presente en las mediciones sea blanco y cumpla con las condiciones determinadas en la ecuación 2.18 se utilizó el toolbox de Matlab distributionFitter, en el asistente de la herramienta se seleccionó un ajuste de distribución normal, ver figura 3.9.

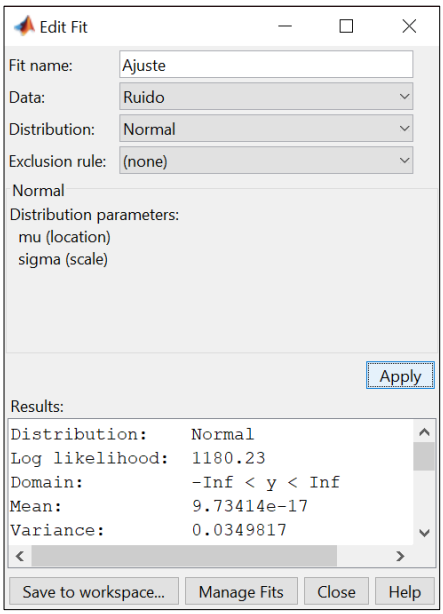


Figura 3.9 Asistente Distribution Fitter de Matlab

Se obtuvo que el ruido tiene distribución normal de media $\mu = 0$ y varianza $\sigma^2 = 0.035$. Finalmente se grafica el histograma de los datos tomados y se aprecia como se ajusta a una distribución normal en la figura 3.10.

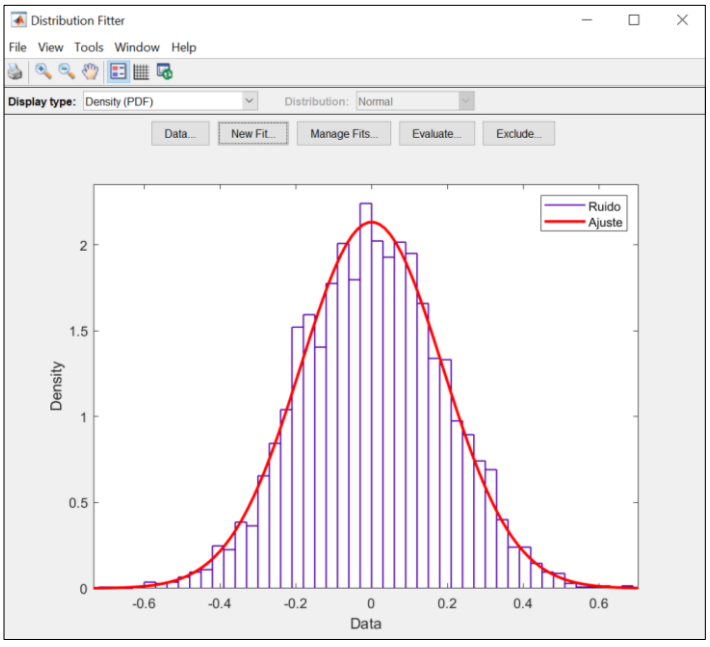


Figura 3.10 Función de densidad de probabilidad del Ruido de las mediciones del Ángulo.

3.4.2 Observabilidad del Sistema

Teniendo claras las matrices del sistema en la forma de observador asintótico [17], es imperativo hacer un análisis de la observabilidad del sistema para saber si podemos obtener información de los estados del sistema a partir del conocimiento que tenemos de la entrada y salida, caso contrario sería un trabajo improductivo poner en marcha la programación de un estimador sin saber si será posible observar los estados.

Con la ayuda del software MATLAB se aplicó el comando “obsv” a las matrices A y H para analizar el resultado del rango de la matriz de observabilidad obtenida, consiguiendo como resultado que el sistema es observable al cumplirse lo indicado en [18]. El archivo .m usado para este fin se puede observar en el anexo y la salida de este aparece a continuación en la figura 3.11.

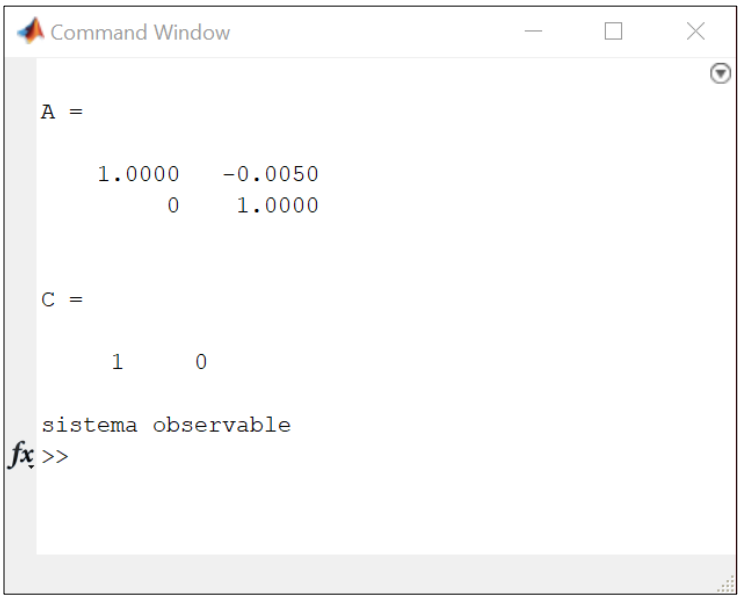


Figura 3.11 Salida de consola de script que calcula observabilidad del sistema.

3.4.3 Implementación del Algoritmo del Filtro de Kalman.

Para implementar el algoritmo del filtro de Kalman se necesita seguir los pasos indicados en la figura 2.10; donde se aprecia claramente la necesidad de conocer previamente los estados iniciales de las variables:

matriz de covarianza del error estimado de la predicción (P_{k-1}) y la variable de estado estimado de predicción en el tiempo K-1 $(\hat{x}_{k-1})$. Para nuestro proceso de fusión del acelerómetro y giroscopio dichos valores iniciales está determinados por las matrices:

$$\hat{x}_0(2\ x\ 1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathrm{P}_0(2\ x\ 2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Las ecuaciones del filtro de Kalman expresadas en términos de las variables de estado utilizadas para la fusión de sensores se presentan a continuación:

Ecuaciones de Predicción.

$$\hat{\theta}_k^- = A\hat{\theta}_{k-1} + B\dot{\theta}_k \tag{3.9}$$

$$P_k^- = AP_{k-1}A^* + Q_k \tag{3.10}$$

Ecuaciones de Corrección.

$$K_k = P_k^- H^*(HP_k^- H^* + R)^{-1} \tag{3.11}$$

$$\hat{\theta}_k = \hat{\theta}_k^- + K_k\left(z_k - H\hat{\theta}_k^-\right) \tag{3.12}$$

$$P_k = (I - K_k H)P_k^- \tag{3.13}$$

Estas igualdades deben ser desarrolladas en el orden como se aprecia en la figura 2.10, empezando por las ecuaciones de actualización en el tiempo, la explicación de cada una de ellas y su representación matricial se explica paso a paso:

1. La ecuación 3.9, representada con sus valores matriciales y reducida se expresa de la siguiente manera:

$$\begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_k^- = \begin{pmatrix} 1 & -\Delta t \\ 0 & 1 \end{pmatrix}\begin{pmatrix} \theta \\ \dot{\theta}_b \end{pmatrix}_{k-1} + \begin{pmatrix} \Delta t \\ 0 \end{pmatrix}\dot{\theta}_k$$

$$\begin{pmatrix}\theta\\ \dot{\theta}_b\end{pmatrix}_k^- = \begin{pmatrix}\theta + \Delta t(\dot{\theta}_k - \dot{\theta}_b)\\ \dot{\theta}_b\end{pmatrix} \tag{3.14}$$

2. La matriz de covarianza de error estimado de la predicción (ecuación 3.10), se calcula al desarrollar expresión:

$$\underbrace{\begin{pmatrix}P_{00} & P_{01}\\ P_{10} & P_{11}\end{pmatrix}_k^-}_{P_k^-} = \left[\underbrace{\begin{pmatrix}1 & -\Delta t\\ 0 & 1\end{pmatrix}}_{A}\underbrace{\begin{pmatrix}P_{00} & P_{01}\\ P_{10} & P_{11}\end{pmatrix}_{k-1}}_{P_{k-1}}\underbrace{\begin{pmatrix}1 & 0\\ -\Delta t & 1\end{pmatrix}}_{A^*}\right] + \left[\underbrace{\begin{pmatrix}Q_\theta & 0\\ 0 & Q_{\dot{\theta}_b}\end{pmatrix}}_{Q_k}\Delta t\right]$$

$$\begin{pmatrix}P_{00} & P_{01}\\ P_{10} & P_{11}\end{pmatrix}_k^- = \begin{pmatrix}P_{00} + \Delta t(\Delta t P_{11} - P_{01} - P_{10} + Q_\theta) & P_{01} - \Delta t P_{11}\\ P_{10} - \Delta t P_{11} & P_{11} + Q_{\dot{\theta}_b}\Delta t\end{pmatrix}_{\substack{k-1\\(2x2)}} \tag{3.15}$$

3. El orden de cálculo una vez terminadas las ecuaciones de predicción se debe mantener al desarrollar las ecuaciones de corrección, por lo tanto, la primera es la ecuación 3.11 que se refiere al cálculo de la ganancia de Kalman. Al tratarse de una fórmula extensa ha sido tratada por partes como se explica a continuación:

$$S_k = H\mathrm{P}_k^- H^* + R$$

$$S_k = (1 \quad 0)\begin{pmatrix}P_{00} & P_{01}\\ P_{10} & P_{11}\end{pmatrix}_k\begin{pmatrix}1\\0\end{pmatrix} + R$$

$$S_k = (P_{00})_k^- + R$$

$$S_k = (P_{00})_k^- + var(v) \tag{3.16}$$

Reemplazando S_k en la ecuación 3.11, obtenemos:

$$K_k = \mathrm{P}_k^-\ H^*\ (S_k)^{-1}$$

$$\begin{pmatrix}K_0\\K_1\end{pmatrix} = \begin{pmatrix}P_{00} & P_{01}\\ P_{10} & P_{11}\end{pmatrix}_k^-\begin{pmatrix}1\\0\end{pmatrix}(S_k)^{-1}$$

$$\begin{pmatrix}K_0\\K_1\end{pmatrix} = \frac{\begin{pmatrix}P_{00}\\P_{10}\end{pmatrix}_k^-}{S_k} \tag{3.17}$$

4. La ecuación 3.12, representa el estado estimado óptimo o como se mencionó en la ecuación 2.20 se trata de la ecuación del observador Filtro de Kalman, en la cual está presente la innovación en este caso

representada por $\tilde{y}_k$, misma que se recomienda calcular por separado para luego reemplazar en la ecuación del estimador.

$$\tilde{y}_k = z_k - H\,\hat{\theta}_k^-$$

$$\tilde{y}_k = z_k - \left[\begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}\theta \\ \dot{\theta}_b\end{pmatrix}_k^-\right]$$

$$\tilde{y}_k = z_k - \theta_k^- \tag{3.18}$$

Sustituyendo la innovación en la ecuación 3.12 se obtiene la expresión:

$$\hat{\theta}_k = \hat{\theta}_k^- + (K_k\,\tilde{y}_k)$$

$$\begin{pmatrix}\theta \\ \dot{\theta}_b\end{pmatrix}_k = \begin{pmatrix}\theta \\ \dot{\theta}_b\end{pmatrix}_k^- + \begin{pmatrix}K_0 \\ K_1\end{pmatrix}_k \tilde{y}_k \tag{3.19}$$

5. La última ecuación de corrección por implementar es la expresada en la ecuación 3.13 correspondiente a la matriz de covarianza del estimador del estado posterior, en la cual se hace uso de una matriz identidad de dimensión $(2x2)$.

$$\underbrace{\begin{pmatrix}P_{00} & P_{01} \\ P_{10} & P_{11}\end{pmatrix}_k}_{P_k} = \left[\underbrace{\begin{pmatrix}1 & 0 \\ 0 & 1\end{pmatrix}}_{I} - \begin{pmatrix}K_0 \\ K_1\end{pmatrix}_k \underbrace{\begin{pmatrix}1 & 0\end{pmatrix}}_{H}\right]\underbrace{\begin{pmatrix}P_{00} & P_{01} \\ P_{10} & P_{11}\end{pmatrix}_k^-}_{P_k^-}$$

$$\begin{pmatrix}P_{00} & P_{01} \\ P_{10} & P_{11}\end{pmatrix}_k = \left[\begin{pmatrix}P_{00} & P_{01} \\ P_{10} & P_{11}\end{pmatrix}_k^-\right] - \begin{pmatrix}K_0P_{00} & K_0P_{01} \\ K_1P_{00} & K_1P_{01}\end{pmatrix} \tag{3.20}$$

Las ecuaciones 3.14 hasta la 3.20 se implementaron en el lenguaje de programación de Arduino con el fin de poner en funcionamiento el Filtro de Kalman para la fusión de sensores. Anexo B.

3.5 Control del Robot

Para controlar el robot son requeridos 3 lazos de control uno de ellos encargado de mantener el equilibrio, dos de ellos para cuando existan ordenes desde Visual Basic para seguir una trayectoria hacia delante o reversa, otro para girar.

La manera de controlar el robot de estos 3 compensadores se ilustra en la figura 3.12, donde se observa que los 3 lazos trabajan al mismo tiempo, recibiendo información necesaria de los sensores, cuantificando su valor de control (Control

Value CV); al final un solo PWM (modulación de ancho de pulso) encargada de variar la tensión media suministrada a cada motor es calculada, misma que se basa en la suma de los 3 valores PWM de cada controlador. Cabe indicar que para cada llanta se utiliza un valor PWM diferente basado en la suma de los CV de los compensadores, con diferencia entre la llanta izquierda y derecha, donde a una de las llantas el PWM estimado para el giro en lugar de sumar es restado, este cálculo se observa en la figura 3.13.

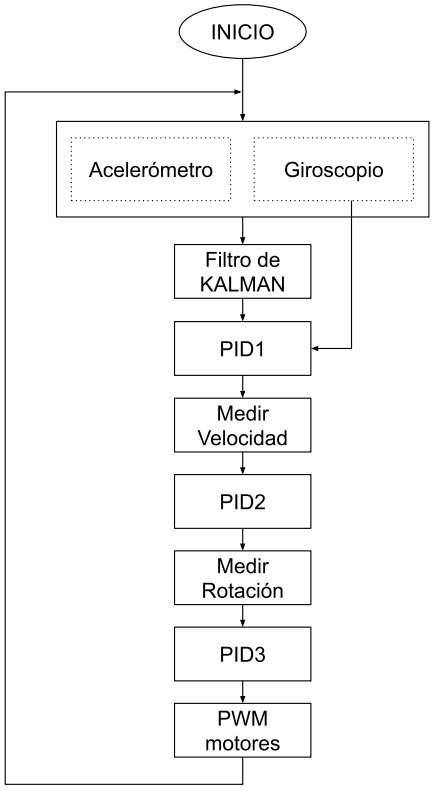


Figura 3.12: Diagrama de Flujo del algoritmo de control del robot

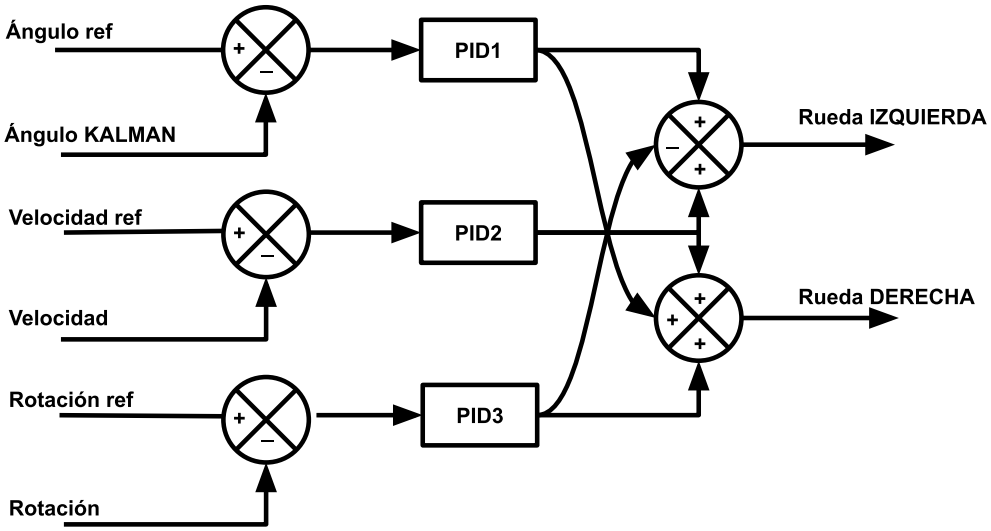


Figura 3.13: Cálculo de valor PWM para cada llanta a partir los 3 controladores del sistema.

El tipo de controlador utilizado para cada tipo movimiento es un control PID (proporcional, integral y derivativo), en sus diferentes combinaciones PI y PD, las cuales son explicadas detalladamente para cada compensador.

3.5.1 Control de Equilibrio

Este control está enfocado en mantener el ángulo $\theta = 0°$, logrando equilibrar el robot tipo péndulo invertido, antes de describir el algoritmo para este control es necesario hacer un breve repaso de las fuerzas y el modelo del péndulo invertido para mejorar la idea de cuáles serán los datos de sensores a utilizar, set point, para mejorar la idea del tipo de sistema al cuál se aplicará el control.

La figura 3.14 muestra el diagrama de cuerpo libre del péndulo invertido donde se aprecian las fuerzas que actúan sobre él, dimensiones, gravedad, etc.

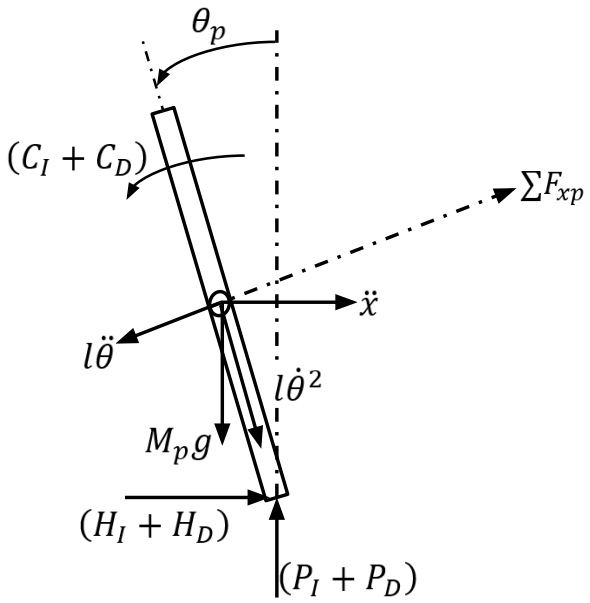


Figura 3.14: Diagrama de cuerpo libre del robot autobalanceado tipo péndulo invertido [20].

Donde:

M_p: Masa del péndulo.

θ_p: Posición Angular.

l: Distancia del centro de las llantas al centro de gravedad del robot.

$H_I + H_D$: Fuerza de Reacción entre las llantas y el chasis.

$P_I + P_D$: Fuerza de Reacción entre las llantas y el chasis.

$C_I + C_D$: Torque de la llanta izquierda y derecha aplicado por el motor DC

I_p: inercia del péndulo alrededor del eje de la llanta.

Como explica [19], el siguiente es un modelo básico de entre muchos que existen, en este modelo no se asumen perturbaciones ni fricción con el piso.

Aplicando la ley de Newton de movimiento:

$$\sum F_x = M_p \ddot{x}$$

$$M_p\ddot{x} = (H_I + H_D) - M_p l\ddot{\theta}_p cos(\theta_p) + M_p l\dot{\theta}_p^2 sen(\theta_p)$$

$$(H_I + H_D) = M_p\ddot{x} + M_p l\ddot{\theta}_p cos(\theta_p) - M_p l\dot{\theta}_p^2 sen(\theta_p) \quad (3.21)$$

La sumatoria de fuerzas perpendicular al péndulo:

$$\sum F_{xp} = M_p\ddot{x}_p cos(\theta_p)$$

$$(H_I + H_D)cos(\theta_p) + (P_I + P_D)sen(\theta_p) - M_p g sen(\theta_p) - M_p l\ddot{\theta}_p = M_p\ddot{x}_p cos\theta_p \quad (3.22)$$

La sumatoria de momentos alrededor del centro de masa del péndulo:

$$\sum M_0 = I\alpha$$

$$-(H_I + H_D)lcos(\theta_p) - (P_I + P_D)lsen(\theta_p) - (C_I + C_D) = I_p\ddot{\theta}_p \quad (3.23)$$

De las ecuaciones de movimiento asociadas a las llantas izquierda y derecha alimentadas por un motor DC [20], la expresión $(C_I + C_D)$ representa el torque en las dos ruedas:

$$(C_I + C_D) = \frac{-2K_m K_e\dot{x}}{Rr} + \frac{2K_m}{R}V_a$$

$$(H_I + H_D) = \frac{-2K_m K_e\dot{x_w}}{Rr^2} + \frac{2K_m}{R}V_a - 2\left(M_w - \frac{I_w}{r^2}\right)\ddot{x}$$

Donde:

K_m: Constante del torque del motor DC.

K_e: Constante de la fuerza electromotriz del motor.

V_a: Voltaje del motor.

R: Resistencia del motor DC.

r: Radio de las llantas.

M_w: Masa de la llanta.

I_w: inercia de la llanta alrededor de su eje.

Sustituyendo en la ecuación 3.23:

$$-(H_I + H_D)lcos(\theta_p) - (P_I + P_D)lsen(\theta_p) = \frac{-2K_m K_e\dot{x}}{Rr} + \frac{2K_m}{R}V_a + I_p\ddot{\theta}_p$$

Multiplicando (3.22) por (l) y sustrayendo de la ecuación anterior:

$$\frac{-2K_mK_e\dot{x}}{Rr}+\frac{2K_m}{R}V_a+I_p\ddot{\theta}_p=-M_pglsen(\theta_p)-M_pl^2\ddot{\theta}_p-M_pl\ddot{x}cos(\theta_p) \quad (3.24)$$

Eliminando (H_I+H_D) de la ecuación 3.21:

$$2\left(M_w-\frac{I_w}{r^2}\right)\ddot{x}=\frac{-2K_mK_e}{Rr^2}\dot{x}_w+\frac{2K_m}{Rr}V_a-M_p\ddot{x}-M_pl\ddot{\theta}_pcos(\theta_p)+M_pl\dot{\theta}_p^2sen(\theta_p) \quad (3.25)$$

Reordenando las ecuaciones 3.24 y 3.25:

$$(I_p+M_pl^2)\ddot{\theta}_p-\frac{2K_mK_e}{Rr}\dot{x}+\frac{2K_m}{R}V_a+M_pglsen(\theta_p)=-M_pl\ddot{x}cos(\theta_p) \quad (3.26)$$

$$\frac{2K_m}{Rr}V_a=2\left(M_w-\frac{I_w}{r^2}\right)\ddot{x}+\frac{2K_mK_e}{Rr}\dot{x}_w+M_p\ddot{x}+M_pl\ddot{\theta}_pcos(\theta_p)-M_pl\dot{\theta}_p^2sen(\theta_p) \quad (3.27)$$

Las ecuaciones 3.26 y 3.27 describen el modelo no lineal del movimiento del robot. Estas ecuaciones son linealizadas alrededor de $\theta_p=0+\phi$, donde (ϕ) representa un ángulo pequeño.

Por lo tanto, como ϕ, se aproxima a cero:

$$cos(\theta_p)\equiv 1, sen(\theta_p)\equiv\phi, \ddot{\theta}_p=0$$

Las ecuaciones del modelo linealizado son:

$$(I_p+M_pl^2)\ddot{\phi}_p-\frac{2K_mK_e}{Rr}\dot{x}+\frac{2K_m}{R}V_a-M_pgl\phi=M_pl\ddot{x} \quad (3.28)$$

$$\frac{2K_m}{Rr}V_a=\left(2M_w-\frac{2I_w}{r^2}+M_p\right)\ddot{x}+\frac{2K_mK_e}{Rr^2}\dot{x}-M_pl\ddot{\phi}_p \quad (3.29)$$

Con el fin de representar el sistema en espacio de estados, despejando $\ddot{\phi}_p$ y $\ddot{x}$:

$$\ddot{\phi}_p=\frac{M_pl}{(I_p+M_pl^2)}\ddot{x}+\frac{2K_mK_e}{Rr(I_p+M_pl^2)}\dot{x}-\frac{2K_m}{R(I_p+M_pl^2)}V_a+\frac{M_pgl}{I_p+M_pl^2}\phi$$

$$\ddot{x}=\frac{2K_m}{Rr\left(2M_w-\frac{2I_w}{r^2}+M_p\right)}V_a-\frac{2K_mK_e}{Rr^2\left(2M_w-\frac{2I_w}{r^2}+M_p\right)}\dot{x}+\frac{M_pl}{Rr\left(2M_w-\frac{2I_w}{r^2}+M_p\right)}\ddot{\phi}_p$$

Sustituyendo estas variables en las ecuaciones del modelo lineal se obtiene la representación del modelo autobalanceado tipo péndulo:

$$\begin{pmatrix}\dot{x}\\ \ddot{x}\\ \dot{\phi}\\ \ddot{\phi}\end{pmatrix} = \begin{pmatrix}0 & 1 & 0 & 0\\ 0 & \frac{2K_mK_e(M_p l r - l_p - M_p l^2)}{Rr^2\alpha} & \frac{M_p^2 g l^2}{\alpha} & 0\\ 0 & 0 & 0 & 1\\ 0 & \frac{2K_mK_e(R\beta - M_p l)}{Rr\alpha} & \frac{M_p^2 g l \beta}{\beta} & 0\end{pmatrix}\begin{pmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{pmatrix} + \begin{pmatrix}0\\ \frac{2K_m(I_p + M_p l^2 - M_p l r)}{Rr\alpha}\\ 0\\ \frac{2K_m(M_p l - r\beta)}{Rr\alpha}\end{pmatrix} V_a \quad (3.30)$$

$$y = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}x\\ \dot{x}\\ \phi\\ \dot{\phi}\end{pmatrix} + \begin{pmatrix}0\\ 0\end{pmatrix} V_a \quad (3.31)$$

Donde:

$$\beta = 2M_w - \frac{2I_w}{r^2} + M_p \quad y \quad \alpha = I_p\beta + 2M_p l^2\left(M_w + \frac{I_w}{r^2}\right)$$

A partir de la ecuación de salida del modelo (ecuación 3.31), se puede observar que este modelo posee como entrada de control el voltaje que será aplicado a las llantas, las salidas son el desplazamiento vertical de las llantas y la posición angular del chasis. El sistema del robot se aprecia en la figura 3.15.

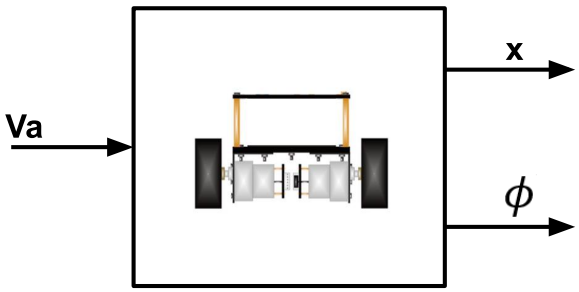


Figura 3.15: Diagrama de bloques de entradas y Salidas para el control del sistema robot autobalanceado péndulo invertido.

Basado en el modelo en espacio de estados (ecuaciones 3.30 y 3.31) se realizó una simulación utilizando Matlab con los valores del robot usado (tabla 1), para observar varias características del sistema como el comportamiento del sistema en lazo abierto, determinar si es controlable,

etc. La simulación del sistema a lazo abierto hecha en Simulink se presenta en la figura 3.16 y sus resultados en la figura 3.17.

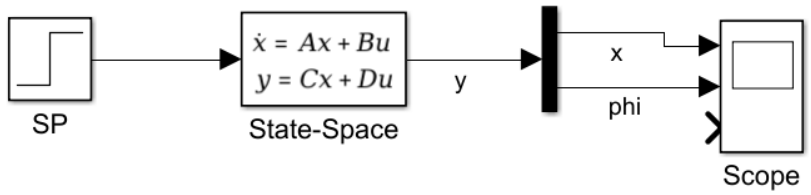


Figura 3.16: Simulación del Sistema en lazo abierto.

Al realizar la simulación a lazo abierto se pudo apreciar que se trata de un sistema inestable, al aplicar la entrada impulso se observó que las salidas del sistema se elevan de manera descontrolada en apenas 3 segundos de tiempo.

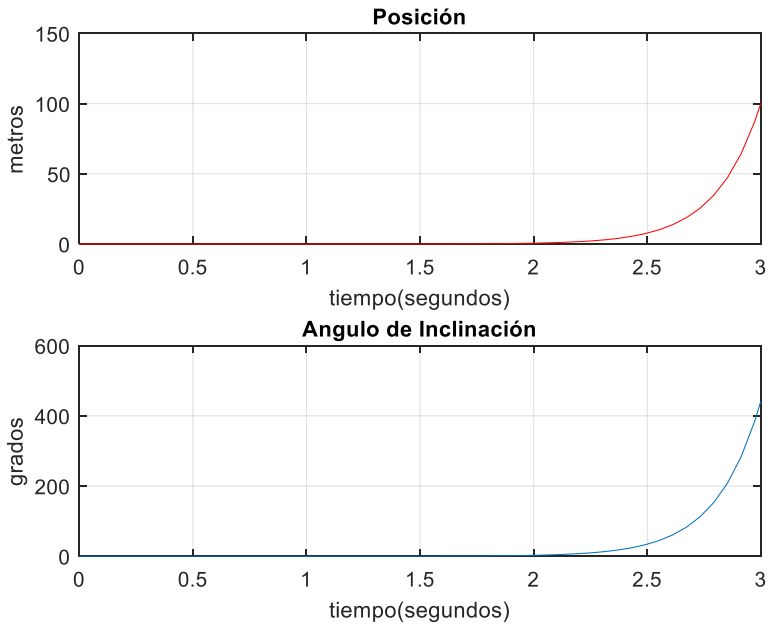


Figura 3.17: Salidas del Sistema simulado en lazo abierto.

Esta inestabilidad se debe a un polo en el lado derecho del lugar geométrico de las raíces como se observa en la figura 3.18.

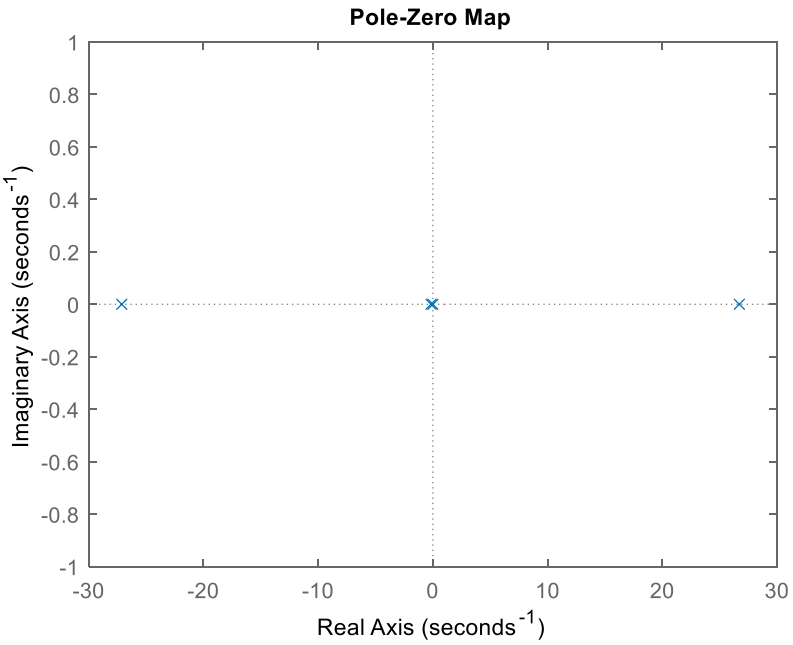


Figura 3.18: Lugar geométrico de las raíces.

Es necesario antes de empezar a desarrollar el diseño de un controlador conocer si el sistema es controlable, usando Matlab con el comando “ctrb(A,B)” la matriz de controlabilidad y su rango se calculó para determinar si es controlable, con un archivo .m desarrollado para el sistema de equilibrio se obtuvo como resultado que el sistema es controlable, como se observa en la figura 3.19

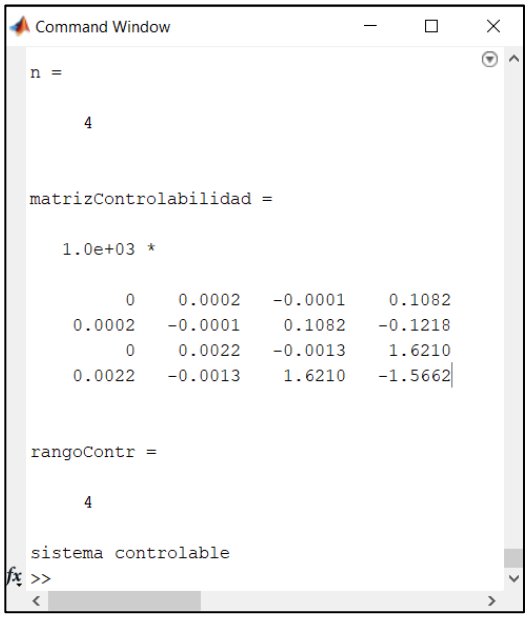


Figura 3.19: Análisis de Controlabilidad del sistema.

Al desarrollar la simulación del sistema a lazo cerrado, la búsqueda de estabilizar el sistema con un controlador PID fue infructífera, al revisar literatura como [21], donde habla de la posibilidad de implementar un control PD para este sistema por lo críticamente inestable, se decidió separar las variables de salida para diseñar el compensador para el ángulo ϕ que es lo más importante para alcanzar el equilibrio del robot, a partir de modelo en espacio de estados separamos las salidas a partir de la entrada y se obtuvo la siguiente función de transferencia:

$$\phi = \frac{2.236 + 2.493e^{-16}}{s^3 + 0.5629\, s^2 - 724.8s - 115.5}\, V_a \tag{3.32}$$

Con la ayuda del tool de Matlab “sisotool” se diseñó un compensador tipo PD, con función de transferencia [22]:

$$C(s) = K_p(1 + t_d\, s) \tag{3.33}$$

La figura 3.20 muestra la ecuación 3.32 representada en “sisotool”, donde a partir del lugar geométrico de las raíces se diseñó un compensador capaz de estabilizar el Sistema en menos de 1 segundo debido a la rapidez que se necesita en el control para evitar que el robot pierda el

equilibrio, como requerimiento está el que no exista overshoot en la respuesta al escalón, pues se desea una respuesta rápida y sin oscilación, el resultado al añadir un compensador PD se observa en la respuesta al escalón, donde es visible, un tiempo de estabilización pequeño, no hay presencia de sobreimpulso.

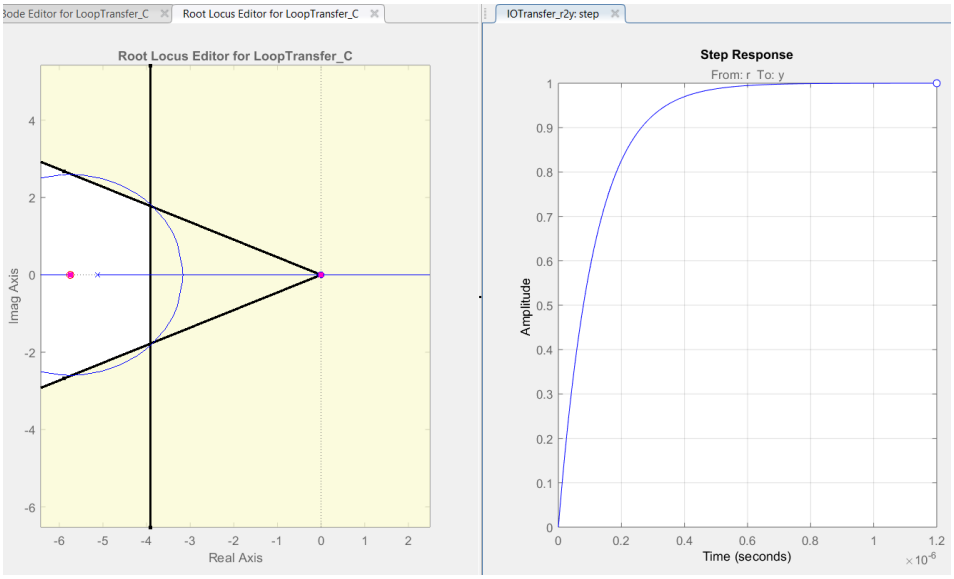


Figura 3.20: Diseño del compensador en SISOTOOL de Matlab.

Este controlador debe funcionar para controlar el sistema completo pues, como tal, se trata de un sistema MIMO, la simulación del controlador se ilustra en la figura 3.21.

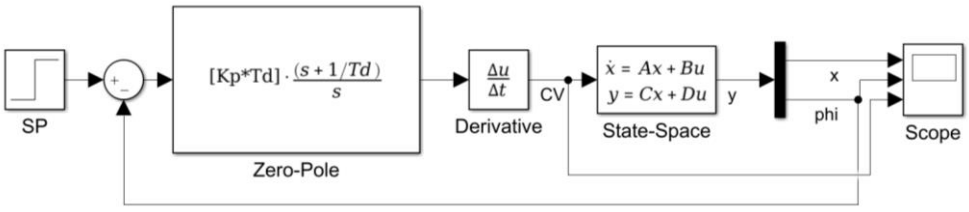


Figura 3.21: Simulación del sistema en lazo cerrado.

En la figura 3.22 se muestra el resultado al controlar el ángulo de inclinación deseado y se ve claramente el trabajo del controlador derivativo. Se logró un tiempo de estabilización de 1.5 segundos, 0% de sobre impulso.

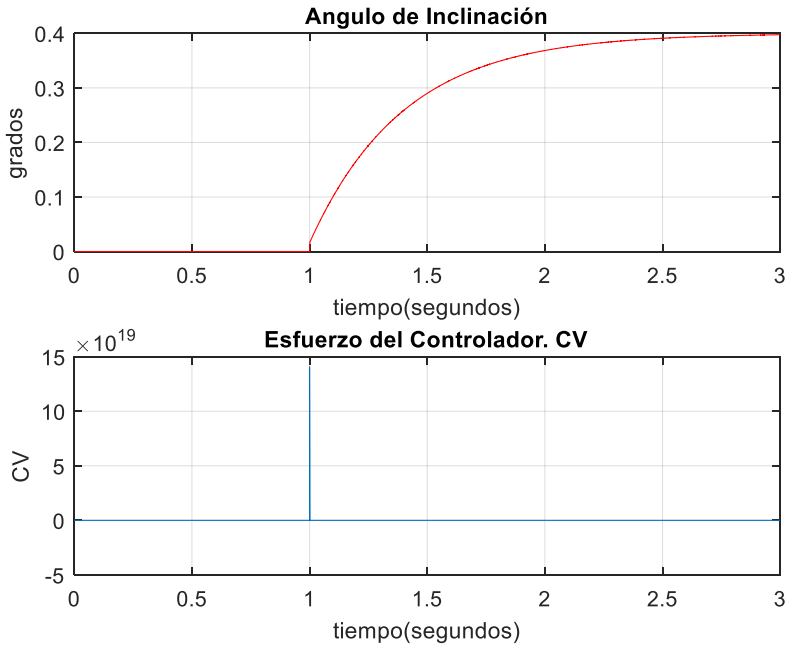


Figura 3.22: Ángulo de inclinación en lazo cerrado.

Con estas simulaciones trabajadas sobre el sistema del robot autobalanceado péndulo invertido, se procede a implementar el algoritmo de control en la tarjeta Arduino, para ello es necesario determinar cuáles son los valores de consigna, variable a controlar y variable controlada. La figura 3.23 muestra el diagrama de bloques de un lazo de control cerrado. El sistema a controlar está representado por el bloque G, el compensador PD bloque C, la variable a controlar el ángulo de inclinación (ϕ), la realimentación H proviene del ángulo (ϕ) obtenido del filtro de Kalman, (u) representa el valor deseado o de consigna que en este caso es $0°$ de inclinación, la variable controlada o CV se refiere al valor que permitirá controlar el voltaje suministrado a los motores para variar la velocidad con las que se moverán la llantas hacia adelante y atrás para equilibrar el robot, esto se da mediante la variación del ancho de pulso PWM.

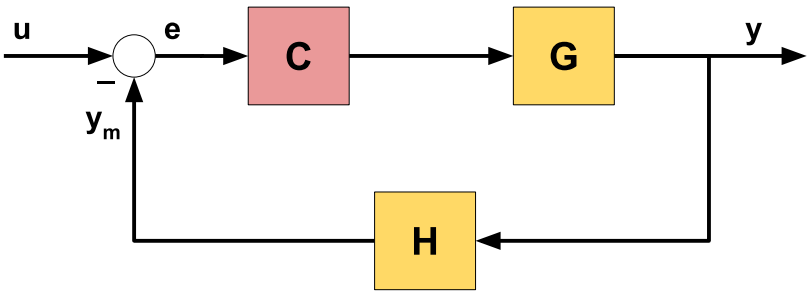


Figura 3.23: Sistema de Control en lazo cerrado.

3.5.2 Control de Trayectoria

Como se explicó en el capítulo 2 el robot autoabalanceado de acuerdo con la clasificación cinemática de los robots es un autómata que describe una locomoción con guiado diferencial, siguen una trayectoria recta o giran dependiendo de la velocidad aplicada a sus llantas. Las coordenadas (x, y) definen la posición respecto a las coordenadas globales y el ángulo $(\varphi \; o \; YAW)$ la orientación. La configuración por guiado diferencial se ilustra en la figura 3.24.

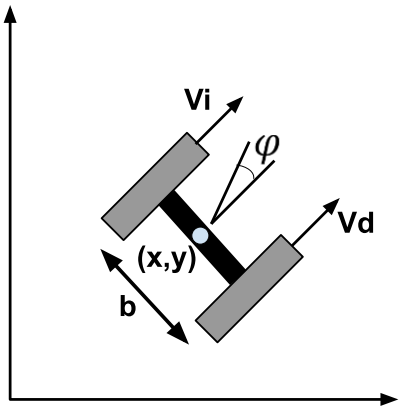


Figura 3.24: Configuración de la cinemática del robot por guiado diferencial [3].

Para determinar la posición y orientación del vehículo es necesario integrar la ecuación que definen el modelo jacobiano de la cinemática del robot diferencial (ecuación 3.34) [23].

$$\begin{bmatrix} x' \\ y' \\ \varphi' \end{bmatrix} = \begin{bmatrix} -sen\ \varphi & 0 \\ cos\ \varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.34}$$

Integrando la ecuación 3.34, conociendo la posición y orientación inicial $p_0 = [x_0 \quad y_0 \quad \varphi_0]^*$:

$$\begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \varphi_0 \end{bmatrix} + \begin{bmatrix} \int_0^t -v\ sen(\varphi)\ dt \\ \int_0^t v\ cos(\varphi)\ dt \\ \int_0^t \omega\ dt \end{bmatrix} \tag{3.35}$$

La ecuación 3.35 expresada para el caso de direccionamiento diferencial:

$$\begin{bmatrix} x \\ y \\ \varphi \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \varphi_0 \end{bmatrix} + \begin{bmatrix} \int_0^t \frac{-c\ sen(\varphi)}{2}(\omega_d + \omega_i)dt \\ \int_0^t \frac{c\ cos(\varphi)}{2}(\omega_d + \omega_i)dt \\ \int_0^t \frac{c}{b}(\omega_d - \omega_i)dt \end{bmatrix} \tag{3.36}$$

Donde:

c: radio de las llantas.

b: distancia entre las llantas.

ω_i, ω_d: velocidad angular de las llantas izquierda y derecha respectivamente.

La representación de la ecuación 3.36 en Simulink se presenta en la figura 3.25. Donde $c = 3.25\ cm$, $b = 14.5\ cm$, y $p_0 = (x_0 = 0 \quad y_0 = 0 \quad \varphi_0 = 0°)$.

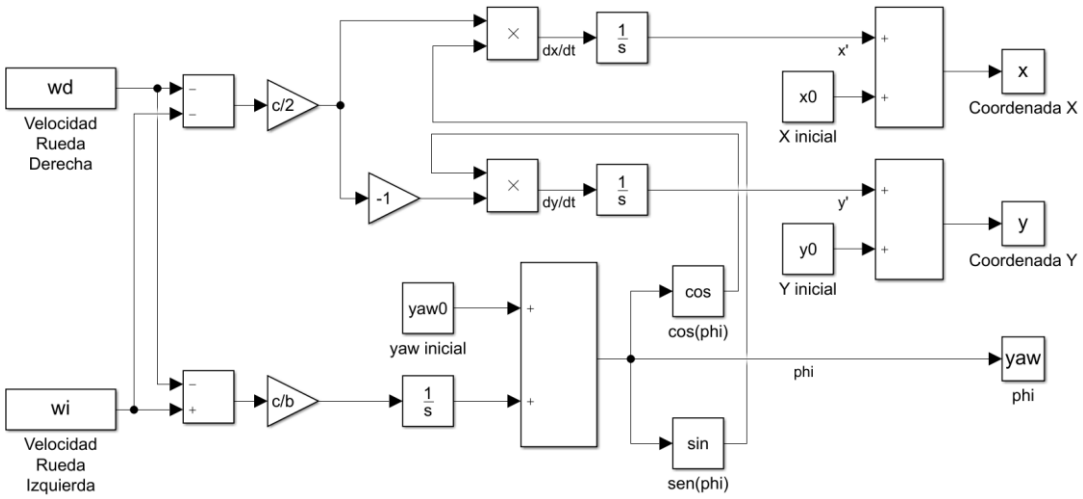


Figura 3.25: Diagrama en Simulink para determinar la orientación y posición de robots móviles con guiado diferencial.

La figura 3.26 expone el resultado de la simulación para el caso donde las velocidades angulares de la llanta izquierda y derecha tienen la misma velocidad angular $\omega_i = \omega_d = 1(\mathrm{rd/sg})$, por lo cual el robot avanza en línea recta a partir del punto inicial. La orientación (φ) descrita por el robot al recorrer la trayectoria, se puede apreciar que no existe variación en el ángulo pues al avanzar en línea recta no gira.

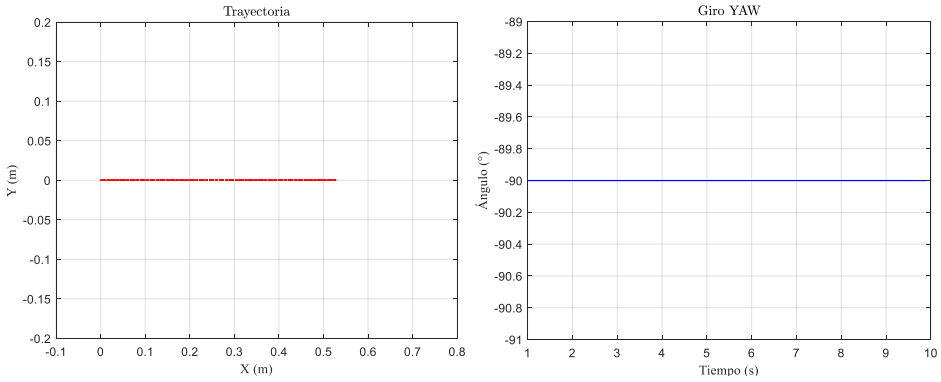


Figura 3.26: Orientación y posición de robot con guiado diferencial con velocidades angulares iguales $\boldsymbol{\omega_i = \omega_d = 1(\mathrm{rd/sg})}$.

La figura 3.27 presenta en caso donde al aplicar la misma velocidad angular a las llantas, pero en sentidos opuestos, el vehículo es capaz de girar sobre su propio eje.

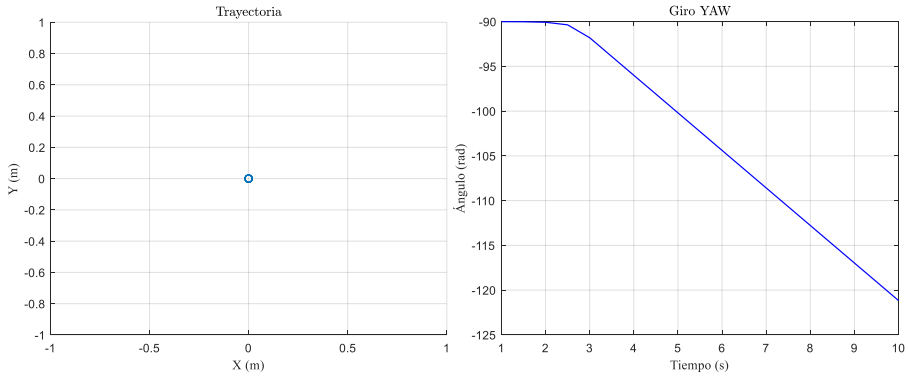


Figura 3.27: Orientación y posición de robot con guiado diferencial con velocidades angulares iguales con en sentidos opuestos.

Para implementar este control se define las variables medidas y controladas, de la misma manera que se hizo para el control de equilibrio, definiendo el lazo cerrado de la figura 3.23. El set point SP es el valor deseado de velocidad máximo con el que girarán las ruedas, el valor de proceso PV es la velocidad angular medida de los encoder, la variable a controlar CV la velocidad de las ruedas como en los 3 controladores implementados en este robot. Cada control aporta al cálculo de uno solo valor de salida de voltaje a las llantas.

Para este tipo de sistema de velocidad se aplicó un PI pues es el mejor tipo de control para la velocidad, la ecuación 3.37 define su funcionamiento:

$$C(t) = K_p\, e(t) + K_i \int_0^t e(t)\, dt \tag{3.37}$$

3.5.3 Control de Giro

En este control el punto de consiga es un valor del ángulo pequeño que girará el robot, aproximadamente 1°; cada vez que se presente la orden de girar se accionará el control de giro y continuará girando, aumentando el desplazamiento angular.

El valor controlado es el ancho de pulso PWM enviado a los motores.

El valor de proceso PV es el ángulo $(\varphi \ o \ YAW)$ proveniente del giroscopio.

Para controlar el giro del robot se aplicó como controlador uno de tipo PD, debido a que el robot está equipado con objetos pesados como su misma estructura, caja de baterías, etc. Posee un gran valor de inercia lo que podría causar sobre impulso en el control y causar que se pierda el equilibrio. Con el fin de evitar este inconveniente es necesario proveer de una acción diferencial sobre el ángulo controlado.

La figura 3.28 presenta la pantalla en Visual Basic mediante la cual se puede acceder a enviar comandos vía bluetooth para accionar los controles de trayectoria hacia adelante, atrás o girar.

Figura 3.28: Ventana en Visual Basic para accionar controles de trayectoria y giro.

CAPÍTULO 4

4. ÁNALISIS DE RESULTADOS.

Para analizar el desempeño del estimador lineal y los algoritmos de control desarrollados, se diseñaron pruebas, toma de datos y gráficas comparativas para establecer si se cumplió con los objetivos trazados al emprender este proyecto.

Lo primordial a revisar es el funcionamiento de los compensadores de equilibrio, velocidad y la fusión del giroscopio y acelerómetro con el Filtro de Kalman, presentados en este capítulo.

4.1 Funcionamiento del Filtro de Kalman

El correcto funcionamiento e implementación de este estimador depende de muchos factores determinantes, entre ellos el tiempo de muestreo, la covarianza del ruido de medición, las matrices de covarianza de error de estados estimados.

Los valores más importantes por considerar son el ruido de la medición y las covarianzas de error estimadas, debido a que, son quienes determinan quien domina la salida del estimador, si la medición o la estimación a partir del modelo del sistema. Esta afirmación está sustentada en el análisis presentado a partir de la ecuación 2.23 que define la ganancia de Kalman, donde a partir del estudio del denominador se conocerá si el estado estimado de la predicción o la medición tiene menos error y logra el objetivo de reducir error cuadrático lineal.

El ruido de la medición y las covarianzas de error estimadas son importantes en el análisis del desempeño del filtro de Kalman, pues si estas cantidades no son las correctas, se obtendrán lecturas erróneas de los sensores evitando eliminar el ruido del acelerómetro o la desviación en el giroscopio.

Los datos tomados por la tarjeta Arduino fueron enviados por comunicación serial bluetooth hacia Visual Basic, donde se pueden observar las gráficas. Para este análisis se guardó estos vectores de lecturas y se enviaron a Matlab para procesarlos y poder mejorar su presentación, mejorando así la apreciación.

Empezamos por representar el ángulo filtrado con los valores que por búsqueda literaria y cálculo de ruido encontramos para R y Q, asumiendo que son los valores correctos. La figura 4.1 muestra las lecturas del filtro para un ángulo de inclinación $\theta = 3°$, con valores $R = 0.035$, $Q_\theta = 0.001$ y $Q_{\dot{\theta}_b} = 0.003$ mismas que como se puede apreciar filtran el ángulo deseado, pudiendo tomarlas como aceptables.

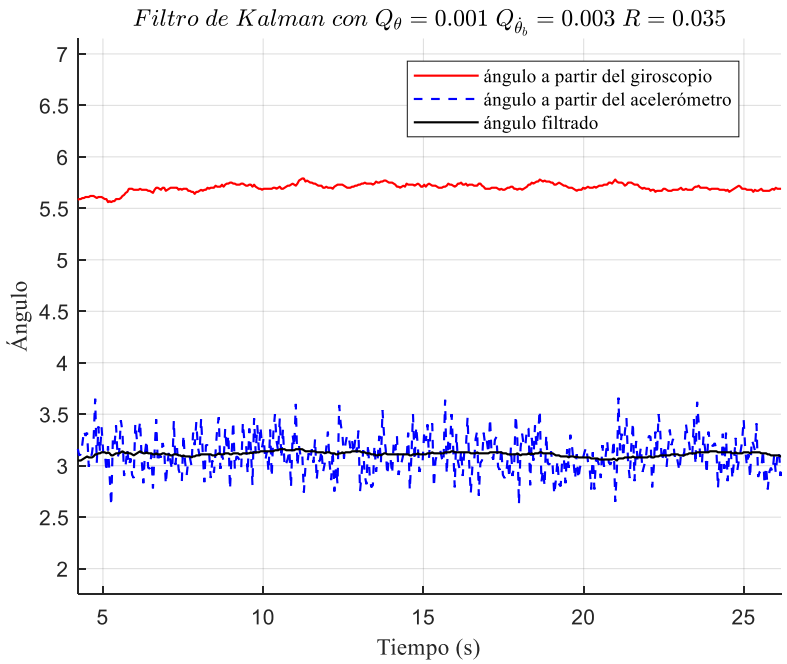


Figura 4.1: Lecturas con valores para R=0.035, Q_θ=0.001 y $Q_{\dot{\theta}_b}$=0.003

Como se explicó la matriz de covarianza de ruido del modelo en muchos casos es asumida con cuantía cero pues toma parte en la ecuación 2.29, donde se calcula el error estimado de la predicción $\boldsymbol{P_k^-}$ el cuál por sí mismo debiese ser mínimo o tender a cero en las actualizaciones del tiempo ya que en un proceso donde todas las variables han sido correctamente modeladas puede llegar a valores muy pequeños. Para nuestro caso se tomó como valido $\boldsymbol{Q_\theta = 0.001}$ y $\boldsymbol{Q_{\dot{\theta}_b} = 0.003}$ debido a que son valores pequeños y provocaron un buen desempeño del filtro.

El único valor que podemos manipular para el estudio del denominador de la ganancia de Kalman es la varianza del ruido de la medición del acelerómetro R,

empezamos con valores mayores a $R = 0.035$, el cual se tomó como válido. Las figuras 4.2 y 4.3 reflejan que cuando el ruido de la medición es alto el filtro no confiará en dichas lecturas y deberá tomar como válido el estado estimado de predicción, sin embargo, al ser alto el ruido provoca que el filtro se vuelva lento, mientras más ruidoso más lento responde el filtro. La figura 4.2 muestra como el observador se demora en empezar a filtrar, pero finalmente comienza a trabajar, caso opuesto en la figura 4.3 donde el ruido es demasiado alto comparado con el valor óptimo $R = 0.035$, pasados 25sg de experimento no logró filtrar el valor del ángulo.

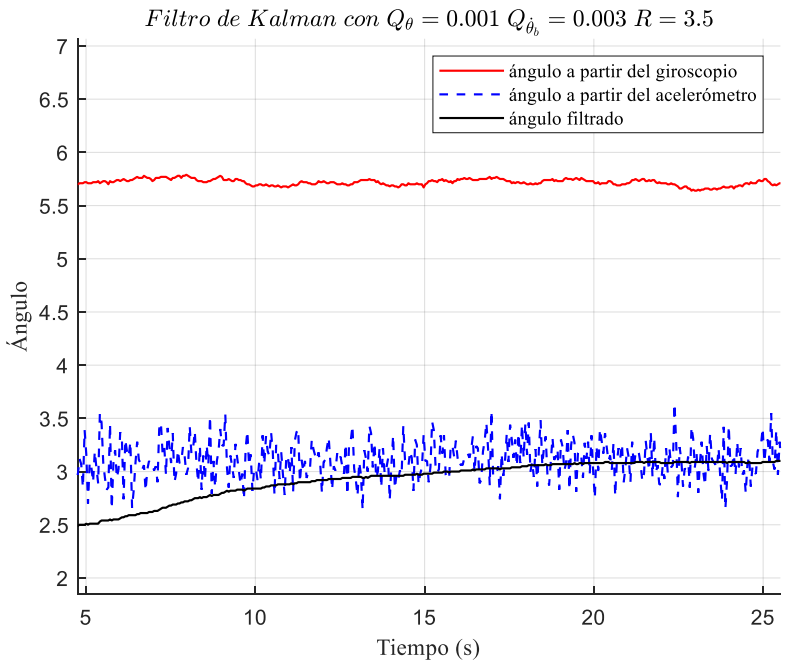


Figura 4.2: Lecturas con valores para R=3.5, Q_θ=0.001 y $Q_{\dot{\theta}_b}$=0.003

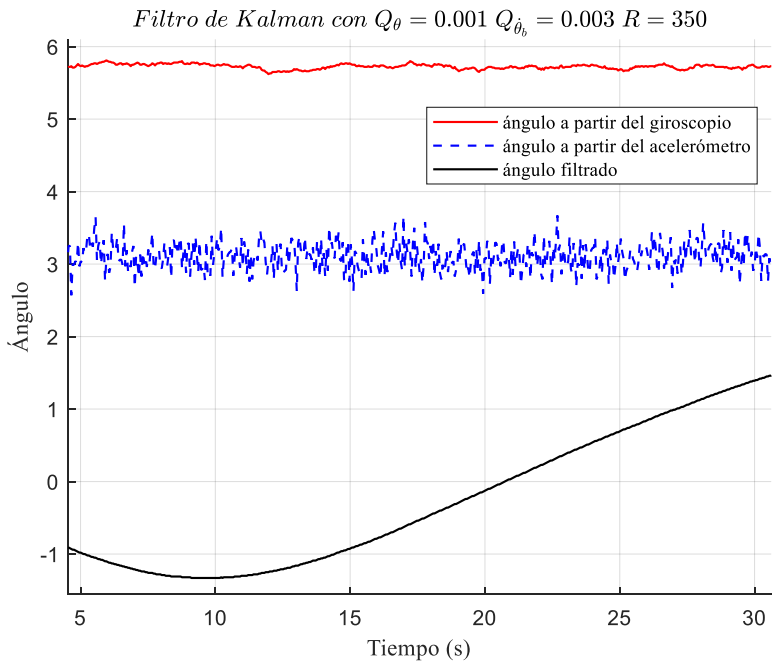


Figura 4.3: Lecturas con valores para R=350, Q_θ=0.001 y $Q_{\dot{\theta}_b}$=0.003

Las figuras 4.4 y 4.5 muestran el caso contrario valores pequeños para el ruido de la medición donde mientras esta medida es menor, cercana al cero, el filtro confía más en la medición del acelerómetro, pues asume que no existe ruido presente en esta medición, como se observa en ambas figuras el valor del filtro se parece al del acelerómetro, más distinguible en la figura 4.5 donde la salida del filtro es tan ruidosa como la medida por el acelerómetro. De esta manera podemos constatar como el estimador de acuerdo con las medidas de error calcula quien produce menor error cuadrático y lo toma como para su salida.

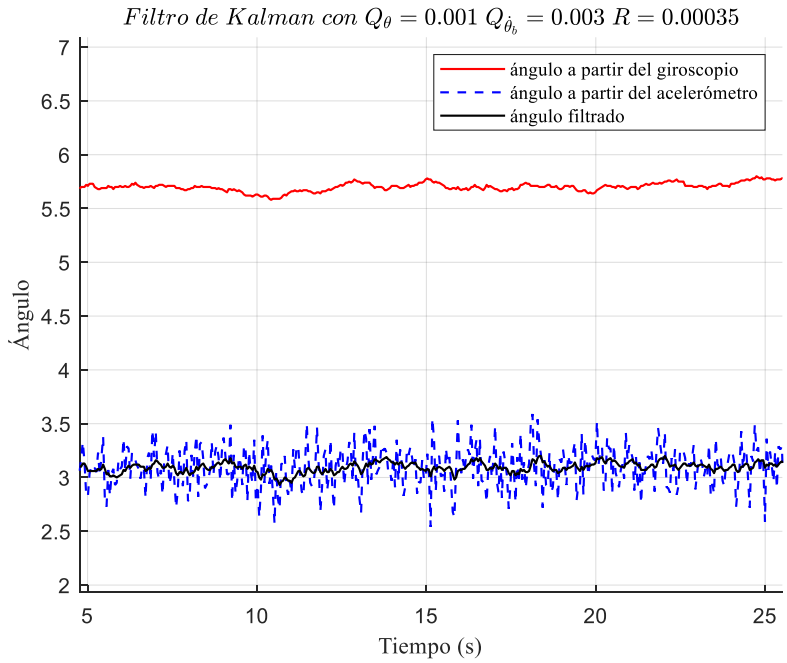


Figura 4.4: Lecturas con valores para R=0.00035, Q_θ=0.001 y $Q_{\dot{\theta}_b}$=0.003

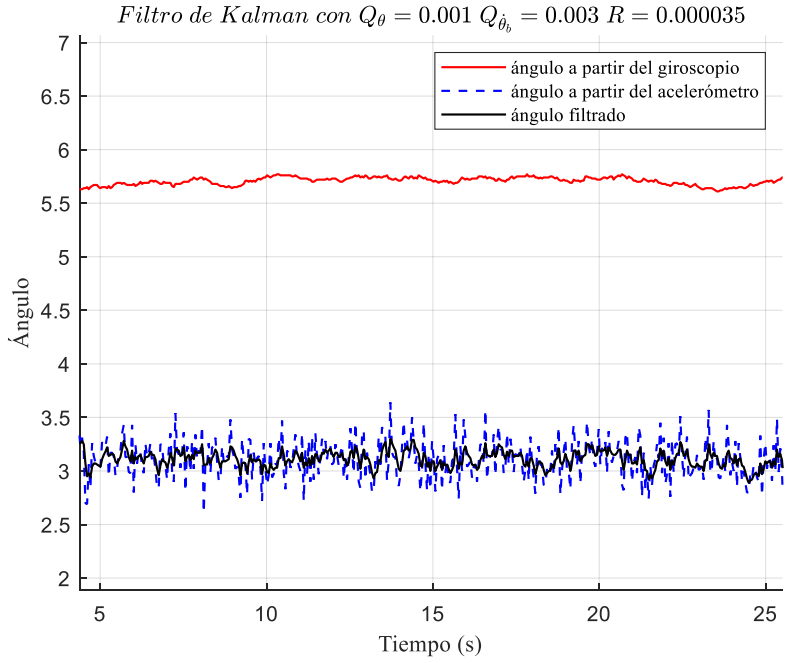


Figura 4.5: Lecturas con valores para R=0.000035, Q_θ=0.001 y $Q_{\dot{\theta}_b}$=0.003

4.2 Comportamiento de los algoritmos de Control

Para comprobar el funcionamiento de los lazos de control se hace para cada caso un análisis del ángulo de inclinación (ϕ), velocidad angular de las llantas (ω), posición (x, y), ángulo de giro YAW (φ).

4.2.1 Desempeño del Control de Equilibrio

La implementación del control Proporcional Derivativo PD está basado en la simulación del sistema tratada en el capítulo 3. Este control se probó con las constantes obtenidas en la simulación del sistema, debido a que el modelo dinámico desarrollado no es exacto en su totalidad, estas ganancias sirvieron como base para ajustar el controlador, finalmente la ecuación 3.33 que define el control PD implementado es:

$$C(s) = 50(1 + 0.6s)$$

Las figuras 4.6 a 4.8 presentan los resultados de los estados, esfuerzo del controlador del sistema real controlado con el compensador PD.

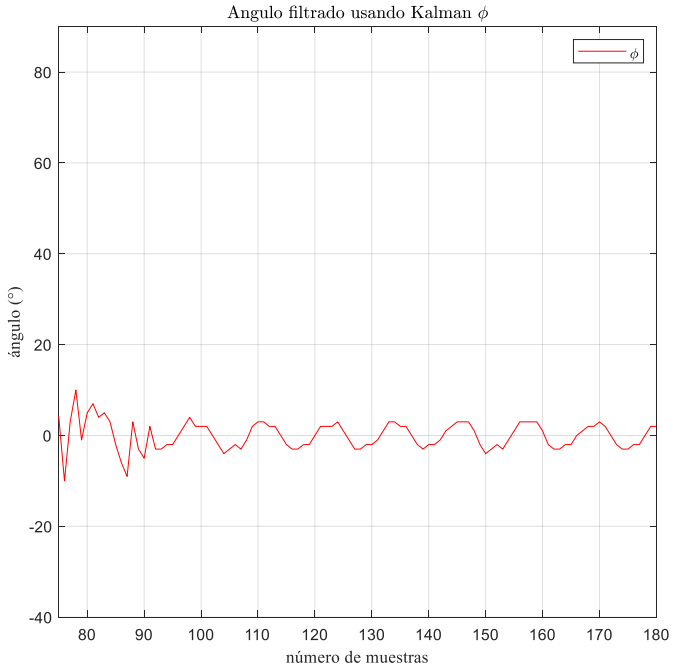


Figura 4.6: Datos en Tiempo Real control Proporcional Derivativo (Ángulo de inclinación ϕ).

La respuesta del estado (ϕ) dista un poco de la respuesta obtenida en la simulación (figura 3.22), el tiempo de asentamiento del sistema real es 2 segundos, presenta oscilaciones hasta estabilizar el sistema, finalmente el ángulo de inclinación converge a cero, esto se observa en la figura 4.6 donde cada 10 muestras representan 1 segundo. El comportamiento del sistema presenta diferencias del modelo simulado, debido entre muchos factores que no siempre se trabaja con valores idénticos a la vida real y este modelo no asumía parámetros como deslizamiento de las llantas en el piso, fricción o las siempre presentes perturbaciones.

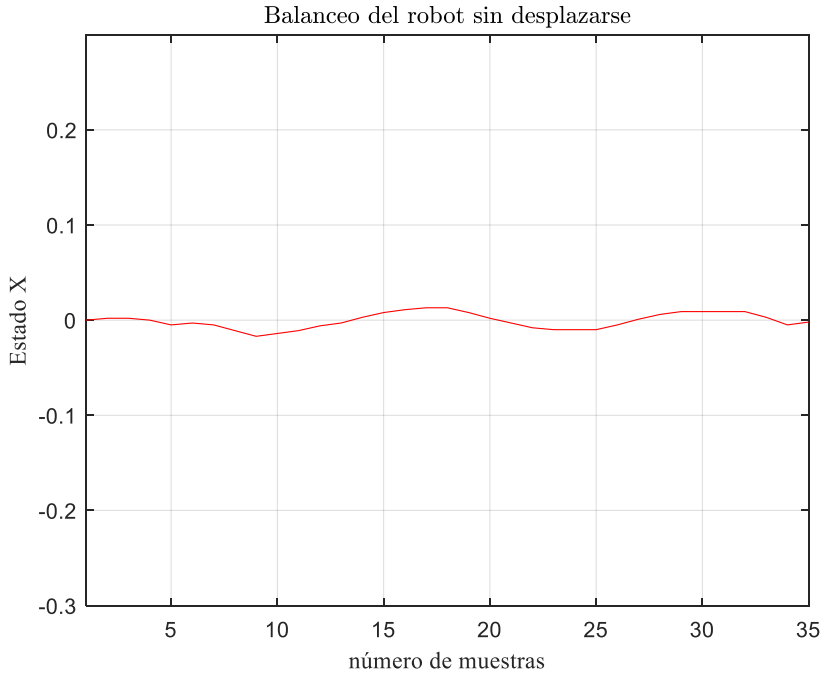


Figura 4.7: Datos en Tiempo Real control Proporcional Derivativo (x).

Las constantes de compensador estaban enfocadas en controlar el estado (ϕ), pues es la variable imprescindible para alcanzar el equilibrio, la figura 4.7 muestra el estado (x) correspondiente a la posición, misma que se mantiene en cero mientras el robot se balancea, mostrando que pese a tener más ponderación el control del ángulo de inclinación, una vez equilibrado el sistema, el controlador es capaz de controlar la posición.

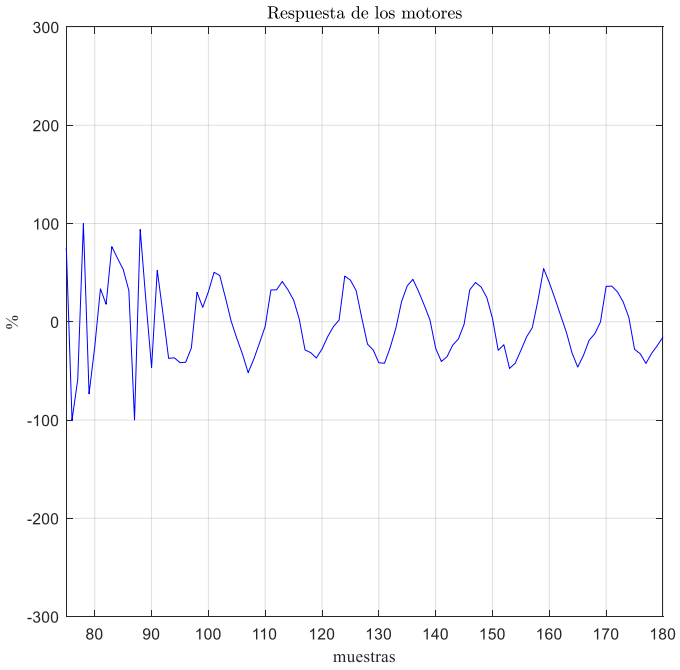


Figura 4.8: Datos en Tiempo Real control Proporcional Derivativo (% acción del motor).

4.2.2 Desempeño del Control de Trayectoria

Las ganancias para los controladores de trayectoria y giro se obtuvieron aplicando el método de sintonización Ziegler y Nichols de lazo cerrado [24].

1. Asignar valor a K_p hasta que presente una oscilación sostenida pequeña. Obteniendo en este contexto la ganancia límite (o de oscilación) K_{cu} y el período de oscilación P_n.
2. Ajustar el controlador de acuerdo con los parámetros

$$K_p = 0.6K_{cu}$$

$$T_i = 0.5\,P_n, \qquad K_i = \frac{K_p}{T_i}$$

$$T_d = 0.125\,P_n, \qquad K_d = K_p T_d$$

Las constantes del controlador Proporcional Integral PI para controlar la velocidad en el desplazamiento de acuerdo con la ecuación 3.37 son:

$$K_p = 5 \qquad K_i = 0.15$$

En las figuras 4.9 a 4.11 se muestra los resultados alcanzados al aplicar el control PI encargado de verificar y actuar sobre la velocidad del robot.

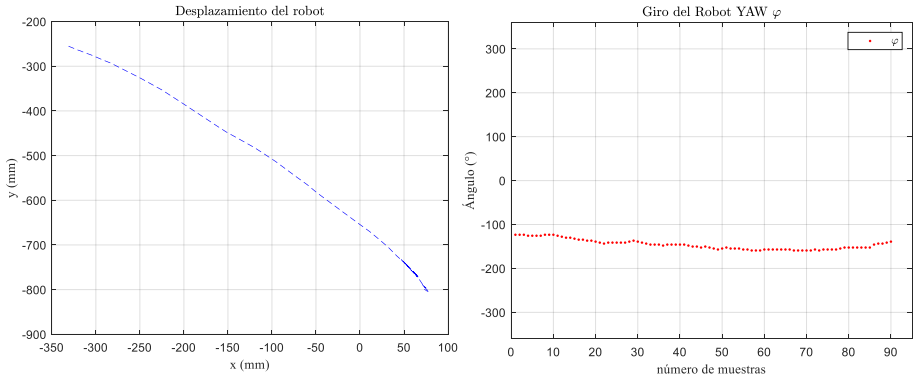


Figura 4.9: Datos en Tiempo Real control Proporcional Integral para la velocidad angular de las llantas (x, φ).

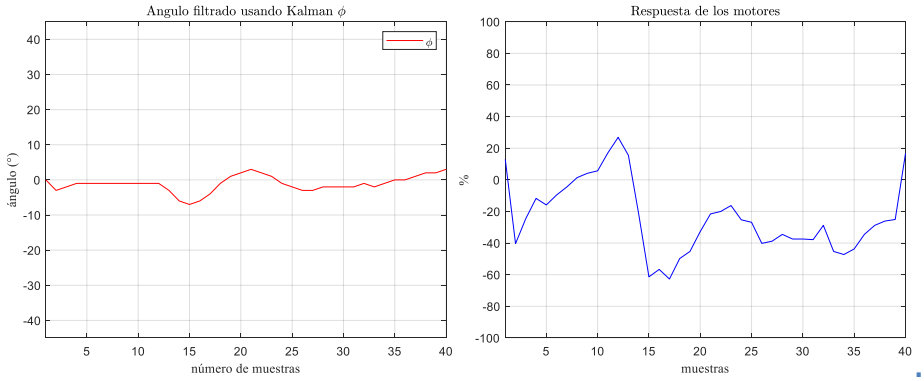


Figura 4.10: Datos en Tiempo Real control Proporcional Integral para la velocidad angular de las llantas $(\phi, \%\ acción\ del\ motor)$.

En la simulación para el control de trayectoria los estados primordiales son el desplazamiento (x) y giro del robot (φ ángulo YAW), pues se desea que siga una trayectoria en línea recta, los resultados obtenidos son parecidos a los simulados, se aprecia que al enviar vía bluetooth que se dé la marcha hacia adelante la trayectoria descrita es una línea bastante recta y no existe una mayor variación en el ángulo de giro. Para el sistema real se hace necesario también verificar que el autómata no solo se desplace cumpliendo la premisa deseada, mientras controla la velocidad

angular de las llantas es imperativo supervisar y controlar el ángulo de equilibrio para evitar un desbalance, la figura 4.10 muestra que el ángulo (ϕ) se mantiene en cero mientras se da el control de trayectoria, el esfuerzo del controlador difiere de la manera de controlar cuando se trata únicamente del control de equilibrio donde la acción proporcional y derivativa se aprecia mejor, en este caso se observa la acción PD cuando hay serias variaciones en el ángulo de inclinación y variaciones menos bruscas mientras controla la velocidad.

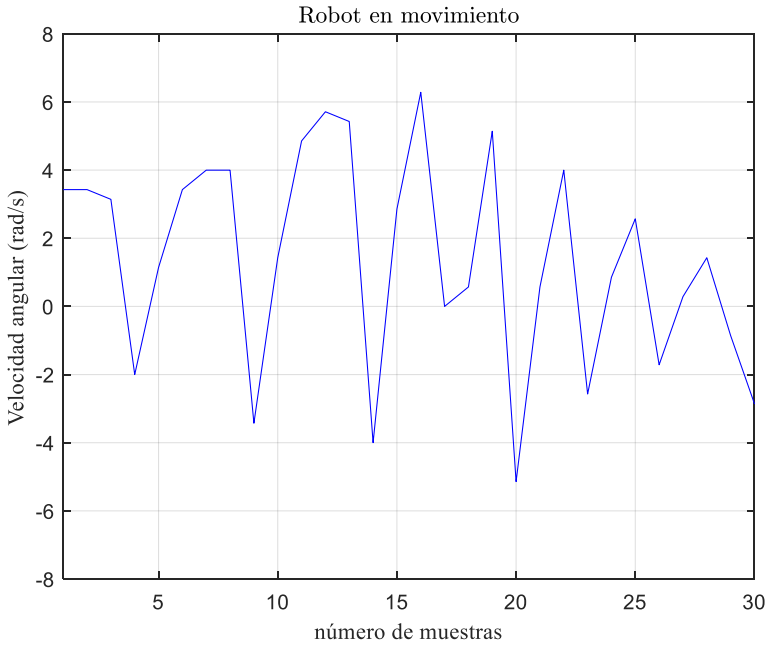


Figura 4.11: Datos en Tiempo Real control Proporcional Integral para la velocidad angular de las llantas ($\boldsymbol{\omega}$).

4.2.3 Desempeño del Control de Giro

Las constantes determinadas para este compensador son:

$$K_p = 15 \qquad K_d = 1.4$$

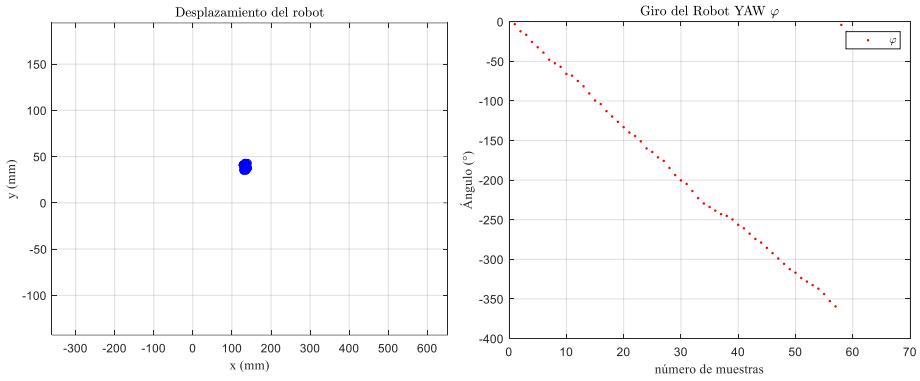


Figura 4.12: Datos en Tiempo Real control Proporcional Derivativo para giro (x, φ).

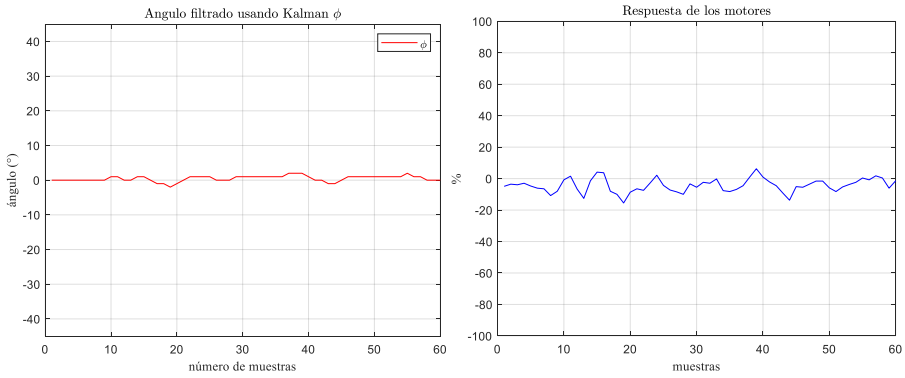


Figura 4.13: Datos en Tiempo Real control Proporcional Derivativo para giro $(\phi, \%$ acción del motor$)$.

Para comparar con los resultados de la simulación el robot giro una vez 360°, se observa que la trayectoria descrita es un punto en las coordenadas (x,y), pues eso se obtiene al girar las llantas a la misma velocidad y sentido contrario una de otra, comprobando una de las características principales del guiado diferencial, el ángulo (φ) muestra claramente los 360° girados. De la misma manera mientras este movimiento se desarrolló el robot mantuvo el equilibrio ángulo $\phi = 0°$ (figura 4.13).

CONCLUSIONES Y RECOMENDACIONES

El filtro de Kalman fue implementado exitosamente, se logró los hitos propuestos obtención de un modelo de medición correcto para fusionar el acelerómetro y giroscopio, una señal del ángulo de inclinación del robot sin ruido presente, eliminación de la desviación del giroscopio. En la obtención del modelo y su implementación lo más importante fue descubrir que el correcto funcionamiento del filtro depende de la buena estimación que se haga de la covarianza del ruido en el acelerómetro (R) y buen cálculo de error de la estimación del modelo, para ello fue necesario determinar el valor (R), mediante los conocimientos de estadística, ruido, procesos estocásticos y análisis frecuencial.

El control del equilibrio es el más importante en un robot autobalanceado. El uso del modelo matemático para determinar el tipo de controlador necesario fue de mucha ayuda, pues así se comprobó que pese a ser poco usado el control PD para este tipo de sistema era el adecuado para balancear el robot. De la misma forma la simulación mejoró el entendimiento del tipo de sistema tan inestable a tratar, fue posible observar en el lugar geométrico de las raíces los polos que causan la inestabilidad, y mejorar la idea del compensador necesario.

La comunicación vía bluetooth con Visual Basic para enviar comandos de giro o puesta en marcha funcionó de manera adecuada y ayudó también para guardar datos necesarios para las gráficas de las variables y su respectivo análisis, ayudando en cierta forma en la sintonización y supervisión del comportamiento del sistema.

El control Proporcional Derivativo usado para el equilibrio funcionó de manera correcta, manteniendo el valor del ángulo controlado en el punto deseado, más con el afán de mejorar el conocimiento del sistema y el control del ángulo de inclinación se recomienda utilizar métodos para el control de sistemas multivariable como el control LQR o ubicación de polos. Los compensadores PID aplicados al control de trayectoria cumplieron con las expectativas de mantener la posición y el ángulo de giro, si se desea mejorar los tiempos de respuesta o precisión es recomendable usar el modelo matemático y controladores basados en el mismo.

BIBLIOGRAFÍA

[1] E. Diez, "*Using random signals as part of a methodology for calibrating seismometers and accelerometers*", 2014

[2] D. Pozo, N. Sotomayor, J. Rosero y L. Morales, "*Medición de Ángulos de Inclinación por Medio de Fusión Sensorial Aplicando Filtro de Kalman",* Revista Politécnica, 33(1), 2014. Recuperado a partir de https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/144

[3] I. Maza y A. Ollero, *"Robótica. Manipuladores y Robots Móviles",*01 2001, pp. 422–440

[4] F. Corona, J. Abarca y C.J. Mares, "*Sensores y actuadores: Aplicaciones con Arduino*", 2015

[5] L. Amezquita, "*Modelado y Control de Giroscopio*", 2005

[6] MPU-6000/MPU-6050 Product Specification. INVENSENSE. EE. UU. (2013). PS-MPU-6000A-00. Revision 3.4.

[7] S. Salas. "*Conceptos básicos. Todo sobre sistemas embebidos"* (pp. 1-39), Lima: Universidad Peruana de Ciencias Aplicadas, 2015

[8] F. Bellido, J. De la Cruz, M. Torres y J. Gistas, "*Comunicación Inalámbrica con Bluetooth*", 2004. Revista Técnica Industrial Especial Eléctricidad y Electrónica. Recuperado a partir de: http://www.tecnicaindustrial.es/tiadmin/numeros/15/06/a06.pdf

[9] T. Kailath, A. H. Sayed y B. Hassibi, *"Linear Estimation*" IEEE Trans. Autom. Control, vol. 48, no. 1, pp. 177–180, Jan. 2003.

[10] G. Welch y G. Bishop, "*An introduction to the kalman filter*", USA,Tech. Rep., 1995

[11] R. Sira-Ramirez, F. Marquez, F. Rivas, y O. Santiago, *“Control de Sistemas No Lineales: Linealización aproximada, extendida, exacta*”, 2018

[12] A. Nevado Reviriego, P. Cabrera Cámara and J. Martín Sánchez, “*Conceptos básicos de filtrado, estimación e identificación*”, 2015.

[13] K. Lauszus, "*TKJ Electronics » A practical approach to Kalman filter and how to implement it"*, Blog.tkjelectronics.dk, 2012. [Online]. Available: http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/. [Accessed: 23- Jun- 2020].

[14] D. Vara. “*Sistemas para determinar la posición y orientación de herramientas quirúrgicas en operaciones de cirugía laparoscópica*”, España, 2014

[15] M. Aranda. “*Estudio y aplicación del Filtro de Kalman en fusión de sensores en UAVs*”, España, 2017

[16] F. González. “*Estudio y modelado de errores que afectan a una Unidad de Medidas Inerciales de bajo coste”,* España, 2016

[17] V. Alcaraz, R. Salazar, V. González y J.L. Gouzé. “*A Tunable Asymptotic Observer for Biochemical Reactors. Application to Anaerobic Digestion”,* Información Tecnológica-Vol. 15 N° 2-2004, págs.: 69-74. 2004

[18] S. Dominguez, P. Campoy, J. Sebastián y A, Jimenez. “*Control en espacio de estado*”, 2da Edición, Madrid, Pearson Education S.A, 2006

[19] M. Khaled, A. Mohammed, M. Ibraheem, y R. Ali, “*Balancing a two wheeled robot,”* Ph.D. dissertation, 07 2009.

[20] Ooi R. Chi, “*Balancing a Two-Wheeled Autonomous Robot*”, University of Western Australia, School of Mechanical Engineering, Final Thesis 2003

[21] Self Balancing Car. [Online] Disponible en: https://wiki.keystudio.com

[22] E. Tacconi, R. Mantz, J Solsona y P. Puleston. “*Controladores Basados en estrategias PID*”, 2005.

[23] A. Rañada, *“Dinámica Clásica*”, Alianza Universidad Textos, 1994.

[24] D. Chuck, *“Los sistemas de primero orden y los controladores PID*”, 2012.

ANEXOS

ANEXO A

Determinación de Observabilidad del modelo de medición para el Filtro de Kalman

```
clc
clear all

%A=[1.1 0.3 1.5;
%   0.1 3.5 2.2;
%   0.4 2.4 1.1 ]
%B=[0.1 0;
%   0   1.1;
%  1.0 1.0]
%C=[1 2 0;
%    0 0 1]
%D=[1 2;
%    0 1]

A=[1 0;
   0 1]
B=[0 0;1 0 ; 0 0 ;0 1 ];
C=[1 0 ]
D=[0 0 ];
[m,n]=size (A);

%observabilidad
observ=obsv(A,C);
rangoObs=rank(observ);

if rangoObs==n
   disp('sistema observable')
else
   disp('sistema no observable')
end
```

ANEXO B

Simulación del modelo de Robot Autobalanceado para el Control de Equilibrio.

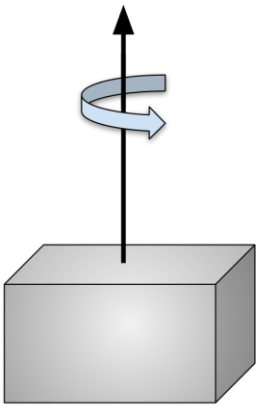
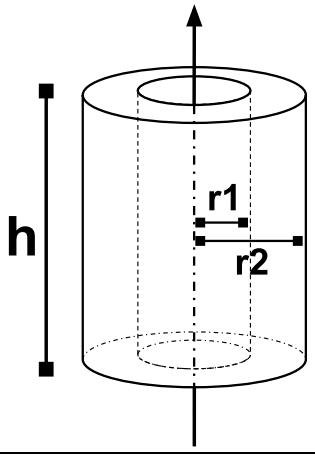
Parte	Peso (Kg)	Momento de Inercia (J)	Cálculo de J
Péndulo Invertido Chasis	**0.8**	$\frac{3}{210773}$	$J = \frac{1}{12} m(a^2 + l^2)$ 
Llanta	**0.04**	$\frac{1}{32000}$	$J = \frac{1}{2} m(r_1{}^2 + r_2{}^2)$ 

Tabla 1. Valores de la estructura del Robot para la simulación del control de equilibrio.

Simulación del modelo en Matlab

```
altura = 0.01; %altura desde el eje de ruedas hasta el arduino en metros
ancho = 0.012; %ancho del chasis en metros
mp=0.7; % masa del péndulo en kg
mw=0.04; % masa de las ruedas en kg
L=0.08; % Length to CG (Center of Gravity)
r=0.0325; % Radius of the wheel
r2=0.0225; % radio de la parte hueca de la llanta
ip=(1/12)*mp*((altura^2) + (ancho^2)); % inertia of the pendulum around the wheel axis
iw=(1/2)*mw*((r^2)+(r2^2)); % inertia of the wheel around the wheel axis
R=5.6; % The resistance of the DC Motor
ke=0.0879; % Voltage constant for the DC Motor
km=0.0045; % Motor Torque Constant of the DC Motor
g=9.8; % Gravity Accel.

beta = 2*mw+((2*iw)/(r*r))+mp;
alpha = (ip*beta)+(2*mp*L*L*(mw+(iw/(r*r))));

a22=(2*km*ke*(mp*L*r-ip-mp*L*L))/(R*r*r*alpha);
a23=(mp*mp*g*L*L)/alpha;
a42=(2*km*ke*(r*beta-mp*L))/(R*r*r*alpha);
a43=(mp*g*L*beta)/alpha;
b21=(2*km*(ip+mp*L*L-mp*L*r))/(R*r*alpha);
b24=(2*km*(mp*L-r*beta))/(R*r*alpha);
a=[0 1 0 0;0 a22 a23 0;0 0 0 1;0 a42 a43 0];
b=[0 ;b21 ;0 ;b24];
c=[1 0 0 0;0 0 1 0];
d=[0;0];
Td=-2;
Kp=300;
sim('penduloBelen')

 states = {'xDot' 'xDDot' 'PhaiDot' 'PhDDot'};
inputs = ('Va');
outputs= {'x' 'Phai'};
sistema = ss(a,b,c,d,'statename',states,'inputname',inputs,'outputname',outputs)
twip_ss = sistema;
tf(twip_ss)

%revision de controlabilidad del sistema
[m,n]=size (a)
matrizControlabilidad=ctrb(a,b)
rangoContr=rank(matrizControlabilidad)
if rangoContr==n
   disp('sistema controlable')
else
   disp('sistema no totalmente controlable')
end
```

ANEXO C

Programa en la tarjeta Arduino

```
//Programa en Arduino para el balanceo de Robot de dos ruedas
//implementado Filtro de Kalman

#include <FlexiTimer2.h>
#include <BalanceCar.h>
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"

#include <math.h>

//pines driver puente H l298n
#define IN1 9
#define IN2 8
#define IN3 7
#define IN4 6
#define ENA 10
#define ENB 5

//interrupciones encoders
#define PinA_right 2  //rueda derecha
#define PinA_left 3   //rueda izquierda

//variables para calcular el angulo con el giroscopio
float elapsedTime = 0, currentTime, previousTime;

//instancia para usar el sensor mpu6050
MPU6050 mpu;

//instancia para librería que de los PIDs
BalanceCar balancecar;

//variables para las lecturas del mpu6050
int16_t ax, ay, az, gx, gy, gz;

//variables para el control
double Outputs = 0;
int valor = 0;

double kp = 50, ki = 0, kd = 0.6;                    //EQUILIBRIO
double kp_velocidad = 5, ki_velocidad = 0.15, kd_velocidad = 0.0;  //VELOCIDAD
double kp_giro = 15, ki_giro = 0, kd_giro = 1.4;          //GIRO

int valor_velocidad_recta = 100;        //Ajuste de velocidad recta

//Variable velocidad recta
double setp0 = 0;

//********************datos para el angulo*********************//
```

```
float K1 = 0.05; //peso para el valor del acelerometro
float angle0 = -1.00; //angulo de referencia
//********************datos para el angulo*********************//

//***************Valores filtro de KALMAN*********************//
float Q_angle = 0.001, Q_gyro = 0.005; //Ruido del sistema
float R_angle = 0.035; //Ruido de las mediciones
float dt = 0.005; //tiempo de muestreo para el filtro de kalman [ms]
float C_0 = 1; //variable auxiliar filtro de kalman
//***************Valores filtro de KALMAN*********************//

//***************Contador de vueltas*********************//
volatile long contador_rueda_derecha = 0;  //contador de vueltas para el encoder de la llanta
derecha
volatile long contador_rueda_izquierda = 0;   //contador de vueltas para el encoder de la llant
a izquierda
int speedcc = 0;
int timer = 0;

int lz = 0;
int rz = 0;
int rpluse = 0;
int lpluse = 0;
int sumam;
//***************Contador de vueltas*********************//

//***************Parametros para el giro*********************//
int contador_giro = 0; //Turn intervention time calculation
float salida_giro = 0;
//***************Parametros para el giro*********************//

//***************Control Bluetooth*********************//
int front = 0;//variable frente
int back = 0;//variable atras
int turnl = 0;//giro izquierda
int turnr = 0;//giro derecha
int spinl = 0;//rotar a la izquierda
int spinr = 0;//rotar a la derecha
//***************Control Bluetooth*********************//

char mensaje = "X"; //string que será leído desde visual basic

char i = 0;

//*****************************************************************
//********Variables para el filtro de Karman***********************
float Gyro_x, Gyro_y, Gyro_z;
float accelz = 0;
float angle; //angulo de salida del filtro de kalman
float angle6 = 0;

//calular angulo de giro a partir del acelerometro
float yaw = 0, delta_yaw;
```

```
float angle_err, q_bias;
//Matrices para el filtro de kalman
float Pdot[4] = { 0, 0, 0, 0};
float P[2][2] = {{ 1, 0 }, { 0, 1 }};
//Variables auxiliares para el filtro de kalman
float PCt_0, PCt_1, E, K_0, K_1, t_0, t_1;
float angle_dot;
//*****************************************************************

void RecibirDatos()
{

  if (Serial.available() > 0)
  {
    mensaje = Serial.read();
    switch (mensaje) {
      case '1': front = valor_velocidad_recta;   break;
      case '2': back = -valor_velocidad_recta;   break;
      case '3': turnl = 1; front = 0; back = 0;   break;                    //
      case '4': turnr = 1; front = 0; back = 0;   break;                    //
      case '5': spinl = 1; front = 0; back = 0;   break;                   //
      case '6': spinr = 1; front = 0; back = 0; break;                      //
      case '8': spinl = 0; spinr = 0;  front = 0; back = 0;  turnl = 0; turnr = 0;  break;
      case '0': front = 0; back = 0; turnl = 0; turnr = 0; spinl = 0; spinr = 0; salida_giro = 0; break;
       // Make sure the button is released for parking operation

      //PID1
      case 'Q': kp = kp + 1; break;
      case 'W': kp = kp - 1; break;
      case 'E': kp = kp + 0.1; break;
      case 'R': kp = kp - 0.1; break;
      case 'A': ki = ki + 1; break;
      case 'S': ki = ki - 1; break;
      case 'D': ki = ki + 0.1; break;
      case 'F': ki = ki - 0.1; break;
      case 'Z': kd = kd + 1; break;
      case 'X': kd = kd - 1; break;
      case 'C': kd = kd + 0.1; break;
      case 'V': kd = kd - 0.1; break;

      //PID2
      case 'q': kp_velocidad = kp_velocidad + 1; break;
      case 'w': kp_velocidad = kp_velocidad - 1; break;
      case 'e': kp_velocidad = kp_velocidad + 0.1; break;
      case 'r': kp_velocidad = kp_velocidad - 0.1; break;
      case 'a': ki_velocidad = ki_velocidad + 1; break;
      case 's': ki_velocidad = ki_velocidad - 1; break;
      case 'd': ki_velocidad = ki_velocidad + 0.1; break;
      case 'f': ki_velocidad = ki_velocidad - 0.1; break;
      case 'z': kd_velocidad = kd_velocidad + 1; break;
      case 'x': kd_velocidad = kd_velocidad - 1; break;
      case 'c': kd_velocidad = kd_velocidad + 0.1; break;
```

```
      case 'v': kd_velocidad = kd_velocidad - 0.1; break;

      //PID3
      case 't': kp_giro = kp_giro + 1; break;
      case 'y': kp_giro = kp_giro - 1; break;
      case 'u': kp_giro = kp_giro + 0.1; break;
      case 'i': kp_giro = kp_giro - 0.1; break;
      case 'g': ki_giro = ki_giro + 1; break;
      case 'h': ki_giro = ki_giro - 1; break;
      case 'j': ki_giro = ki_giro + 0.1; break;
      case 'k': ki_giro = ki_giro - 0.1; break;
      case 'b': kd_giro = kd_giro + 1; break;
      case 'n': kd_giro = kd_giro - 1; break;
      case 'm': kd_giro = kd_giro + 0.1; break;
      case ',': kd_giro = kd_giro - 0.1; break;

      default: front = 0; back = 0; turnl = 0; turnr = 0; spinl = 0; spinr = 0; salida_giro = 0; break;
    }
  }
}

void countpluse()
{

  lz = contador_rueda_izquierda;    //lee el valor acumulado del encoder derecho
  rz = contador_rueda_derecha;   //lee el valor acumulado del encoder izquierdo

  contador_rueda_izquierda = 0;     //reinicia los acumuladores de los encoder
  contador_rueda_derecha = 0;

  //reasigna los valores de los encoder a nuevas variables
  lpluse = lz;
  rpluse = rz;

  //Juicio de la dirección del movimiento del automóvil hacia atrás (PWM que es el voltaje neg
ativo del motor) el número de pulso es negativo
  if ((balancecar.pwm1 < 0) && (balancecar.pwm2 < 0))
  {
    rpluse = -rpluse;
    lpluse = -lpluse;
  }
  //Cuando la dirección de movimiento del automóvil para juzgar hacia adelante (PWM que es
 voltaje de motor positivo) el número de pulso es positivo
  else if ((balancecar.pwm1 > 0) && (balancecar.pwm2 > 0))
  {
    rpluse = rpluse;
    lpluse = lpluse;
  }
  //
  else if ((balancecar.pwm1 < 0) && (balancecar.pwm2 > 0))
  {
    rpluse = -rpluse;
    lpluse = lpluse;
```

```
  }
  //La dirección del movimiento del automóvil para determinar el número de pulso a la izquier
da a la derecha es negativo El número de pulso izquierdo es positivo
  //Cuando la dirección de movimiento del automóvil para juzgar hacia la derecha, El número
de pulso derecha es negativo, El número de pulso izquierdo es positivo
  else if ((balancecar.pwm1 > 0) && (balancecar.pwm2 < 0))
  {
    rpluse = rpluse;
    lpluse = -lpluse;
  }

  balancecar.stopr += rpluse;
  balancecar.stopl += lpluse;

  //cada 5 ms actualiza los valores para el control
  balancecar.pulseright += rpluse;
  balancecar.pulseleft += lpluse;
  sumam = balancecar.pulseright + balancecar.pulseleft;
}

//***************PID 1 para el angulo*********************//
void angleout()
{
  balancecar.angleoutput = -kp * (angle + angle0) - kd * Gyro_x;//PD Angle loop control
}
//***************PID 1 para el angulo*********************//

////////////////////////////////////////////////////////////
//////////////////Interrupción cada 5 ms//////////////////////
////////////////////////////////////////////////////////////
void inter()
{
  sei();
  countpluse();                    //cuenta los pulsos de los encoders
  previousTime = currentTime;        //Se almacena el tiempo actual para calcular dt para integr
ar
  currentTime = millis();           //el giro y poder calcular el angulo de giro usando el giro

  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);     //lectura de los valores del mpu6050
  elapsedTime = (currentTime - previousTime) / 1000; //se divide por 1000 para obtener en se
gundo

  //Se calcula el angulo se llama al filtro de kalman
  Test_Angulo(ax, ay, az, gx, gy, gz, dt, Q_angle, Q_gyro, R_angle, C_0, K1);

  angleout();    //PD para el control del angulo (PID 1)

  //PID 2, ganancias y valores usados para el pid de velocidad
  speedcc++;
  if (speedcc >= 8) //luego de 40 ms entra al lazo de velocidad recta (PID 2)
  {
    Outputs = balancecar.speedpiout(kp_velocidad, ki_velocidad, kd_velocidad, front, back, se
tp0);
```

```
    speedcc = 0;
  }

  //PID 3, ganancias y valores usados para el pid de giro
  contador_giro++;
  if (contador_giro > 4)  //luego de 20ms entra al lazo de giro (PID 3)
  {
    salida_giro = balancecar.turnspin(turnl, turnr, spinl, spinr, kp_giro, kd_giro, Gyro_z);

    contador_giro = 0;
  }
  balancecar.posture++;
  balancecar.pwma(Outputs, salida_giro, angle, Gyro_z, turnl, turnr, spinl, spinr, front, back, a
ccelz, IN4, IN3, IN2, IN1, ENB, ENA);

  if ((-gz / 131 < -2) || (-gz / 131 > 2))  {
    delta_yaw = (gz / 131) * elapsedTime;
    yaw = yaw + (delta_yaw);
  }
  if (yaw > 360) yaw = 0;
  if (yaw < -360) yaw = 0;
  timer++;
}

int contador1 = 0;

void EnviarDatos()
{ //Posicion en el arreglo:
  Serial.print(lz);              //0
  Serial.print(",");
  Serial.print(rz);              //1
  Serial.print(",");
  Serial.print(balancecar.pwm2); //2
  Serial.print(",");
  Serial.print(balancecar.pwm1); //3
  Serial.print(",");
  Serial.print(angle);           //4
  Serial.print(",");
  Serial.print(yaw);             //5
  Serial.print(",");
  Serial.print(Gyro_z);          //6
  Serial.print(",");
  //PIDs
  Serial.print(kp);              //7
  Serial.print(",");
  Serial.print(ki);              //8
  Serial.print(",");
  Serial.print(kd);              //9
  Serial.print(",");
  Serial.print(kp_velocidad);    //10
  Serial.print(",");
  Serial.print(ki_velocidad);    //11
  Serial.print(",");
```

```
  Serial.print(kd_velocidad);    //12
  Serial.print(",");
  Serial.print(kp_giro);         //13
  Serial.print(",");
  Serial.print(ki_giro);         //14
  Serial.print(",");
  Serial.println(kd_giro);       //15
  delay(100);
}

void setup() {
  //inicializar driver l298n
  pinMode(IN4, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(ENA, OUTPUT);

  digitalWrite(IN4, 0);
  digitalWrite(IN3, 1);
  digitalWrite(IN2, 1);
  digitalWrite(IN1, 0);
  analogWrite(ENB, 0);
  analogWrite(ENA, 0);

  pinMode(PinA_left, INPUT);
  pinMode(PinA_right, INPUT);

  Wire.begin();
  Serial.begin(115200); //clockrate con el que se comunica con vb

  delay(1500);
  mpu.initialize();
  delay(2);
  balancecar.pwm1 = 0;
  balancecar.pwm2 = 0;

  FlexiTimer2::set(5, inter);    //5ms
  FlexiTimer2::start();

}

void loop()
{

  //detección del ciclo y la medición de pulso para detectar el cambio de la velocidad del auto
móvil para garantizar la precisión del automóvil.
  attachInterrupt(1, Code_right, CHANGE);   //interrupcion llanta derecha
  attachInterrupt(0, Code_left, CHANGE);    //interrucion llanta izquierda

  EnviarDatos();  //envia los valores via bt comunicacion serial
```

```
  RecibirDatos(); //lee string que recibe

}

//cada que ocurre una interrupcion en el encoder de la llanta izquierda suma 1 el contador izq
uierdo
void Code_left()
{
  contador_rueda_izquierda ++;
}

//cada que ocurre una interrupcion en el encoder de la llanta derecha suma 1 el contador der
echo
void Code_right()
{
  contador_rueda_derecha ++;
}

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

////////////////////////////Calcula el angulo/////////////////////////////////
void Test_Angulo(int16_t ax, int16_t ay, int16_t az, int16_t gx, int16_t gy, int16_t gz, float dt, f
loat Q_angle, float Q_gyro, float R_angle, float C_0, float K1)
{

  float Angle = atan2(ay , az) * 57.3; //ROLL
  Gyro_x = gx / 131;

  //llama al filtro de kalman
  Filtro_Kalman(Angle, Gyro_x, dt, Q_angle, Q_gyro, R_angle, C_0);

  //giro para calcular yaw usando giroscopio
  Gyro_z = -gz / 131;

}

void Filtro_Kalman(double angle_m, double gyro_m, float dt, float Q_angle, float Q_gyro, floa
t R_angle, float C_0)
{
  //*****************************************
  //*************  Prediccion  **************
  //*****************************************

  //angulo estimado
  angle += (gyro_m - q_bias) * dt;
  angle_err = angle_m - angle;

  //Matriz P estimada
  Pdot[0] = Q_angle - P[0][1] - P[1][0];
  Pdot[1] = - P[1][1];
  Pdot[2] = - P[1][1];
  Pdot[3] = Q_gyro;
  P[0][0] += Pdot[0] * dt;
```

```
  P[0][1] += Pdot[1] * dt;
  P[1][0] += Pdot[2] * dt;
  P[1][1] += Pdot[3] * dt;

  //*****************************************
  //**************  Correccion  **************
  //*****************************************

  //Innovacion
  PCt_0 = C_0 * P[0][0];
  PCt_1 = C_0 * P[1][0];
  E = R_angle + C_0 * PCt_0;

  //Ganancia de Kalman
  K_0 = PCt_0 / E;
  K_1 = PCt_1 / E;

  //variables auxiliares
  t_0 = PCt_0;
  t_1 = C_0 * P[0][1];

  //Matriz P corregida
  P[0][0] -= K_0 * t_0;
  P[0][1] -= K_0 * t_1;
  P[1][0] -= K_1 * t_0;
  P[1][1] -= K_1 * t_1;

  //valores corregidos
  angle += K_0 * angle_err; //angulo filtrado
  q_bias += K_1 * angle_err;
  angle_dot = gyro_m - q_bias;
}
```