EVALUACIÓN DE LOS MÉTODOS *UMBMARK Y TRIANGULAR PATH* PARA LA ESTIMACIÓN Y CORRECCIÓN DE ERRORES SISTEMÁTICOS EN ODOMETRÍA.

JHON EDINSON CALDERON GARCIA

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C.
2016

EVALUACIÓN DE LOS MÉTODOS *UMBMARK Y TRIANGULAR PATH* PARA LA ESTIMACIÓN Y CORRECCIÓN DE ERRORES SISTEMÁTICOS EN ODOMETRÍA.

JHON EDINSON CALDERON GARCIA

Proyecto de grado para optar el título de Ingeniero Electrónico

Director

Andrés Camilo Jiménez Álvarez
Ingeniero Electrónico

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRONICA
BOGOTÁ D.C.
2016

Nota de Aceptación:

_____
_____
_____
_____
_____
_____

_____
Firma del presidente del jurado

_____
Firma del jurado

_____
Firma del jurado

Bogotá D.C., 28 de octubre de 2016

DEDICATORIA

Henrytwo; inspirado en mi padre Henry… y mi hermana Luisa.

Contenido

FIGURAS

TABLAS

1 INTRODUCCIÓN

La localización es fundamental en los parámetros de diseño en un robot móvil autónomo, dada por la necesidad de conocer su ubicación con la mayor exactitud posible, para efectuar con éxito la misión asignada, no obstante la investigación de técnicas de localización, busca desarrollar técnicas que permitan fácilmente realizar correcciones necesarias para mejorar precisión la ubicación de un robot

En la robótica móvil, existen dos maneras de referenciar la posición de un robot mediante referencias relativas y absolutas[1]. De las referencias relativas, la odometría es una técnica económica para localizar un robot, ya que toma como referencia su posición inicial para después calcular el desplazamiento a partir del punto de inicio, este cálculo puede ser realizado mediante sensores encoder. Las referencias absolutas están basadas en medidas externas del robot obtenidas de balizas o puntos de referencias. En la odometría existen muchas fuentes de error, debido a las características inherentes del robot (Errores sistemáticos) y a las irregularidades de la superficie del entorno de trabajo del robot (errores no sistemáticos)[2]. Los errores no sistemáticos son los más difíciles de corregir ya que su naturaleza es netamente estocástica. Los errores sistemáticos provienen del diámetro desigual entre las ruedas, distancia efectiva entre ejes y resolución de los sensores encoder. Diversos autores tales como *Johan Boreinstein, Liqiang Feng, Xiaogang Ruan, Yalei Li Y Xiaoqing Zhu,* han sugerido diferentes métodos para reducir este tipo de errores para precisar la localización, teniendo en cuenta diferentes criterios para realizar correcciones. Para estimar y corregir errores sistemáticos de odometría en un robot móvil diferencial, se realizaron experimentos utilizando las técnicas de corrección *UMBmark* y *Triangular Path,* de las cuales se realizaron algunas modificaciones para así ejecutar las versiones Reducidas en el robot diferencial de pruebas "*Henrytwo".* Para facilitar los cálculos y reducir las reprogramaciones del robot se crea una interfaz (GUI).

De la ejecución de las técnicas *UMBmark* Reducida (TUR) y *Triangular Path Reducida (*TPR) se obtuvieron correcciones considerables, a pesar de la gran influencia de los errores no sistemáticos, los factores de corrección de la técnica UMBmark Reducida fueron más confiables para la navegación del robot pruebas "*Henrytwo".*

[1] J.Boreinstein, H. R. Everett, Where am I? Sensors and Methods for Mobile Robot Positioning. 1996. p10
[2] J.Borenstein and L.Feng. UmbMark: A Metod for Measurig, Comparing and Correct ing Dead-reckonin Errors in Mobile Robots. Universidad de Michigan: 1994. Pag. 4

2 PLANTEAMIENTO DELPROBLEMA

En la localización de robots móviles existe una técnica llamada odometría, que está basada en una técnica denominada Dead-Reckoning o navegación por estima, la cual inicialmente fue utilizada en la navegación de barcos tomando como referencia el punto inicial y el rumbo. Uno de los primeros experimentos aplicados en la robótica para la estimación y corrección de errores en odometría fue llamado "uni-directional square path" del año 1987. Johan Boreinstein en 1996 implementó y mejoró este test para dar paso al experimento *"Bi-Directional Square Path Test*" llamado *University Of Michigan Benchmark o UMBmark* como es conocido hoy en dia.

Basicamente la odometria consiste en el cálculo y estimación de la posición actual de un robot a través de la distancia recorrida y el ángulo de giro, tomando como referencia su punto de inicio. Este cálculo es posible gracias a los datos provenientes de diferentes tipos de sensores como, encoder, acelerómetros y brújulas. Los sensores encoder permiten conocer la posición de un robot móvil ya que obtiene los datos mediante cálculos matemáticos. Estos sensores son baratos en comparación con otros dispositivos, pero son muy susceptibles a errores provenientes del entorno de trabajo y de los errores generados por el propio robot móvil. Los errores provenientes de los sensores encoder, se acumulan proporcionalmente a la distancia recorrida por el robot. La corrección de errores de odometría se clasifican en dos tipos: Errores sistemáticos y no sistemáticos, los errores sistemáticos son generados por la resolución del encoder, la distancia real entre ruedas, diferencia entre los diámetros de la ruedas, desbalance en el peso que soporta cada llanta entre otros. Teniendo en cuenta estos parámetros, existen técnicas de corrección que procuran tomar la odometría como una técnica viable en la localización de robots móviles de bajo costo. En la actualidad existen diversas tecnicas para la estimación y correccion de errores en odometria, entre ellas las mas conocidas son el *UMBmark*, *Triangular Path* y *Circular Path*, estas dos ultimas buscan simplificar el Test UMBmark.

Para el robot diferencial de pruebas *"Henrytwo"* fue necesario diseñar un controlador PID para ejecutar trayectorias en linea recta, por ende se utilizo el test *UMBmark* y Triangular Path, ademas el test *UMBmark* y Triangular Path presentan como ventaja la viabilidad de ser aplicados a cualquier tipo de robot movil con llantas.

3 JUSTIFICACIÓN

Precisar la ubicación de un robot presenta dificultades por las sumatorias de errores en su posición, debido a las condiciones físicas del robot y a las características del terreno. Las fuentes de errores tales como la diferencia efectiva entre ejes, generan errores en el ángulo de giro del robot. La diferencia del diámetro de las llantas genera desviaciones a los costados, cuando el robot se desplaza en línea recta, las aceleraciones, desaceleraciones y giros bruscos a gran velocidad, provocan errores en la lectura de los encoder. Estos tipos de errores, están clasificados como errores sistemáticos, que procuran ser corregidos por técnicas como el *Test UMBmark y Triangular Path Test*. Estas técnicas son aplicadas experimentalmente a un robot móvil de tracción diferencial *"Henrytwo"*, para determinar el método que más precise la localización del robot.

En el diseño y construcción del robot de pruebas "Henrytwo" se realizó la implementación de un controlador PID digital para compensar los cambios de velocidad repentinos de los motores. Este controlador se diseñó solo para trayectorias en línea recta por ende se utilizan los *Test UMBmark y Triangular Path* en los experimentos con el robot de pruebas (Se descarta el Test *Circular Path* debido a los criterios de diseño del robot "Henrytwo"). La correcta calibración de un robot móvil permite reducir la incertidumbre en la localización, además ayuda al robot a cumplir con mayor precisión la misión asignada. Con la integración de otros sensores y técnicas de localización como la odometría visual, convierte a la odometría en una técnica eficiente y económica. Esto es el principal objetivo a cumplir en el desarrollo de robots autónomos móviles y concibe las bases de aplicación de técnicas como el SLAM (Simultaneous Localization And Mapping), el cual permite a un robot determinar su ubicación en un mapa construido por el propio robot. Las áreas de aplicación de la odometría son muy amplias, ya que es un criterio muy importante en el diseño y construcción de robots móviles autónomos de servicio, por ejemplo: los robots para búsqueda y rescate, exploración, desactivación de minas Etc.

4 **OBJETIVOS**

Objetivo General

Evaluar los métodos *UMBmark y Triangular Path* para la estimación y corrección de errores sistemáticos en odometría.

Objetivos específicos

- Diseñar y caracterizar un robot móvil diferencial para la estimación y corrección de errores de odometría.
- Ejecutar los experimentos *UMBmark y Triangular Path* para la estimación y corrección de errores en odometría.
- Comparar los errores obtenidos y determinar el método apropiado para el robot móvil diferencial construido.

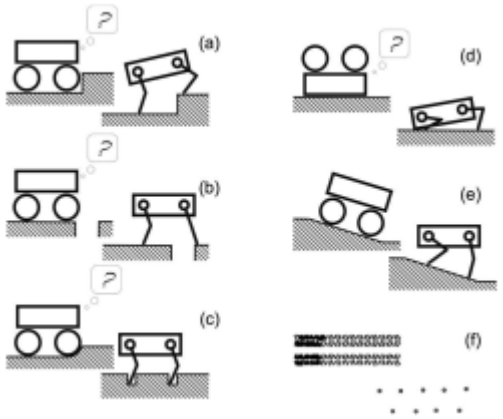
5 MARCO DE REFERENCIA

5.1 MARCO CONCEPTUAL

5.1.1 Robots móviles autónomos

Un robot móvil autónomo es una máquina capaz de realizar tareas sin la necesidad de contar con un operador, este debe tener sensores que permitan procesar y capturar información del entorno para controlar diferentes tipos de actuadores que pueden ser ruedas o articulaciones[3]. Dependiendo si el RMA necesita mayor velocidad y maniobrabilidad se hace uso de actuadores con ruedas, en caso contrario que el entorno presente dificultades este debe hacer uso de actuadores con articulaciones como se aprecia en la figura 1.

Figura 1. Comparación de vehículos con ruedas y con articulaciones para el desplazamiento.



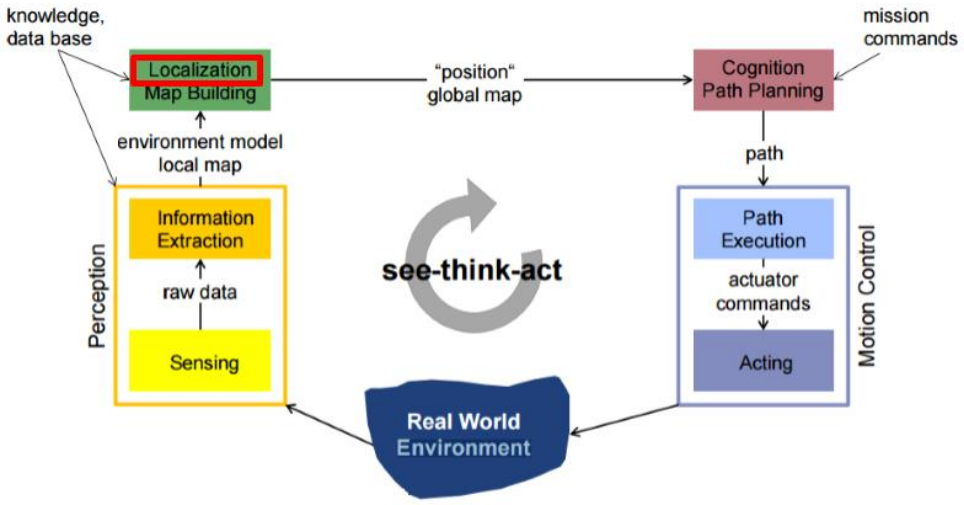
Fuente 1.Introduction to Autonomous Mobile Robots.

Es necesario que el RMA tenga un objetivo claro a cumplir para la planificación de tareas, siendo necesario que este cuente con un modelo del entorno, o este debe ser capaz de construirlo por si solo haciendo uso de sensores especializados como LIDAR o cámaras. No obstante, en todo momento debe ser capaz de procesar el entorno en el que se encuentra, generando de manera simultánea la localización respecto al mismo. El éxito del objetivo del RMA, este debe responder las siguientes preguntas[4] ¿Dónde estoy? ¿A dónde voy? y ¿Cómo llego? Esto es posible al implementar el ciclo *See-Think-Act,* en donde, se deben tener en cuenta las siguientes variables:

[3] Andrade Federico, Llofriu Martin. SLAM: Estado del arte. Montevideo.

[4] Mike Bosse,Marco Hutter, Martin Rufli, Davide Scaramuzza, (Margarita Chli, Paul Furgale), José "Autonomous Mobile Robots". {En línea}.{10 julio de 2015} disponible en internet :(https://edge.edx.org/courses/course-v1%3AETHZ%2BAMRx_Internal1_2015).

Figura 2. Ciclo see-think-act para el desarrollo de robots móviles autónomos.



Fuente 2. See-Think-Act cycle.https://edge.edx.org/courses/course-v1%3AETHZ%2BAMRx_Internal1_2015).

- **Mundo real**

Es el entorno en donde el robot realiza su misión. Las características del entorno determinan las propiedades cinemáticas del robot ya que es fundamental para definir parámetros en el tipo de ruedas, articulaciones, materiales. Etc

- **Percepción**

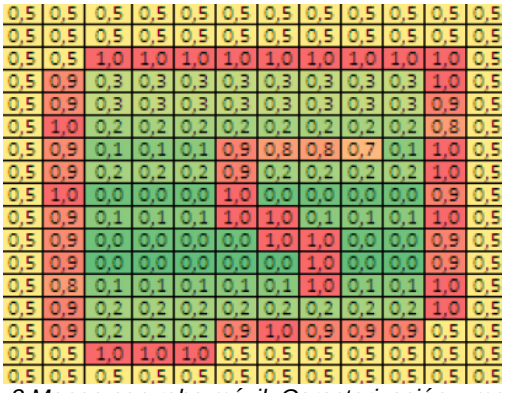
Es la captación de información realizada por el robot del mundo real. La información se percibe por dispositivos denominados sensores. Esta información es interpretada y procesada por microcontroladores o microprocesadores.

- **Localización y construcción de mapas**

Se entiende como localización de un robot móvil, la estimación de la posición en el mundo real y se soluciona tomando puntos de referencia externos por medio de sensores. Existen diversos métodos para la construcción de mapas donde la variación principalmente está dada en la forma de representar las características del entorno. Estos métodos se pueden combinar para lograr mayor precisión al momento de construir el mapa. Alguno de los más destacados son:

Mapas de rejilla de ocupación. Los mapas de rejilla de ocupación, consisten en la identificación de obstáculos mediante estimaciones probabilísticas descritas en una matriz. Esta estimación se realiza con probabilidades desde 0.0 a 1, siendo 0.0 la probabilidad más baja de que exista un obstáculo y 1 la probabilidad que estima que hay 100% de probabilidad de la existencia de un obstáculo. Estas estimaciones están a razón de la calidad y caracterización de los sensores[5].

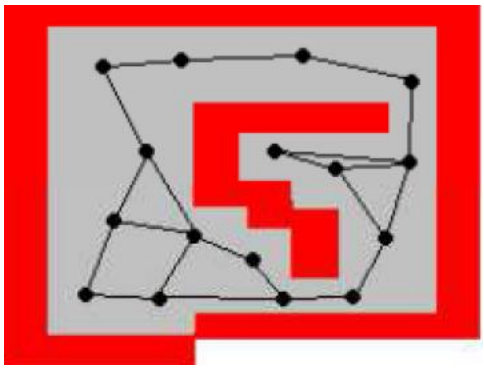
Figura 3.Mapa de rejilla de ocupación



Fuente 3.Mapeo con robo móvil: Caracterización y modelado

Mapas topológicos. Los mapas topológicos consisten en la determinación de puntos clave en un mapa y la trazabilidad entre ellos será determinado como transitable[6].

Figura 4.Mapas topológicos.



Fuente 4. Mapeo con robo móvil: Caracterización y modelado

[5]Martin Pérez, Jorge, Mapeo con robo móvil: Caracterizacion y modelado, Bellaterra, 2012, 46p. Tesis (Especializacion en sistemas electronicos). Universitat Autònoma de Barcelona (UAB).
[6] Ibid.,p.47

Mapas difusos. Los mapas difusos se construyen con estimaciones probabilísticas pero con la difusión de datos. Al igual que los mapas de rejilla, este algoritmo tiene en cuenta la probabilidad de funcionamiento de los sensores. Esta representación también es conocida como representaciones basadas en representaciones difusas.

5.1.2 Robots móviles de tracción diferencial.

Los robots móviles de tracción diferencial están compuestos por dos ruedas con un motor fijado a cada una de. Las ruedas están fijas a una plataforma que a su vez cuenta con una rueda castor, (rueda loca) para mantener el balance, esta rueda no es tenida en cuenta en los parámetros de diseño, ya que no ejerce ningún tipo de tracción. Este tipo de robot es bastante sencillo de construir, la mayor desventaja se presenta al momento de mantener trayectorias en línea recta ya que es muy susceptible a las variaciones del terreno.

Modelo cinemático de robots con tracción diferencial.

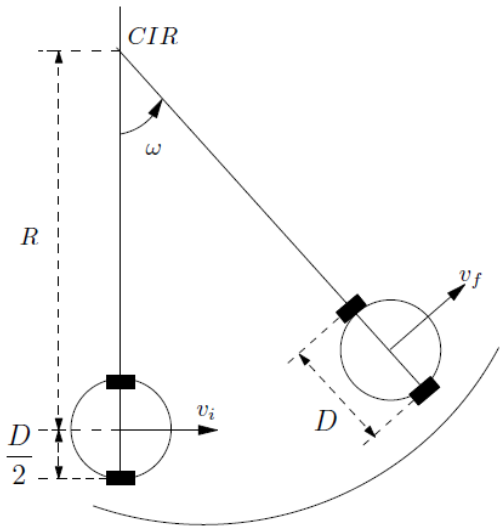
La trayectoria “recta” de un robot diferencial se puede representar como una curva la cual se puede descomponer en velocidad angular ω y velocidad de traslación v.

Velocidad angular. La velocidad angular es el cambio de ángulo $d\delta$ durante el intervalo de tiempo dt.

$$\omega = \frac{d\delta}{dt} \tag{1}$$

Donde $\delta(t)$ es la posición angular en radianes.

Figura 5.Cinemática robots de tracción diferencial.



Fuente 5.Geometría de la cinemática en los RMLD [Dudek00].Calibración de robots de locomoción diferencial por el método de mínimos cuadrados empleando la técnica de suavizado

Velocidad de traslación

La velocidad de traslación hace referencia al cambio de posición del robot respecto centro instantáneo de rotación CIR.

$$v = \omega R \tag{2}$$

Donde *R* es el radio de la circunferencia o radio de curvatura. EL CIR o Centro Instantáneo de rotación, es el punto ubicado en el centro de la circunferencia que describe la trayectoria[7]. Para representar la velocidad del robot se define el vector:

$$[v \quad \omega]^T \tag{3}$$

Las ruedas del robot describen el mismo movimiento circular, pero el radio de curvatura es diferente en cada rueda. De la rueda derecha tenemos:

[7] LOPEZ, Shadai, calibración de robots de locomoción diferencial por el método de mínimos cuadrados empleando la técnica de suavizado y cartografía incremental, Pátzcuaro, 2011, 13-17p. Tesis (Maestro En Ciencias En Ingeniería Eléctrica). Universidad Michoacana de San Nicolás de Hidalgo.

$$R_d = R + \frac{b}{2} \tag{4}$$

Igualmente de la rueda izquierda:

$$R_i = R - \frac{b}{2} \tag{5}$$

Donde b es la distancia entre la ruedas del robot diferencial. Para obtener la velocidad de traslación de cada llanta, reemplazamos las ecuaciones (4) y (5) en la ecuación (2), por lo tanto la velocidad de traslación es:

$$v_d = \omega(R + \frac{b}{2}) \tag{6}$$

Igualmente la de la llanta izquierda es:

$$v_i = \omega(R - \frac{b}{2}) \tag{7}$$

De la diferencia entre las velocidades v_d y v_i se obtiene:

$$v_d - v_i = \omega b \tag{8}$$

Despejando ω se obtiene:

$$\omega = \frac{v_d - v_i}{b} \tag{9}$$

Igualando la velocidad de traslación de cada rueda obtenemos:

$$\frac{v_d}{R + \frac{b}{2}} = \frac{v_i}{R - \frac{b}{2}} \tag{10}$$

De la anterior ecuación podemos definir el radio de curvatura R de la siguiente forma:

$$R = \frac{b(v_d + v_i)}{2(v_d - v_i)} \tag{11}$$

No obstante se define la velocidad de traslación como:

$$v = \frac{(v_i + v_d)}{2} \tag{12}$$

Podemos definir el vector $[v \quad \omega]^T$ como:

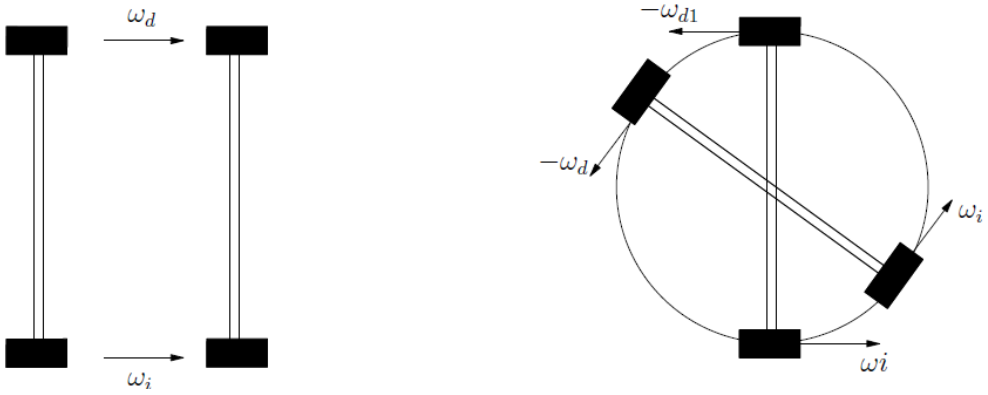
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{b} & \frac{1}{b} \end{bmatrix} \begin{bmatrix} v_d \\ v_i \end{bmatrix} \tag{13}$$

Multiplicando la velocidad angular por la magnitud de su radio:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r_d}{2} & \frac{r_i}{2} \\ \frac{r_d}{b} & \frac{-r_i}{b} \end{bmatrix} \begin{bmatrix} \omega_d \\ \omega_i \end{bmatrix} \tag{14}$$

Siendo ω_d y ω_i la velocidad de cada llanta. De las cuales se obtienen las siguientes acciones de control:

Figura 6. a) Desplazamiento lineal. b) Rotación.



Fuente 6. Acciones de control de los robots de tracción diferencial.

5.1.3 Control automático

El control automático es la intervención en un sistema para mantenerlo en un estado deseado. El control automático se ha implementado significativamente desde el siglo XVIII, debido a la necesidad del ser humano por realizar eficientemente sus actividades cotidianas[8].

Un sistema de control se describe mediante[9]:

1. Objetivos de control (Entradas)
2. Componentes del sistema de control
3. Variables controladas (Salidas)

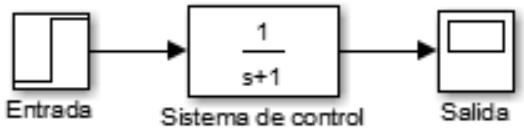
Figura 7. Sistemas de control

Fuente 7: Propia

[8] Ogata, Katsuhiko. Ingenieria de control moderna, 5 ed. Madrid: Prentice Hall, 2010.p2
[9] Kuo, Benjamin. Sistemas de control automatico,7 ed. Madrid: Prentice Hall, 1996. p2

Sistemas de control en lazo abierto

Figura 8. Sistema de control lazo abierto

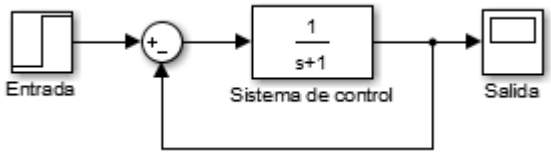


Fuente 8. Propia

En un sistema de lazo abierto la salida no afecta la acción de control ya que no existe comparación entre la entrada y la salida. Los sistemas en lazo cerrado trabajan con una base de tiempo y la precisión de estos depende directamente de una correcta calibración[10].

Sistemas de control en lazo cerrado

Figura 9. Sistema de control lazo cerrado



Fuente 9. Propia

En los sistemas de lazo cerrado existen una relación comparativa entre la salida y la entrada, obteniendo así una señal que nos permitirá efectuar control sobre el sistema. Un sistema en lazo cerrado es relativamente insensible a perturbaciones externas y a las fluctuaciones en los parámetros internos del sistema.

Sistemas de primer orden

Los sistemas de primer de primer orden son aquellos que cuya respuesta corresponde a:

[10] Ogata, op. Cit, p.7

$$\frac{K}{Ts+1} \tag{15}$$

Siendo estos los más sencillos que se pueden hallar dado que solo contiene un polo.

Controladores PID

En el análisis para la aplicación de controladores PID se debe obtener el modelo matemático de la planta. En algunos casos no se facilita obtener estos modelos analíticamente por lo tanto se recurre a métodos experimentales para la obtención del modelo, que nos permita hacer la aplicación de un controlador PID[11].

Un controlador PID matemáticamente presenta la siguiente forma:

$$K_p(1+\frac{1}{T_i s}+T_d s) \tag{16}$$

Donde

K_P: Ganancia proporcional

T_i: Tiempo integral

T_d: Tiempo derivativo

Reglas de Ziegler-Nichols

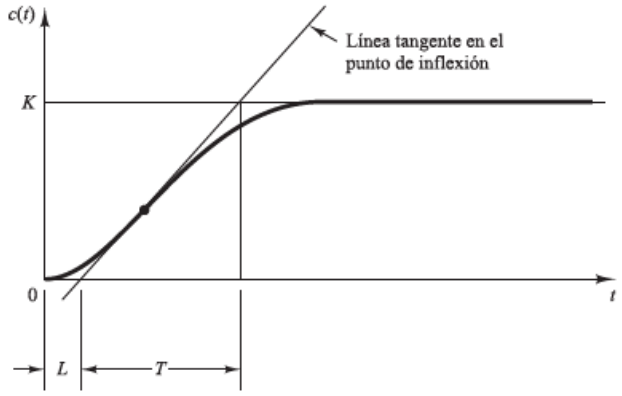
Ziegler y Nichols proponen reglas para hallar los valores de K_P, T_i y T_d, basados en los resultados de la caracterización de la planta, existen principalmente dos métodos de sintonización Ziegler-Nichols para controladores PID. En el primer método se analiza la respuesta del sistema a una señal de entrada escalón y en el segundo método se analiza las oscilaciones permanentes producto de una ganancia crítica K_c.

[11] Ibíd.,p.669

Primer método

En este método se aplica solo si la respuesta de salida obtenida tiene forma de “S” ante una señal escalón a la entrada. Este tipo de respuesta contiene dos parámetros; el tiempo de retardo L y la constante de tiempo T. Esta curva se obtiene de forma experimental o simulada.

Figura 10. Respuesta en forma "S"



Fuente 10.Ogata, Benjamín. Ingeniería de control moderna

La función de transferencia $C(s)/U(s)$ es aproximada a un sistema de primer orden con un retardo con la forma:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{17}$$

A partir de estos parámetros son sintonizan los valores de K_P, T_i y T_d, mediante la siguiente regla:

Tabla 1. Reglas de sintonización PID

Tipo de controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	þ
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Fuente 11. Ogata, Benjamín. Ingeniería de control moderna

Reemplazando los valores en la ecuación (16) obtenemos el controlador PID sintonizado.

$$G(s) = K_p(1 + \frac{1}{T_i s} + T_d s) \tag{18}$$

$$1.2\ \frac{T}{L}(1 + \frac{1}{2Ls} + 0.5Ls) \tag{19}$$

$$0.6T\frac{(s + \frac{1}{L})^2}{s} \tag{20}$$

5.2 MARCO TEORICO

5.2.1 Incertidumbre en robots

Las principales fuentes de incertidumbre en los robots móviles son[12]:

- **Entorno.** El entorno es una gran fuente de incertidumbre debido a su naturaleza desconocida y es tratada de manera estocástica.

- **Sensores.** Los sensores presentan limitaciones al momento de realizar interpretaciones del entorno, ya que están sujetos a dos factores primarios; leyes físicas y la exposición del sensor al ruido, un ejemplo son las limitaciones dadas por la resolución de un encoder y las lecturas de este en una superficie rugosa.

- **Robots.** Este tipo de incertidumbre hace referencia a las características propias de los componentes del robot, como motores y actuadores que están sujetos a desgastes.

- **Controles**. Al momento de ejecutar órdenes de movimiento a los motores del robot, debe actualizar su posición (Odometría). Al momento de ejecutar una orden en una superficie de baja adherencia, el movimiento de las llantas serán completadas pero la información de odometría será errada.

- **Modelos.** Los modelos son representaciones del mundo real, y los errores en estos pueden ser una gran fuente de incertidumbre.

- **Capacidad de cómputo.** Los robots móviles trabajan con información en tiempo real y La limitación en la capacidad de cómputo es una limitación debido a la cantidad de procesos que sea capaz de llevar a cabo el robot.

5.2.2 Métodos de posicionamiento

En la aplicación de robots móviles, existen dos tipos de posicionamiento llamados posicionamiento absoluto y relativo[13].

[12] THRUN, Sebastian. Probabilistic robotics.Edicion Borrador.1999-2000.P2
[13] J.Boreinstein, H. R. Everett, Where am I? Sensors and Methods for Mobile Robot Positioning. 1996. p10

Posicionamiento Absoluto

El posicionamiento absoluto consiste en la ubicación de un objeto en un espacio determinado, usando puntos de referencia tales como balizas, marcas, emparejamiento de mapas, GPS Etc. Estos sistemas generalmente resultan más costosos dado a que requieren instalación, mantenimiento y mayor capacidad de cómputo, igualmente el uso en interiores es limitado ya que la precisión no es lo suficientemente exacta para la navegación en robots móviles.

Posicionamiento Relativo.

El posicionamiento relativo está basado en la técnica denominada Dead-Reckoning la cual toma como referencia la posición inicial del vehículo y mediante sensores de desplazamiento realiza mediciones de la distancia recorrida y los cambios de orientación. Dead-Reckoning presenta como desventaja los errores acumulativos a razón de la distancia recorrida, pero a su vez es una técnica fácil y económica de implementar.

5.2.3 Dead-Reckoning

Dead-Reckoning es un método de localización inicialmente aplicado en la navegación de barcos tomando referencia el desplazamiento y rumbo. Se han realizado implementaciones de este método a la robótica, mediante una técnica denominada odometría.

5.2.4 Odometría

La odometría es una técnica de bajo costo, que sirve para determinar la posición (Localización) de un robot casi en cualquier entorno. La localización del robot está dada por el vector $x = [x \quad y \quad \theta]$ donde x e y son coordenadas y θ la orientación. Este vector es obtenido por medio de sensores de desplazamiento denominados encoder. Esta técnica presenta como desventaja los errores acumula

Errores sistemáticos

Los errores sistemáticos son originados por las características propias del robot producto de la diferencia en el diámetro de las ruedas y la incertidumbre en la distancia efectiva entre ruedas, siendo estas las mayores fuentes de errores. Existen otros errores tales como la resolución de los sensores encoder y los

errores de ensamblaje, se puede solucionar modificando los parámetros de diseño.

Errores no sistemáticos

Los errores no sistemáticos son originados por las características del entorno, debido a las irregularidades de la superficie. Condiciones de baja adherencia, provocan derrapes y deslizamientos que aumentan los errores en la localización. Los errores no sistemáticos son de naturaleza estocástica.[14]

5.2.5 Estimación del desplazamiento mediante odometría.

Para realizar una estimación del desplazamiento realizado por un robot diferencial, se necesita hallar el factor C_m se debe tener la siguiente ecuación:

$$C_m = \frac{\pi D_n}{n C_e} \tag{21}$$

Donde

C_m=Factor de conversión que traslada los pulsos del encoder en desplazamiento lineal.

D_n= Diámetro nominal de la rueda.

C_e=Resolución del encoder.

n = Relación de reducción.

Para calcular la distancia recorrida por cada llanta (Izquierda y derecha):

$$\Delta U_{L/R,t} = C_m N_{L/R,t} \tag{22}$$

Donde:

$N_{L/R}$=Incrementos de pulsos del encoder izquierdo-derecho

i = ie-sima muestra

[14] TANVEER Abbas, Measurement and Correction of Systematic Odometry Errors Caused by Kinematics Imperfections in Mobile Robots. En IEEE (Oct. 2006);Pag. 2.

Desplazamiento lineal con respecto al punto central C del robot diferencial, el cual lo representamos como ΔU_i

$$\Delta U_i = \frac{1}{2}(\Delta U_{Ri} + \Delta U_{Li}) \quad (23)$$

Hallando cambio de orientación:

$$\Delta\theta_i = \frac{1}{b}(\Delta U_{R,i-}\Delta U_{L,i}) \quad (24)$$

Donde igualmente b es la distancia nominal entre las dos ruedas.Para calcular la actual orientación relativa debemos:

$$\theta_i = \theta_{i-1} + \Delta\theta_i \quad (25)$$

Y la posición con respecto al centro de giro C es:

$$X_{i\,=} X_{i-1} + \Delta U_i \cos\theta_i \quad (26)$$

$$Y_{i\,=} Y_{i-1} + \Delta U_i \sin\theta_i \quad (27)$$

La odometría asume que la información suministrada por los encoder corresponde al desplazamiento lineal del robot, pero esto es una idealización, ya que los errores sistemáticos y no sistemáticos aportan lecturas erróneas que impiden una correcta localización del robot.

5.2.6 Técnicas para determinar errores sistemáticos en odometría.

Diferentes autores han realizado técnicas de reducción de errores sistemáticos, tratando los más significativos dados por la diferencia de diámetros de las ruedas y la incertidumbre en la distancia efectiva entre ellas. La mayoría de ellos recurren a la geometría para la medición de los errores. Los test a tratar en este proyecto son:

- Test bi-direccional (*UMBmark*)
- Triangular Path Test

Test bidireccional (UMBmark)

Este test fue desarrollado por la Universidad de Michigan el cual consiste en el seguimiento de una trayectoria cuadrada programada en un robot móvil. Para el desarrollo de este test se debe tener en cuenta que en superficies ásperas los errores no sistemáticos pueden ser dominantes, por lo tanto el desarrollo se debe hacer en una superficie totalmente plana. El método *UMBmark* principalmente trata los errores generados por la diferencia de diámetro entre las llantas y la incertidumbre de la distancia efectiva de los puntos de contacto entre ruedas.

Para la simplificación de los cálculos, *UMBmark* trata estos errores de manera separada, asumiendo que las diferencias de diámetro de las ruedas provocan las trayectorias ligeramente curvas en desplazamientos rectos y la incertidumbre de la distancia efectiva de los puntos de contacto entre ruedas, generan errores en los giros, provocando que este sea mayor o menor de lo estimado. Estos dos tipos de errores son los que significativamente más aportan a los errores sistemáticos. Para evitar derrapes y deslizamientos por cambios de aceleración, las velocidades deben ser inferiores a $1\ m/s$.

Errores por diferencia de diámetro entre llantas

Los robots diferenciales para su desplazamiento utilizan llantas de caucho, las cuales evitan derrapes y deslizamientos por cambios de aceleración[15]. Debido a la compresión del caucho, las llantas se deforman provocando un error por diferencia de diámetros. Este error se intensifica si la distribución de carga del robot no es la correcta, generando errores que principalmente se expresan en los desplazamientos en línea recta. Los errores por diferencia de diámetro entre llantas están dados por la relación:

$$E_d = \frac{d_R}{d_L} \tag{28}$$

Donde d_R y d_L son los diámetros de las llantas derecha e izquierda respectivamente donde E_d nominalmente es igual a 1.

[15] J.Borenstein and L.Feng. UmbMark: A Metod for Measurig, Comparing and Correct ing Dead-reckonin Errors in Mobile Robots. Universidad de Michigan: 1994. Pag. 5

Errores por incertidumbre de la distancia efectiva de los puntos de contacto entre ruedas

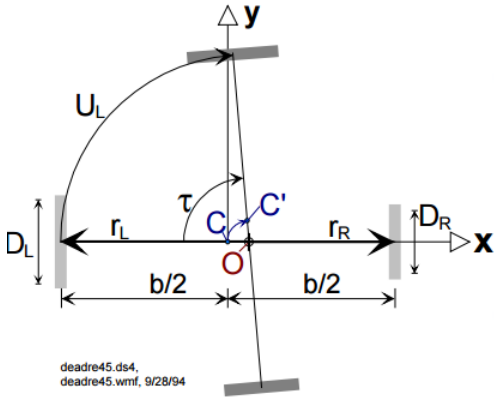
Este tipo de error es consecuencia de la incertidumbre del punto real de contacto entre las ruedas y el suelo, esto se debe principalmente al caucho de las llantas, ya que no es posible saber con exactitud el punto real de contacto. En robots comerciales este error por lo general es del 1%. El factor E_b esta dado por:

$$E_b = \frac{b_{real}}{b_{nominal}} \tag{29}$$

b_{real}=Distancia actual de las base de las ruedas.

$b_{nominal}$=Distancia nominal de la base de las ruedas.

Figura 11. Cambio del centro instantáneo de rotación



Fuente 12. Borenstein, Johann.UMBmark- A Method for Measuring, Comparing, and Correcting Dead-Reckoning Errors in Mobile Robots.

En la figura se aprecia el giro realizado por un robot diferencial. Para realizar el giro ambas ruedas deben girar a la misma velocidad y en sentido contrario una respecto a la otra. Se puede asumir que la velocidad angular es igual, pero debido a los errores de diámetro de las llantas, ocasiona que el centro geométrico del vehículo no coincida con el centro instantáneo de rotación (CIR) al completar el

giro. Esto ocasiona un desplazamiento lateral que puede ser ignorado ya que se anula en los giros restantes[16].

$$\frac{r_R}{r_L} = \frac{D_R}{D_L} \tag{30}$$

Donde $r_{R/L}$ es la distancia desde el CIR a la llanta izquierda y derecha. Despejando r_R, obtenemos:

$$r_R = \frac{D_R}{D_L} r_L \tag{31}$$

Bajo condiciones normales, el CIR está siempre sobre el eje, por lo tanto:

$$r_R + r_L = b \tag{32}$$

Sustituyendo (31) en (32) obtenemos:

$$r_L = \frac{D_L}{D_R + D_L} b \tag{33}$$

Suponiendo que el desplazamiento es curvilíneo como consecuencia de E_d obtenemos:

$$U_{L,n} = \pi D_{L,n} N_L \tag{34}$$

$U_{L,n}$= Desplazamiento curvo

$D_{L,n}$=Diámetro nominal izquierdo

N_L= Incrementos del encoder izquierdo

[16] Borenstein, Johann. Measurement and Correction of Systematic Odometry Errors in Mobile Robots. En: IEEE. Vol.;12. No (Dic,1996); p. 3.

Bajo condiciones nominales la llanta izquierda podría girar alrededor de *C* con un ángulo τ_n:

$$\tau_n = \frac{U_{L,n}}{b/2} = \frac{2\pi D_{L,n} N_L}{b} \tag{35}$$

Suponiendo que la llanta derecha es más pequeña que la izquierda. La rotación es ahora sobre el punto *C'* y el ángulo corresponde a su desplazamiento curvilíneo es:

$$\tau = \frac{U_L}{r_L} = \frac{\pi D_{L,n} N_L}{r_L} \tag{36}$$

Despejando N_L de (35) y reemplazando en (36):

$$\tau = \frac{\tau_n b D_L}{2 r_L D_{L,n}} \tag{37}$$

Sustituyendo (31) en (32), obtenemos:

$$\tau = \tau_n \frac{(D_R + D_L)}{2 D_{L,n}} \tag{38}$$

Para interpretar este resultado más fácilmente, definimos el promedio del diámetro actual de la rueda:

$$D_{promedio} = \frac{(D_R + D_L)}{2} \tag{39}$$

Y reescribimos como:

$$\frac{D_{promedio}}{\tau} = \frac{D_{L,n}}{\tau_n} \tag{40}$$

La ecuación (33) puede ser expresada como el promedio del diámetro actual de la rueda relaciona al actual ángulo de giro con el diámetro nominal por el ángulo nominal de giro[17].

Procedimiento para la ejecución del experimento

El experimento UMBmark se debe ejecutar:

1. Medir la posición y orientación (opcional) inicial del punto de partida del robot.
2. Realizar el recorrido del cuadrado en sentido horario.
3. Hacer mediciones cuando el robot “regrese” al punto de inicio.
4. Comparar la posición de llegada con la de salida.
5. Repetir el ejercicio cuatro veces más.
6. Repetir el todos los pasos en sentido anti-horario.
7. Representar los resultados gráficamente.

Centro de gravedad

Para determinar los centros de gravedad del experimento debemos

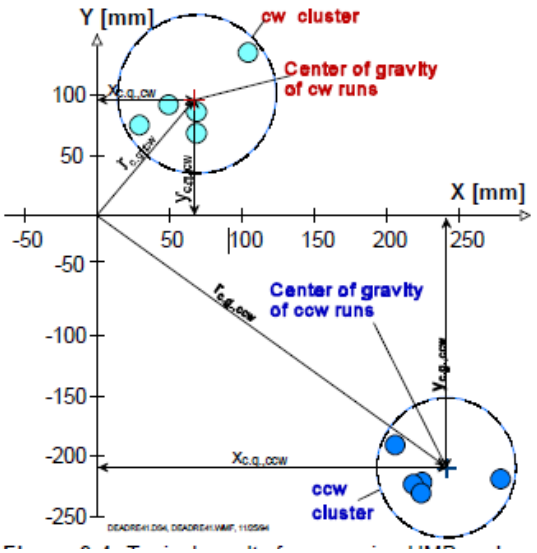
$$x_{c.g}, {}^{CW}/_{CCW} = \frac{1}{n}\sum_{i=1}^{n} \epsilon x_i, {}^{CW}/_{CCW} \tag{41}$$

$$y_{c.g}, {}^{CW}/_{CCW} = \frac{1}{n}\sum_{i=1}^{n} \epsilon y_i, {}^{CW}/_{CCW} \tag{42}$$

$$n = 5$$

[17] Ibíd., p.4

Figura 12. Típicos resultados del Test UMBmark CW y CCW



Fuente 13. Borenstein, Johann.UMBmark- A Method for Measuring, Comparing, and Correcting Dead-Reckoning Errors in Mobile Robots

Las distancias de los dos centros de gravedad desde el origen, están dados $r_{c.g.}, CW$ y $r_{c.g.}, CCW$ y se hallan mediante:

$$r_{c.g.,CW} = \sqrt{\left(x_{c.g.,CW}\right)^2 + \left(y_{c.g.,CW}\right)^2} \tag{43}$$

$$r_{c.g.,CCW} = \sqrt{\left(x_{c.g.,CCW}\right)^2 + \left(y_{c.g.,CCW}\right)^2} \tag{44}$$

De lo cual hallamos el máximos de las dos expresiones halladas.

$$E_{max,syst} = \max(r_{c.g.,CW}; r_{c.g.,CCW}) \tag{45}$$

De esta expresión (38) se debe tener en cuenta lo siguiente:

- El motivo por el que no se usa el promedio de $r_{c.g.,CW}$ y $r_{c.g.,CCW}$, se debe a razones prácticas, ya que se debe tratar el error más grande posible hallado por odometría[18].

- El error de orientación $\epsilon\theta$ no es considerado en la expresión $E_{max,syst}$, la razón de esto es porque $\epsilon\theta$ está implícito en todo el sistema por los errores hallados en la posición final.

Corrección De Errores De Odometría

El método UMBmark clasifica los errores en Tipo A y Tipo B en función de la orientación. Los errores Tipo A se definen como el error de orientación que se reduce (o incrementa) durante el experimento en sentido horario (CW) y anti-horario (CCW). Los errores Tipo B son definidos como el error de orientación que se reduce(o incrementa) la orientación en una dirección, pero incrementa (o reduce) la rotación cuando va en la otra dirección. Los errores Tipo A y Tipo B ocurren de manera simultánea, pero el problema consiste en distinguirlos uno del otro en el momento de realizar la prueba. Para simplificar asumimos[19]:

1. E_d y E_b Son los errores sistemáticos dominantes en la odometría.
2. Los valores de los errores de base E_b causa errores solo durante los giros.
3. La diferencia de los diámetros E_d causa errores durante los desplazamientos en línea recta.
4. E_b causa solo los errores Tipo A pero no los Tipo B.
5. E_d causa solo los errores Tipo B pero no los Tipo A.

Como consecuencia tenemos:

1. Asumiendo #1, eliminando E_b elimina los errores Tipo A casi completamente
2. asumiendo #1, eliminando E_d elimina los errores Tipo B casi completamente

[18] Borenstein, Johann, op. cit, p.6
[19] Ibíd., p.7

Errores Tipo A en sentido anti-horario

$$x_1 = x_0 + L \tag{46}$$
$$y_1 = y_0 \tag{47}$$
$$x_2 = x_1 + L\sin\alpha \approx L + L\alpha \tag{48}$$
$$y_2 = y_1 + L\cos\alpha \approx L \tag{49}$$
$$x_3 = x_2 - L\cos 2\alpha \approx L\alpha \tag{50}$$
$$y_3 = y_2 + L\sin 2\alpha \approx L + 2L\alpha \tag{51}$$
$$x_4 = x_3 - L\sin 3\alpha \approx -2L\alpha \tag{52}$$
$$y_4 = y_3 - L\cos 3\alpha \approx 2L\alpha \tag{53}$$

Errores Tipo A en sentido horario

$$x_1 = x_0 + L \tag{54}$$
$$y_1 = y_0 \tag{55}$$
$$x_2 = x_1 + L\sin\alpha \approx L + L\alpha \tag{56}$$
$$y_2 = y_1 - L\cos\alpha \approx -L \tag{57}$$

$$x_3 = x_2 - L\cos 2\alpha \approx L\alpha \tag{58}$$
$$y_3 = y_2 - L\sin 2\alpha \approx -L - 2L\alpha \tag{59}$$

$$x_4 = x_3 - L\sin 3\alpha \approx -2L\alpha \tag{60}$$
$$y_4 = y_3 + L\cos 3\alpha \approx -2L\alpha \tag{61}$$

Errores tipo B en sentido anti horario

$$x_1 = x_0 + L\cos(\beta/2) \approx L \tag{62}$$
$$y_1 = y_0 + L\sin(\beta/2) \approx L\beta/2 \tag{63}$$

$$x_2 = x_1 - L\sin(3\beta/2) \approx L - 3L\beta/2 \quad (64)$$

$$y_2 = y_1 + L\cos(3\beta/2) \approx L\beta/2 + L \quad (65)$$

$$x_3 = x_2 - L\cos(5\beta/2) \approx -3L\beta/2 \quad (66)$$

$$y_3 = y_2 - L\sin(5\beta/2) \approx -2L\beta + L \quad (67)$$

$$x_4 = x_3 + L\sin(7\beta/2) \approx 2L\beta \quad (68)$$

$$y_4 = y_3 - L\cos(7\beta/2) \approx -2L\beta \quad (69)$$

Errores Tipo B en sentido horario

$$x_1 = x_0 + L\cos(\beta/2) \approx L \quad (70)$$

$$y_1 = y_0 + L\sin(\beta/2) \approx L\beta/2 \quad (71)$$

$$x_2 = x_1 + L\sin(3\beta/2) \approx L + 3L\beta/2 \quad (72)$$

$$y_2 = y_1 - L\cos(3\beta/2) \approx L\beta/2 - L \quad (73)$$

$$x_3 = x_2 - L\cos(5\beta/2) \approx 3L\beta/2 \quad (74)$$

$$y_3 = y_2 - L\sin(5\beta/2) \approx -2(L\beta + L) \quad (75)$$

$$x_4 = x_3 - L\sin(7\beta/2) \approx -2L\beta \quad (76)$$

$$y_4 = y_3 + L\cos(7\beta/2) \approx -2L\beta \quad (77)$$

Reemplazando los errores Tipo A y errores Tipo B para el experimento en sentido horario en la dirección x, tenemos:

$$x_{CW}: -2L\alpha - 2L\beta = -2L(\alpha + \beta) = x_{c.g.,CW} \quad (78)$$

$$x_{CCW}: -2L\alpha + 2L\beta = -2L(\alpha - \beta) = x_{c.g.,CCW} \quad (79)$$

Sustrayendo

$$-4L = x_{c.g.CW} - x_{c.g.,CCW} \tag{80}$$

O

$$\beta = \frac{x_{c.g.CW} - x_{c.g.,CCW}}{-4L} \frac{(180°)}{\pi} \tag{81}$$

Para β en grados. Comparando en términos de la dirección y obtenemos un resultado similar

$$\beta = \frac{x_{c.g.CW} + x_{c.g.,CCW}}{-4L} \frac{(180°)}{\pi} \tag{82}$$

Usando relaciones geométricas, el radio de curvatura R puede ser encontrado del triángulo ABM

$$R = \frac{L/2}{\sin(\beta/2)} \tag{83}$$

Una vez el radio R es calculado, es fácil determinar la relación entre las dos ruedas dado que es la causa de que el robot realice trayectorias curvas

$$RE_d = \frac{D_R}{D_L} = \frac{R + b/2}{R - b/2} \tag{84}$$

La relación entre las dos ruedas puede ser usada para corregir los errores Tipo B. De igual manera α puede ser encontrada:

$$-4L\alpha = x_{c.g.CW} + x_{c.g.,CCW} \tag{85}$$

O

$$\alpha = \frac{x_{c.g.CW} + x_{c.g.,CCW}}{-4L} \frac{(180°)}{\pi} \tag{86}$$

Ahora podemos calcular el error de base E_b. La base b es inversamente proporcional a la cantidad actual de rotación. Usamos la ecuación:

$$\frac{b_{actual}}{90°} = \frac{b_{nominal}}{90° - \alpha} \tag{87}$$

Despejando

$$b_{actual} = \frac{90°}{90° - \alpha} b_{nominal} \tag{88}$$

Por odometría

$$E_b = \frac{90°}{90° - \alpha} \tag{89}$$

5.2.7 Corrección de errores sistemáticos en odometría

Una vez conocemos los valores de E_d y E_b, es fácil corregir el error en software. Para los errores de base se deben tomar de la ecuación (88) y realizar las correcciones correspondientes. La corrección de los errores de diámetro desigual de la ruedas, se hacen tomando los valores de E_d. Sin embargo cuando aplicamos el factor de compensación, debemos estar seguros de no cambiar el promedio del diámetro.

$$D_a = \frac{(D_{D+} D_I)}{2} \tag{90}$$

Resolviendo las ecuaciones (84) y (90) las cuales son dos ecuaciones lineales con dos incógnitas.

$$D_L = \frac{2}{E_d + 1} D_a \tag{91}$$

Y

$$D_R = \frac{2E_d}{(E_d + 1)} D_a \tag{92}$$

Ahora podemos definir dos factores de corrección:

$$c_L = \frac{2}{E_d + 1} \tag{93}$$

$$c_R = \frac{2}{(1/E_d) + 1} \tag{94}$$

Dichos resultados se deben implementar en el algoritmo de odometría, redefiniéndolo como:

$$\Delta \boldsymbol{U_{L/R,i}} = \boldsymbol{c_{L/R} C_m N_{L/R,i}} \tag{95}$$

Triangular Path Test

El método *UMBmark* trata los errores E_b Y E_d de manera independiente clasificándolos como errores Tipo A y Tipo B respectivamente. Esto no es del todo cierto ya que esta conjetura puede afectar la precisión en el experimento. El test del triángulo equilátero primero propone efectuar un test en línea recta para calibrar los diámetros de las llantas; segundo efectuar el test en forma triángulo equilátero[20].

Test en línea recta

Como primera medida se debe determinar la posición inicial del robot en coordenadas (x_0, y_0), para que luego el robot efectuara una trayectoria en línea recta de longitud L. Debido a los errores generados por la diferencia de diámetros entre las llantas la trayectoria será ligeramente curvilínea. Para reducir la influencia de los errores no sistemáticos, el experimento se debe repetir cinco veces. Una vez efectuado el experimento se calcula el promedio de los valores de

[20] Xiaogang, Ruan. Yalei Li y Xiaoqing Zhu, Kinematic Parameter Calibration of Two-wheeled Robot. En: IEEE. (Agosto, 2012); Pág 2.

la posición final (x_1, y_1), asumiendo β como el ángulo de cabeceo del robot tenemos[21]:

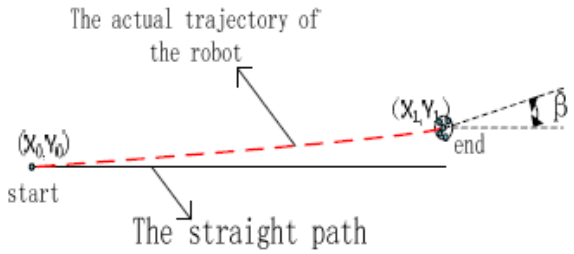
$$X_1 = X_0 + L\cos(\beta/2) \approx L \tag{96}$$

$$Y_1 = Y_0 + L\sin(\beta/2) \approx L\beta/2 \tag{97}$$

Y luego despejando el valor de β:

$$\beta = \frac{2Y_1}{L} \tag{98}$$

Figura 13. Comportamiento del robot en línea recta



Fuente 14. Xiaogang, Ruan. Kinematic Parameter Calibration of Two-wheeled Robot

Con la relación geométrica de la curva del camino curvo del robot, podemos calcular el radio aproximado de la curvatura R.

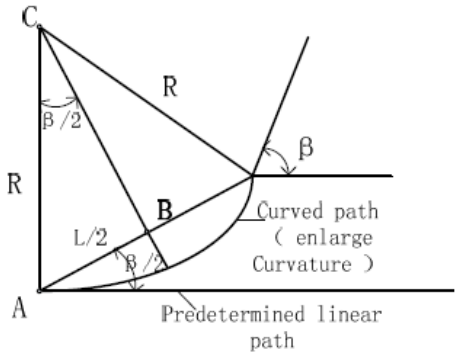
$$R = \frac{L/2}{\sin(\beta/2)} \tag{99}$$

Y luego obtenemos

[21] Ibíd., p.2

Selección de la longitud de la línea recta L

Figura 14. Relación geométrica de la trayectoria curva realizada por un Robot.



Fuente 15.Xiaogang, Ruan. Kinematic Parameter Calibration of Two-wheeled Robot

La selección de la longitud de la línea recta se hace teniendo en cuenta los siguientes criterios:

- Si L es muy largo, el ángulo β será mayor a cinco grados, por lo tanto el experimento es inviable.

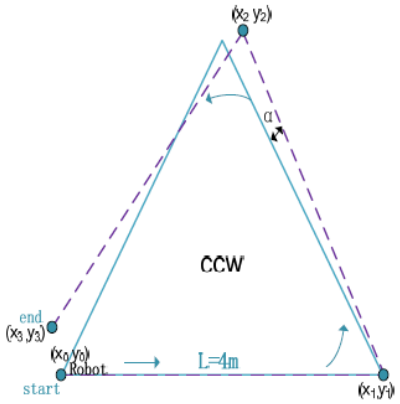
- Si L es muy corto, el cambio de coordenadas en la llegada será muy pequeña y la influencia de los errores aleatorios será muy grande.

Test triángulo equilátero

Como inicialmente se neutralizaron los errores de diámetro gracias al test en línea recta, en este caso solo es necesario efectuar el test triángulo equilátero en un solo sentido, haciendo de este más simple de realizar que el test *UMBmark.* En el experimento del triángulo equilátero el robot gira dos veces con un ángulo de 120°, por lo tanto la desviación del ángulo α incrementará cuando el robot realiza el giro, permitiendo así medir y reducir con mayor precisión el error de base[22].

[22] Xiaogang, Ruan. Yalei Li y Xiaoqing Zhu, op. cit, p.3

Figura 15. Diagrama de un Triángulo equilátero realizado por un robot



Fuente 16.Xiaogang, Ruan. Kinematic Parameter Calibration of Two-wheeled Robot

De la relación geométrica de la figura 13 tenemos:

$$X_1 = X_0 + L \tag{101}$$

A partir de X_1 obtenemos X_2:

$$X_2 = X_1 - \frac{L}{2} + Lsin\alpha \approx \frac{L}{2} + L\alpha \tag{102}$$

Hallando X_2, tenemos X_3:

$$X_3 = X_2 - \frac{L}{2} - Lsin2\alpha \approx -L\alpha \tag{103}$$

Hallando X_3, hallamos el ángulo α en grados:

$$\alpha = \frac{X_3}{-L}\frac{180°}{\pi} \tag{104}$$

Al igual que en *UMBmark*, hallamos la relación de E_b y obtenemos:

$$E_b = \frac{90°}{90° - \alpha} \tag{105}$$

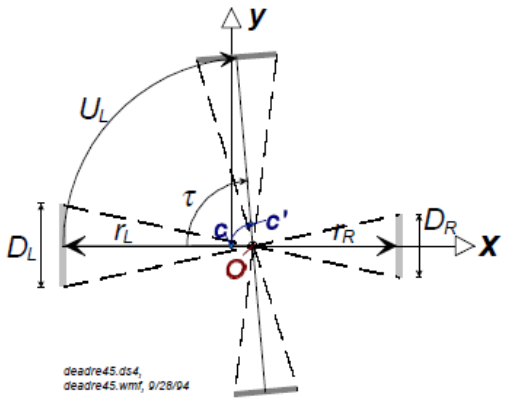
Igualmente podemos obtener el valor real de la distancia entre el punto de contacto de las llantas:

$$B_a = \frac{90°}{90 - \alpha} B_n \tag{106}$$

Al igual que el test UMBmark, las corrección halladas de E_d y E_b se hacen por software teniendo en cuenta la ecuación de desplazamiento (16).

5.2.8 EFECTO DE DIÁMETROS DESIGUALES DURANTE EL GIRO

Figura 16. Desplazamiento del Centro Instantáneo de Rotación CIR



Fuente 17. Borenstein, Johann. UMBmark- A Method for Measuring, Comparing, and Correcting Dead-Reckoning Errors in Mobile Robots

La diferencia de diámetros entre las ruedas, afectan el giro en los robots diferenciales. Para que un robot realice el giro correctamente se requiere que las ruedas giren a la misma velocidad en sentido contrario alrededor del Centro instantáneo de rotación (CIR). Debido al diámetro desigual de las llantas, el CIR es diferente al punto central C, no obstante el giro estará acompañado de un desplazamiento lateral que se anula durante los giro en UMBmark.

$$\frac{r_R}{r_L} = \frac{D_R}{D_L} \tag{107}$$

Siendo r_R y r_L la distancia desde el centro instantáneo de rotación CIR hacia la llanta izquierda y derecha respectivamente. Por lo tanto la distancia b entre las ruedas es:

$$b = r_R + r_L \tag{108}$$

Despejando la ecuación (107) para la llanta derecha obtenemos:

$$r_R = \frac{D_R}{D_L} r_L \tag{109}$$

Reemplazando la ecuación (109) en la ecuación (110) obtenemos:

$$r_L = \frac{D_L}{D_R + D_L} b \tag{110}$$

Para calcular la distancia lineal equivalente a un giro de 90°, corresponde a:

$$U_{n,90} = \frac{\pi b}{4} \tag{111}$$

A razón del número de pulsos, el giro de 90° está dado por:

$$N_{n,90} = \frac{nC_e}{4D_n} b \tag{112}$$

De las relaciones geométricas de la *figura14 t*enemos que:

$$\frac{U_L}{\tau} = \frac{2\pi r_L}{360°} \tag{113}$$

Despejando τ obtenemos:

$$\tau = \frac{360°}{2\pi r_L} U_L \tag{114}$$

U_L Es el recorrido de la llanta izquierda con diámetro D_L. Para hallar U_L debemos calcularlo a partir de los pulsos del encoder $N_{n,90}$ tomando de nuevo la ecuación 2.2:

$$U_L = \frac{\pi D_L}{nC_e} N_{n,90} \tag{115}$$

Sustituyendo la ecuación (111) en la ecuación (115) obtenemos:

$$U_L = \frac{\pi D_L}{4D_n} b \tag{116}$$

Sustituyendo:

$$\tau = \frac{360°}{8r_L} \frac{D_L}{D_n} b \tag{117}$$

La relación del promedio del diámetro al actual ángulo del giro es:

$$\tau = \frac{45°(D_R + D_L)}{D_n} \tag{118}$$

Por lo tanto el promedio del diámetro de la llanta actual es:

$$\frac{D_{avrg}}{\tau} = \frac{D_n}{90°} \tag{119}$$

De las anteriores ecuaciones se puede concluir:

- El error de orientación debido a la desigualdad de los diámetros de las llantas es independiente de cual rueda es más grande o más pequeña que el diámetro nominal. Por lo tanto el mismo error será experimentado en ambos sentidos, horario y anti horario durante el giro. El error es tipo A

- El error de orientación depende del promedio actual del diámetro $D_{avrg} = (D_L + D_R)/2$

Si $D_{avrg} > D_n$ Luego el vehículo girara más que la cantidad nominal.

Si $D_{avrg} < D_n$ el vehículo girara menos

- El error de orientación durante el giro no depende de la relación del diámetro actual $E_d = D_R/D_L$, más bien E_d tiene un menor efecto en la posición x e y del centro C, por que el actual centro de rotación C', no coincide con C.

5.2.9 Medición y corrección del escalamiento Es

Nosotros menciónanos ese error de escalamiento Es, es fácil de medir con una cinta de medida. Un simple procedimiento, requiere que el robot sea programado para ir en camino recto por ejemplo, tres metros. La marca de los experimentos inicia y se detiene posicionándolos en el suelo. Usando la cinta de medida la distancia entre las dos marcas puede ser medida con una precisión aproximada de más o menos 1mm 0.03% a full escala para el ejemplo de los tres metros. La actual distancia es luego comparada de la distancia calculada, basado en el error Es es definido:

$$E_S = L_{calc}/L_{actual} \quad (120)$$

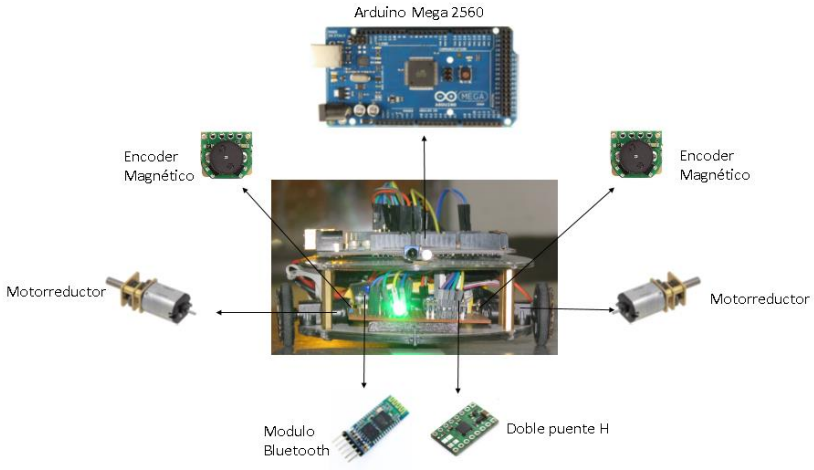
La compensación por el error de escalamiento Lineal es lograda reemplazando el original diámetro nominal por el calculado con el Es mediante la ecuación:

$$D_n *= E_S D_n \quad (121)$$

6 METODOLOGIA

6.1 CONSTRUCCION DEL ROBOT DIFERENCIAL “*HENRYTWO”*

Figura 17. Robot Diferencial Experimental "HenryTwo"



Fuente 18. Propia

“HenryTwo” es un robot diferencial prototipo construido específicamente para las pruebas de errores de odometría. Su construcción se hizo manualmente, por lo tanto es susceptible a errores de ensamblaje, que si bien pueden ser muy pequeños, aportan errores significativos en la realización de los experimentos.

Se decidió construir el robot debido a que actualmente en el mercado no existen robots a un precio asequible, que cumplan con las especificaciones mínimas para llevar a cabo este tipo de proyectos. El chasis de este robot está construido con discos compactos reciclados, lo que permite una reducción costos considerables y una estructura lo suficientemente consistente para los requerimientos de las pruebas.

Diámetro nominal de las llantas: 32mm

Distancia nominal entre las ruedas: 130mm

Reducción de motores: 100:1

Resolución de los encoder: 6 Pulsos/Revolución del motor

Alimentación: Batería LiPo 1000mAh 7.4V

Comunicación: Bluetooth

Microcontrolador: Arduino Mega 2560

6.1.1 Micromotor Motorreductor

El robot consta de un par de micromotores (con eje extendido para la adecuación de los encoder) con una relación de reducción 100.37:1 y un torque de 2.2 Kg-cm. Estos micromotores son ideales para los experimentos de odometría, debido a su pequeño tamaño y a su capacidad de funcionar a bajas velocidades. Como se mencionó anteriormente, esto es recomendable a la hora de realizar las pruebas, para evitar deslizamientos y derrapes.

Figura 18. Motorreductor

Fuente 19. https://www.pololu.com/product/2204

6.1.2 Sensores encoder

Los sensores encoder son dispositivos capaces de transformar la posición angular de un eje (ya sea de un motor) en pulsos digitales y con ayuda de un microcontrolador es posible interpretar estos pulsos en velocidad, aceleración, posición etc.

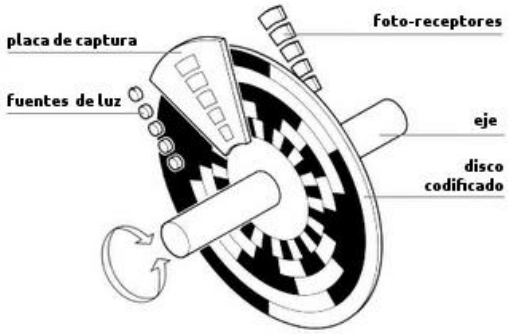
Tipos de encoder

Respecto al tipo de medición, existen principalmente dos tipos de encoder, encoder absoluto y encoder incremental:

- **Encoder absoluto**

En un encoder absoluto el disco presenta un patrón único que permite relacionarlo con una posición igualmente única del eje, por lo tanto tiene mayor precisión que un encoder incremental, permitiendo así al operador conocer el punto de inicio y final de un eje. Los encoder absoluto se reclasifican en encoder absoluto de vuelta simple y encoder absoluto de vuelta múltiple.

Figura 19. Encoder Absoluto

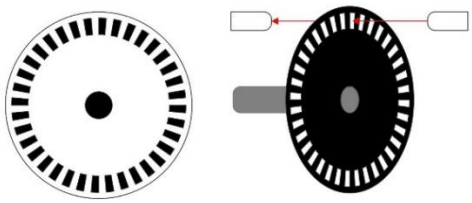


Fuente 20.http://www.lbaindustrial.com.mx/que-es-un-encoder-absoluto/

- **Encoder incremental**

Las mediciones de este tipo de encoder están determinadas por la cantidad de pulsos generados, por cada interrupción de un haz de luz o un disco magnético. De igual manera se requiere de un micro controlador que interprete dichas interrupciones y determine velocidad, aceleración Etc. Estos encoder presentan como ventaja bajo costo y fácil adquisición en el mercado.

Figura 20. Encoder Incremental



Fuente 21.http://profeitm.blogspot.com.co/2016/05/encoders-incrementales-y-absolutos.html

Encoder Magnético

Los encoder magnéticos proveen información del desplazamiento del robot ya que detectan los pulsos generados por la pastilla magnética en cada giro. La lectura de los discos magnéticos se realiza gracias a dos sensores de efecto hall. Los pulsos obtenidos se transforman en distancia recorrida gracias al microcontrolador del robot. La transformación de estas distancias es realizada según la ecuación (16).

La resolución de este encoder está dada gracias a que contiene 6 polos magnéticos por pastilla, entregando así 6 pulsos por revolución que multiplicado por la relación de reducción de los micro motores equivalente a 100.37:1, se obtiene una resolución de 602.22 pulsos por revolución de la llanta (0.59°/Pulso).

Figura 21. Encoder Magnético

Fuente 22.https://www.pololu.com/product/2598

6.1.3 Microcontrolador

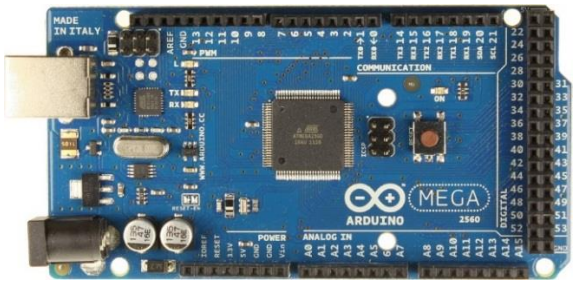
Un microcontrolador es un circuito programable para una tarea específica y a su vez interactúa otros componentes por lo que reduce significativamente el consumo de energía, peso, volumen entre otros. En la actualidad existen diversas placas de desarrollo que contiene microcontroladores y los componentes necesarios para el correcto funcionamiento sin necesidad de usar protoboard o componentes adicionales, convirtiéndolos en dispositivos muy versátiles.

Placa de desarrollo Arduino Mega

Para el desarrollo del robot *Henrytwo* se utilizó la placa de desarrollo Arduino Mega 2560 la cual está basado en el micro controlador ATmega2560. Algunos aspectos a resaltar de esta placa son sus 54 puertos digitales que la componen, de los cuales 15 pueden ser usados como salidas de PWM, 16 entradas analógicas con una resolución de 1024 bits, 4 puertos físicos UART (Universal Asynchronous Receiver-Transmiter). Contiene un oscilador de 16MHz, conexión I2C y conexión USB para la programación de la placa.

Esta placa es ideal para la construcción del robot diferencial, gracias al gran número de puertos digitales, puertos con interrupciones (Para la lectura de los encoder) y puertos seriales para la comunicación mediante Bluetooth.

Figura 22. Arduino Mega 2560



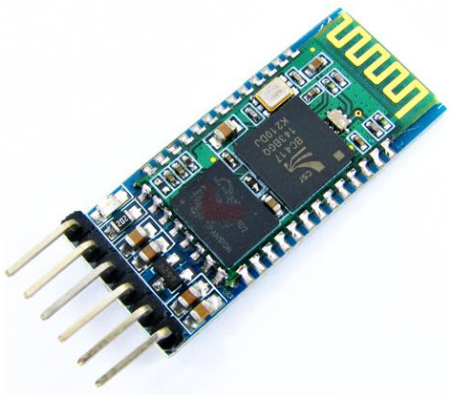
Fuente 23.https://www.arduino.cc/en/Main/ArduinoBoardMega2560

6.1.4 Comunicación

La comunicación de *Henrytwo* se realiza mediante el Módulo bluetooth *HC-05*, el cual permite configurarse como maestro o esclavo, la configuración de este dispositivo se puede realizar mediante comandos *AT,* por donde podemos configurar el nombre del dispositivo, velocidad de trasmisión, cambiar la contraseña de acceso Etc.

El objetivo de esta comunicación es tener la posibilidad de conectarse a un computador portátil para inicializar y escoger los test. La selección de los test se realiza mediante una interfaz gráfica (GUI).

Figura 23. Modulo Bluetooth Hc-05



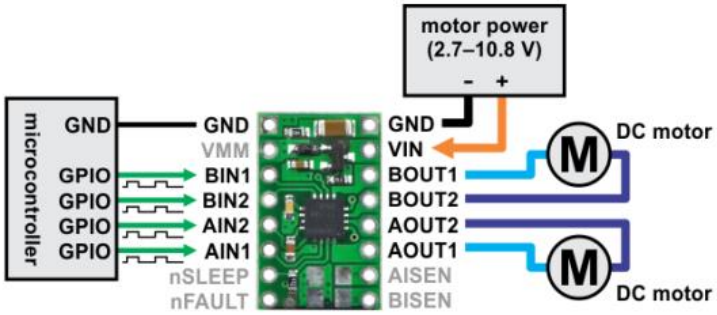
Fuente 24.http://blog.roman-mueller.ch/index.php/2013/04/24/connecting-to-arduino-using-processing-and-bluetooth-hc-05-hc06/

6.1.5 Motor driver

Para el acoplamiento de los micromotores y de la placa de desarrollo Arduino Mega 2560, se utilizó el Dual motor driver DRV8833 de Pololu, el cual consiste en dos puentes H que permite controlar los motores en ambos sentidos de giro. Este dispositivo no requiere dispositivos adicionales para su funcionamiento y su implementación es bastante sencilla. Se conectan cuatro entradas PWM que son las que definirán el sentido de giro y la velocidad de los motores.

Este dispositivo presenta como ventajas sus pequeñas dimensiones, bajo costo, protección de temperatura, corrientes altas además de bajos niveles de tensión.

Figura 24. Configuración Motor Driver



Fuente 25.https://www.pololu.com/product/2130

6.2 Control automático del robot

Los robots diferenciales son muy susceptibles a cambiar de trayectoria deliberadamente, si no se implementa un controlador apropiado. El control automático del robot se hace con la finalidad de que los dos motores del robot giren a la misma velocidad. La falta de un controlador para los motores, ocasionan erráticos imposibilitando la medición de errores generados por odometría.

6.2.1 Caracterización de motores del robot

Para la caracterización de los motores se introdujo una entrada escalón a la planta, observando mediante los sensores encoder el comportamiento de la misma. Para obtener mejores resultados de la caracterización, se tomaron las siguientes condiciones:

- Se les aplicó carga a cada uno de los motores, equivalente al peso del robot que cada rueda del motor debe soportar en condiciones normales, es decir se ubica el robot en el suelo.
- Al momento de realizar las observaciones, se aplicó al motor contrario un valor igual de PWM para aproximar las condiciones a las que estaría sometido el motor objeto de estudio.
- La caracterización se realizó con el robot de pruebas totalmente terminado.

6.2.2 Caracterización del motor derecho

De la caracterización del motor derecho se obtuvieron los siguientes datos:

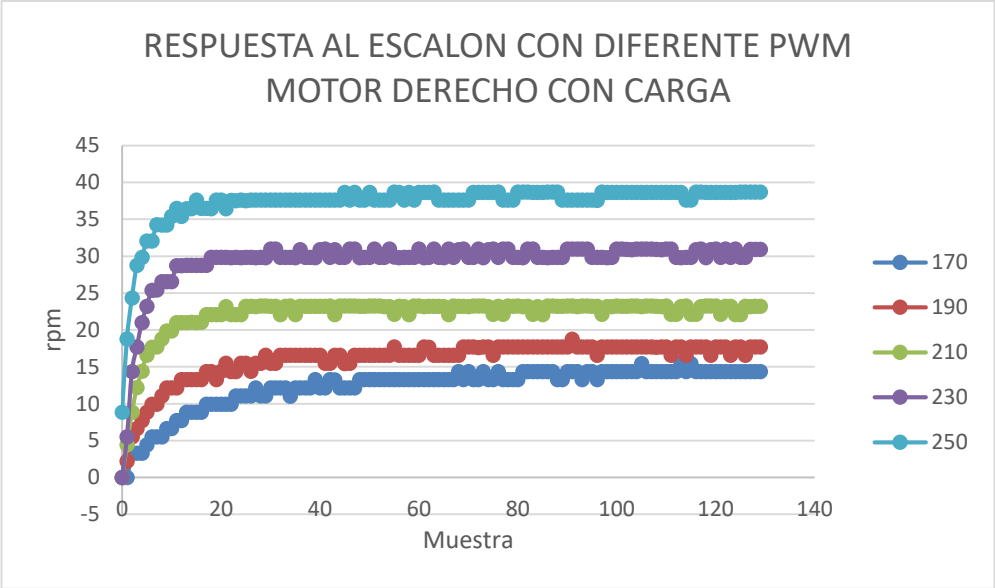
Tabla 2.Datos de caracterización del motor derecho

Muestra	**170**	**190**	**210**	**230**	**250**	**CURVA**	
15ms	**PWM**	**PWM**	**PWM**	**PWM**	**PWM**	**PROM**	**TIEMPO**
	RPM	**RPM**	**RPM**	**RPM**	**RPM**		ms
0	0	0	0	0	0	0	0
1	0	2.21	4.42	5.53	8.84	4.2	0.015
2	3.32	5.52	8.85	14.38	18.8	10.174	0.03
3	3.32	6.64	12.18	17.71	24.35	12.84	0.045
4	3.32	7.73	14.39	21	28.78	15.044	0.06
5	4.43	8.85	16.6	23.22	29.85	16.59	0.075
6	5.53	9.94	17.68	25.37	32.05	18.114	0.09
7	5.53	9.94	17.69	25.43	32.05	18.128	0.105
…	…	…	…	…	….	…	….
127	14.36	17.69	23.19	30.93	38.71	24.976	1.905
128	14.36	17.7	23.2	30.92	38.7	24.976	1.92
129	14.36	17.7	23.23	30.94	38.69	24.984	1.935

Fuente 26. Propia

Graficando el número de muestra vs las revoluciones por minuto por cada valor de PWM, se obtiene la siguiente gráfica:

Figura 25. Respuestas del motor derecho con diferentes valores de PWM



Fuente 27. Propia

Promediando el resultado de las curvas se obtuvo la siguiente gráfica:

Figura 26. Respuesta promedio motor derecho

Fuente 28. Propia

Aplicando el criterio del 63.3% del valor de establecimiento, la función de transferencia obtenida del motor derecho es:

$$\frac{C(s)}{U(s)} = \frac{25}{0{,}0685s + 1} \tag{121}$$

Con tiempo de muestreo de *0.015s*, la función de transferencia en tiempo discreto es:

$$G(Z) = \frac{4.917}{Z - 0.8033} \tag{122}$$

6.2.3 Caracterización del motor izquierdo

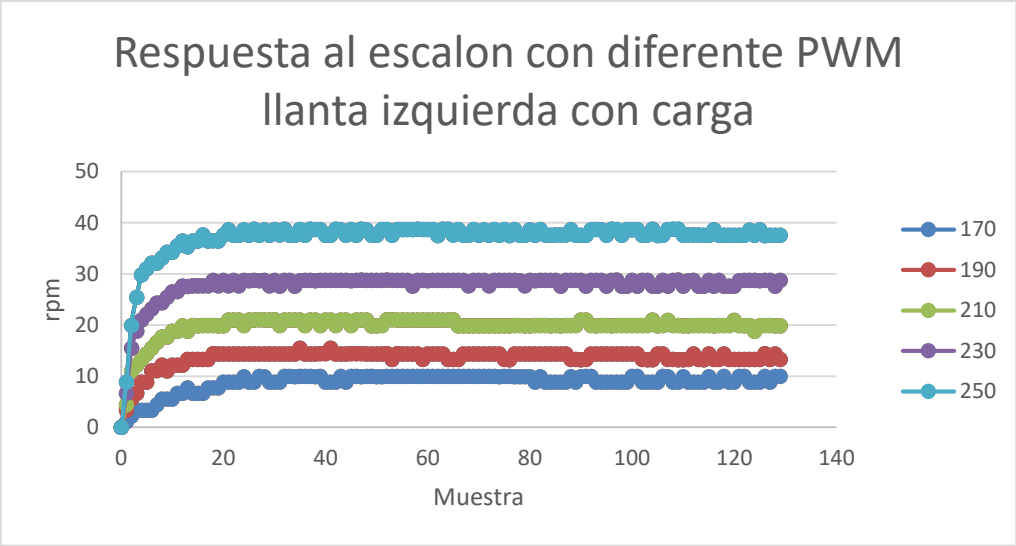
Tabla 3.datos de caracterización del motor izquierdo

Muestra	170	190	210	230	250		
15ms	PWM	PWM	PWM	PWM	PWM	PROM	TIEMPO
	RPM	RPM	RPM	RPM	RPM		ms
0	0	0	0	0	0	0	0
1	1.11	3.32	4.42	6.63	8.86	4.868	0.015
2	2.21	5.53	11.06	15.48	19.92	10.84	0.03
3	3.32	6.64	12.16	18.77	25.44	13.266	0.045
4	3.32	8.86	13.27	20.98	29.79	15.244	0.06
5	3.32	8.84	14.38	22.11	31	15.93	0.075
6	3.31	11.06	15.49	23.15	32.09	17.02	0.09
7	4.43	11.07	16.59	24.3	32.09	17.696	0.105
.	.	.	.	.	.	.	.
127	8.84	13.26	19.9	28.77	37.57	21.668	1.905
128	9.96	14.38	19.88	27.62	37.57	21.882	1.92
129	9.95	13.26	19.86	28.77	37.57	21.882	1.935

Fuente 29. Propia

Al igual que el motor derecho se grafica el número de muestra vs las revoluciones por minuto por cada valor de PWM, obteniendo la siguiente gráfica:

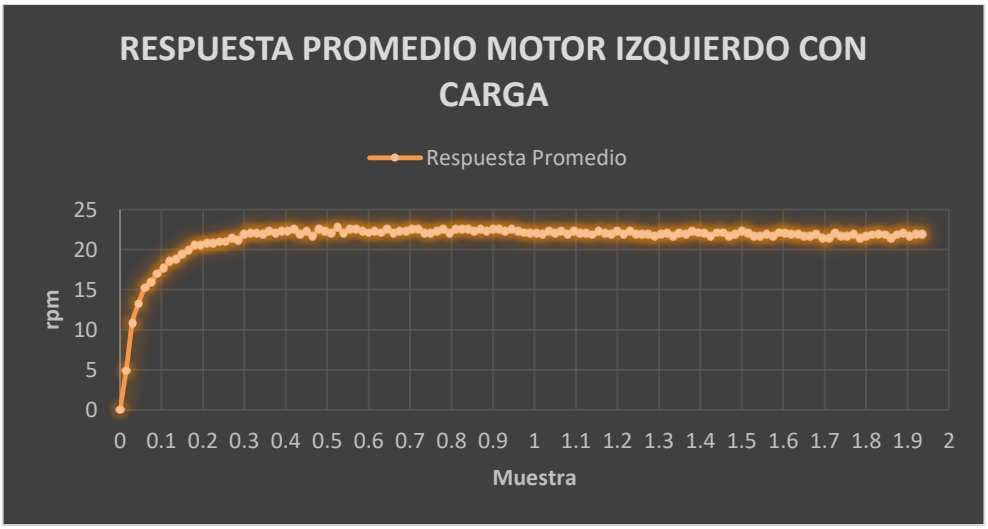
Figura 27. Respuestas del motor izquierdo con diferentes valores de PWM



Fuente 30. Propia

Promediando el resultado de las curvas, se obtiene:

Figura 28. Respuesta promedio motor izquierdo



Fuente 31. Propia

Aplicando el criterio del 63.3% del valor de establecimiento, la función de transferencia obtenida del motor izquierdo es:

$$\frac{C(s)}{U(s)} = \frac{22}{0{,}055s + 1} \quad (123)$$

Con tiempo de muestreo de *0.015s*, la función de transferencia en tiempo discreto es:

$$G(Z) = \frac{0.5919}{Z - 0.9731} \quad (124)$$

6.3 Calculo de PID

Para el cálculo del controlador PID se obtuvieron los valores de las constantes $K_{p,}T_i$ y T_d , mediante el primer método de Ziegler-Nichols.

Las constantes del motor derecho son:

	Kp	*Ti*	*Td*
PID	2.74	0.06	0.015

Y la ecuación del controlador del motor derecho es:

$$\frac{0.041s^2 + 2.73s + 45.54}{s} \quad (125)$$

Las constantes del motor izquierdo son:

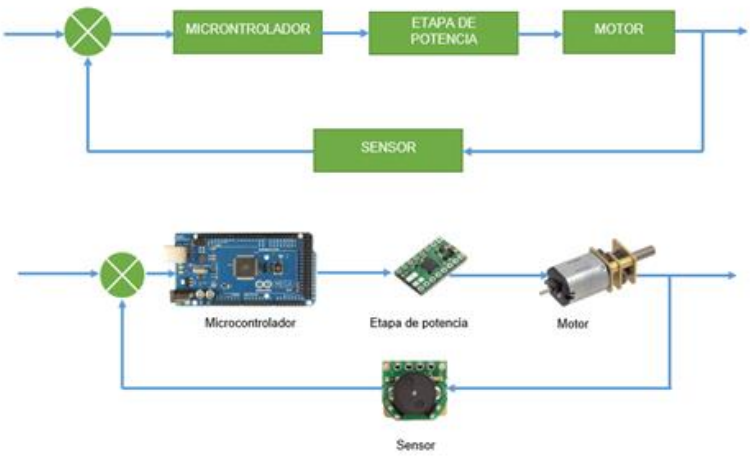
	Kp	*Ti*	*Td*
PID	2.20	0.06	0.015

Y la ecuación del controlador del motor izquierdo es:

$$\frac{0.33s^2 + 2.19s + 36.65}{s} \quad (126)$$

El control de velocidad de cada motor consiste en el siguiente esquema.

Figura 29. Diagrama de bloques de los motores



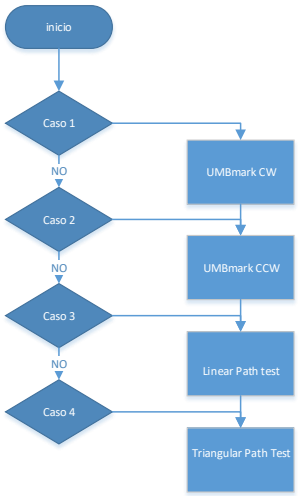
Fuente 32. Propia

6.4 Algoritmos

6.4.1 Selector de Test

Para poder ejecutar los test del robot sin necesidad de reprogramarlo, se ha implementado un selector *Switch Case*, que permite elegir la prueba a realizar. La selección se realiza mediante bluetooth de un equipo portátil de forma remota.

Figura 30. Algoritmo del selector (Arduino)



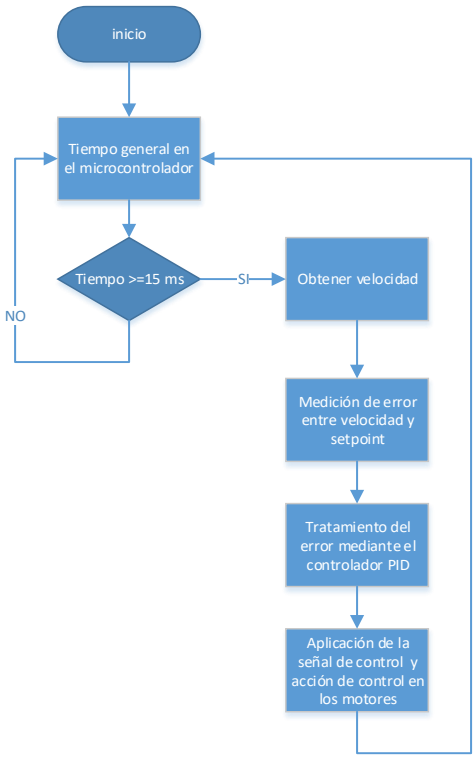
Fuente 33. Propia

6.4.2 Controlador PID

El tiempo del microcontrolador está a razón de la función *millis()* que genera un conteo de *0* hasta *50* días, la utilización de esta función nos permite generar el tiempo de muestreo que es de 15 ms.

Al cumplirse la condición de tiempo, el microcontrolador toma los valores de velocidad de cada una de las ruedas aplicando las correcciones respectivas para lograr una determinada velocidad la cual es controlada mediante señales de *pwm* realización correcciones cada *15ms.*

Figura 31. Algoritmo de control PID



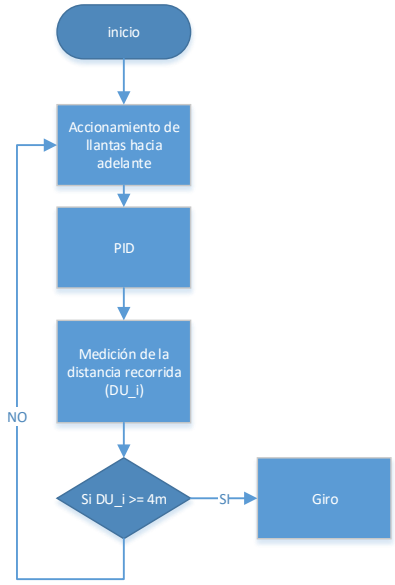
Fuente 34. Propia

6.4.3 Línea recta

Uno de las tareas más importante que debe realizar el robot es recorrer trayectorias en línea recta, ya que permite medir los errores generados por odometría, estas trayectorias en línea recta se realizan para las pruebas UMBmark y triangular.

El cálculo del desplazamiento se realiza con la *ecuación (22)* Esta medición es importante hacerla ya que nos permite dimensionar las pruebas y determinar las distancia para realizar los giros.

Figura 32. Algoritmo de línea recta

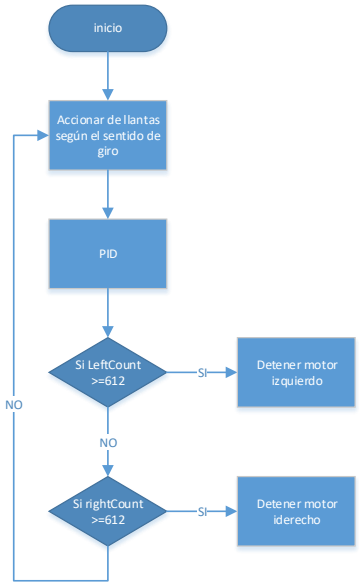


Fuente 35. Propia

6.4.4 Giro del robot

El robot para realizar un giro, las llantas deben funcionar en sentido inverso un respecto a la otra, teniendo en cuenta el sentido del giro que se quiera realizar. La cantidad de pulsos que debo tener para realizar un giro de θ grados esta descrito por la *ecuación (122).*

Figura 33. Algoritmo de giro



Fuente 36. Propia

El criterio para la selección de los experimentos se tomó de acuerdo a las limitaciones del robot.

6.5 INTERFAZ GRAFICA GUI

La realización de una interfaz gráfica GUI facilita los cálculos de los experimentos realizados, igualmente permite tener conocimiento que test está corriendo sin necesidad de reprogramar el robot.

Las principales características de esta interfaz Gráfica GUI son:

- Permite obtener fácilmente las constantes de corrección cL, cR y Ba.
- Para obtener las constantes cL, cR y Ba se debe ingresar cinco coordenadas, consecuencias de cinco experimentos por cada sección de los Test.
- Cada sección del Test se ejecuta automáticamente en el robot al momento de presionar el botón "***Iniciar"***.
- Cada sección del Test permite visualizar los datos ingresados en cada sección del Test en un plano cartesiano.
- El Test *UMBmark* El Test *Triangular Path Test*, se divide en dos partes, en el *Linear Path Test* y *Triangular Path Test*.

6.5.1 Programa *Inicio*

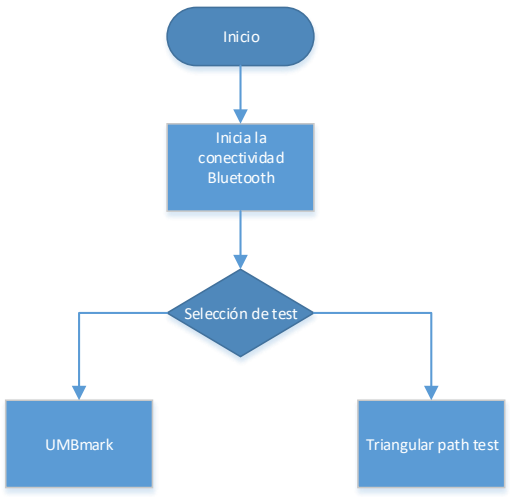
Este programa nos permite seleccionar la prueba a realizar, ya sea el experimento UMBmark o Triangular (*Triangular Path Test*). Antes de ejecutar los experimentos se debe activar la comunicación con el Robot mediante el botón “Enlazar Robot” para seleccionar la respectiva prueba.

Figura 34. Pantalla principal del programa "Inicio"

Fuente 37. Propia

El algoritmo del *Programa de inicio*, es el siguiente:

Figura 35. Algoritmo del programa "Inicio"

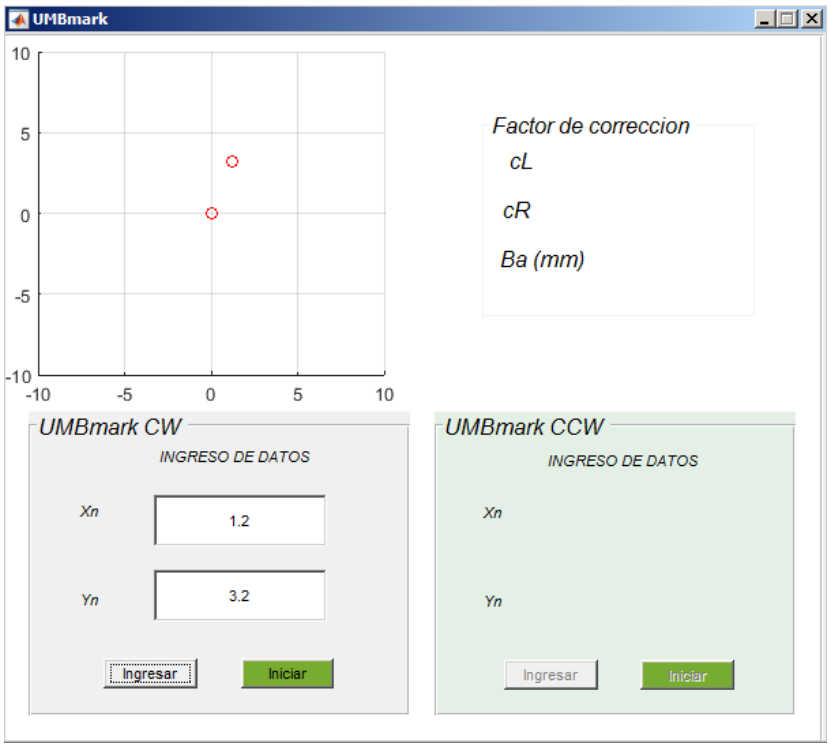


Fuente 38. Propia

6.5.2 *UMBMARK* GUI

Al seleccionar de la pantalla de inicio la opción *UMBmark*, emerge una ventana donde podemos ingresar los datos de los experimentos realizados. Este Test se divide en dos partes, *UMBmark CW* y *UMBmark CCW*, obligando al operador ingresar los datos del Test en este mismo orden. Para distinguir las coordenadas de cada sección de los experimentos, se determina los círculos rojos los datos obtenidos del *UMBmark CW* sentido horario y cruces negras para el Test *UMBmark CCW* sentido anti horario.

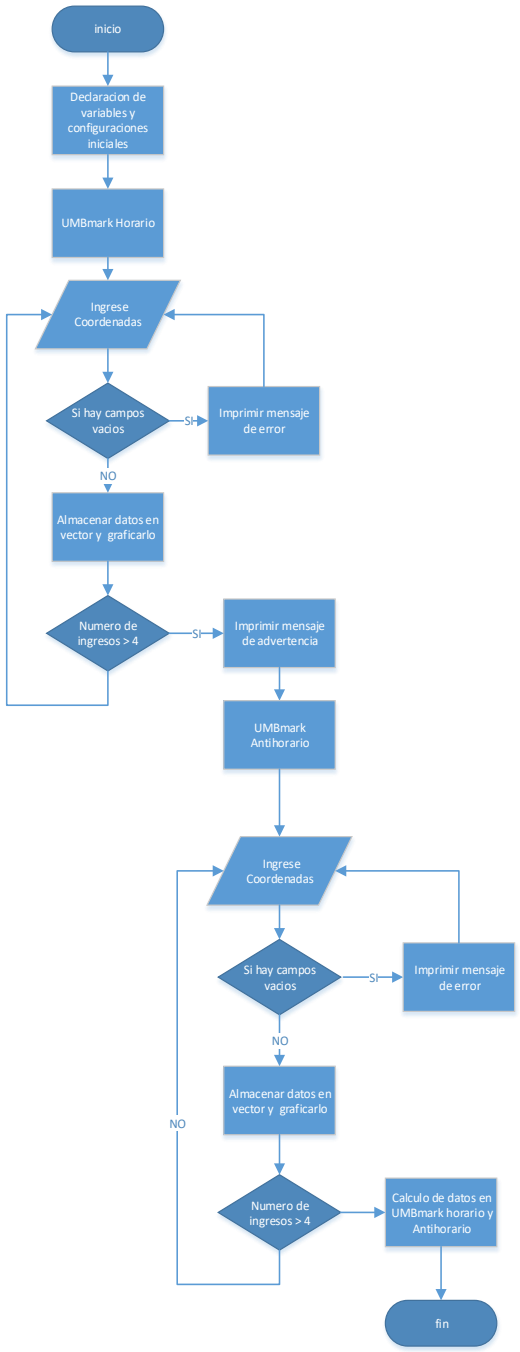
Figura 36. Interfaz gráfica para cálculos "UMBmark"



Fuente 39. Propia

Diagrama de flujo de la interfaz *UMBmark* es el siguiente:

Figura 37. Algoritmo "UMBmark"

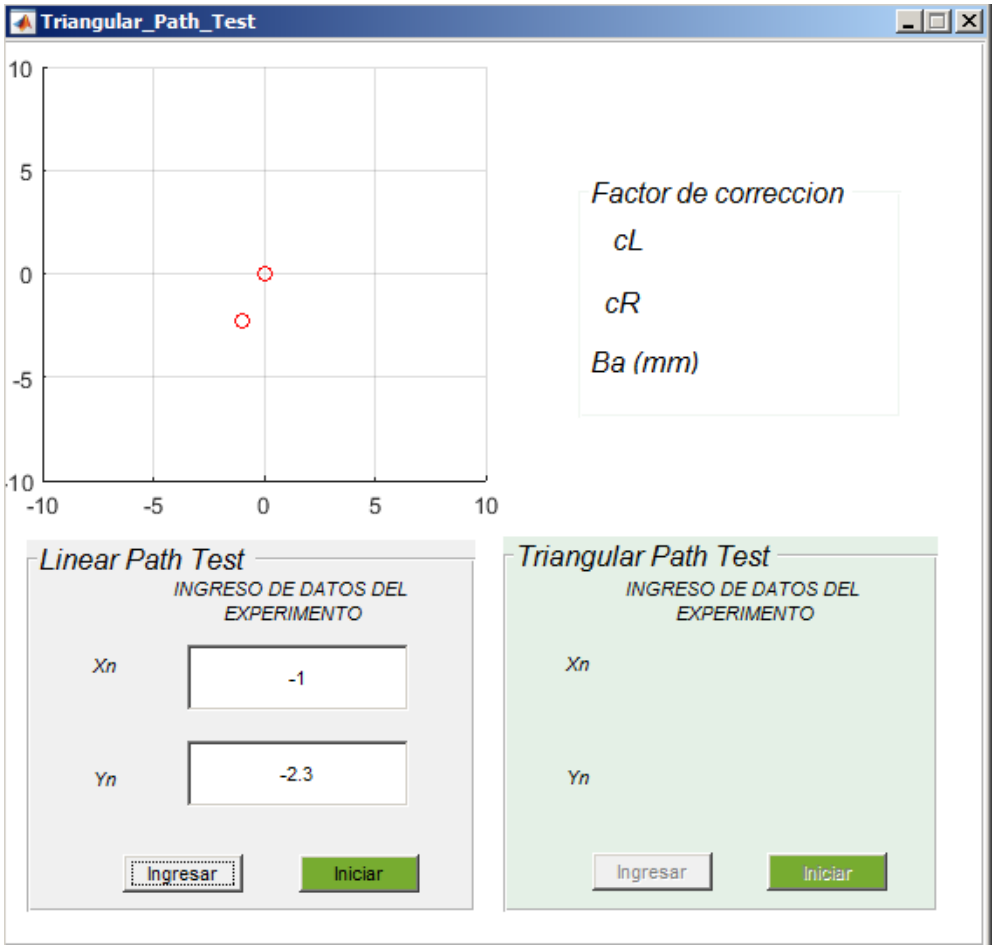


Fuente 40. Propia

6.5.3 *Triangular Path Test* GUI

Esta interfaz es muy similar a la interfaz del test UMBmark, solo que con su respectivo orden se ejecutan las pruebas. Para mejorar las distinciones de la sección de pruebas se tomó como los datos ingresados del *Linear Path Test* como círculos rojos y el *Triangular Path Test* como cruces negras.

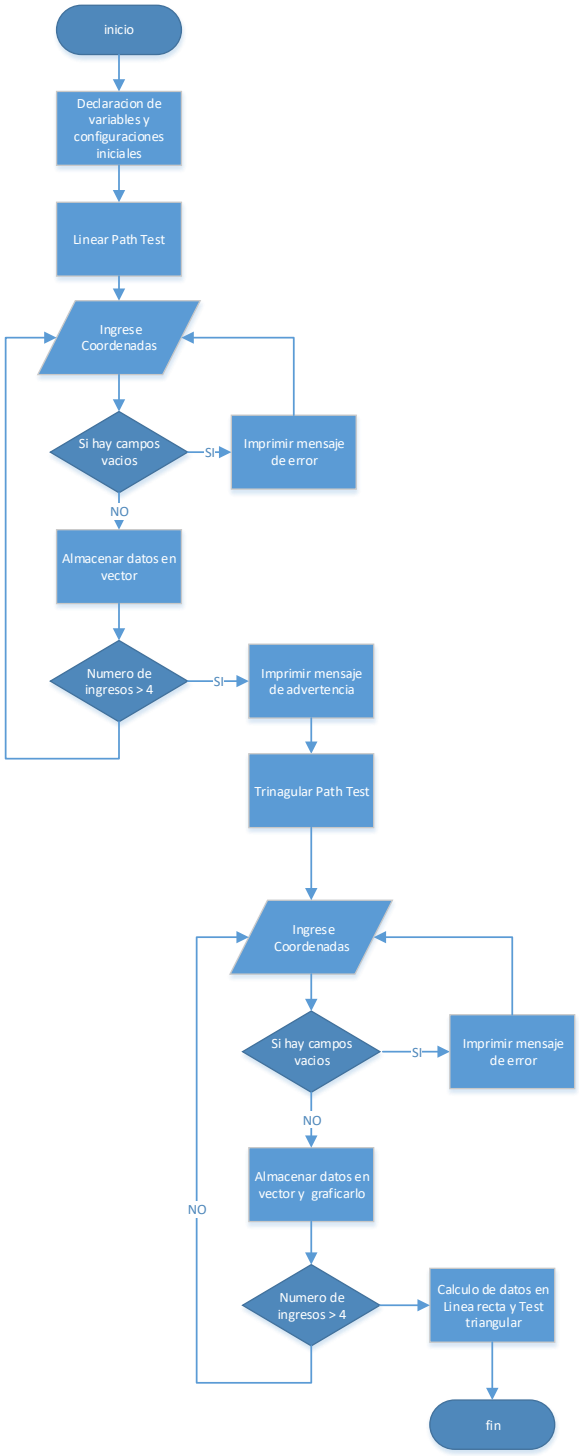
Figura 38. Interfaz gráfica para cálculos del Triangular Path Test



Fuente 41. Propia

Algoritmo Triangular Path test:

Figura 39. Algoritmo Triangular Path Test



Fuente 42. Propia

6.6 Ejecución de los experimentos UMBmark reducido y Triangular Path Test reducido.

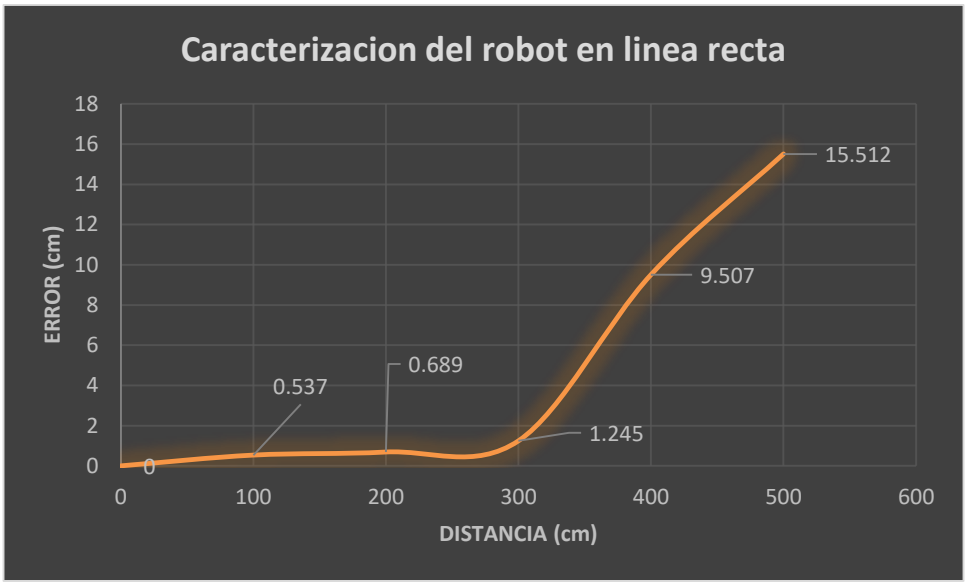
6.6.1 Acondicionamiento del robot para los experimentos

Figura 40. Características del robot de pruebas

Fuente 43. Propia

En la *figura 38* Se aprecia algunos errores de ensamblaje que aportan errores e imprecisiones en el comportamiento del robot al momento de efectuar los experimentos. En este caso la “desviación de llantas” afecta directamente en los errores de base E_b cambiando la distancia de la base real entre llantas del robot.

Figura 41. Caracterización del robot en Línea recta



Fuente 44. Propia

Teniendo en cuenta el tamaño del robot y el diámetro de las llantas, se realizan pruebas a diferentes distancias con el fin de conocer el comportamiento del robot además de la influencia de los errores sistemáticos y no sistemáticos en una misma superficie. Para este experimento se tomaron muestras desde 1 hasta 5 metros de lo cual se obtuvo la *gráfica 39* donde se observa el comportamiento del robot de las distancias con incremento de 1 metro por experimento. Los errores obtenidos en las pruebas de 3, 4 y 5 metros son demasiado grandes para ejecutar el test con estas distancias. Se decide realizar las pruebas *UMBmark* y *Triangular Path Test* con la distancia de 2 metros, debido a que los errores a esta distancia son lo suficientemente grandes para obtener correcciones considerables de dichas pruebas de corrección de odometría.

Antes de la realización del test *UMBmark* se realizó el ajuste del error de escalamiento E_s, el cual es sugerido por *Boreinstein*. A excepción del error de escalamiento E_s, los parámetros de la *tabla 4* del robot son aplicados en ambos experimentos.

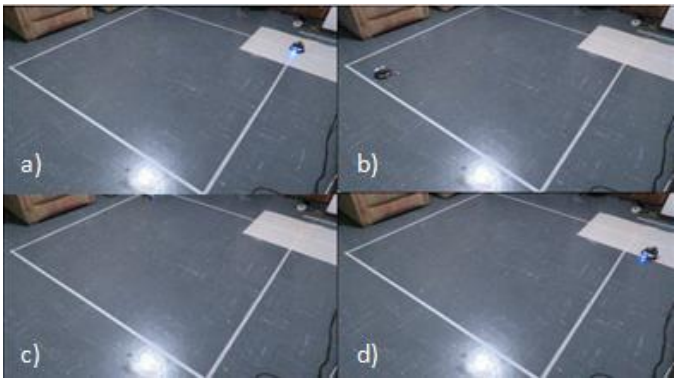
Tabla 4. Parámetros de inicio del robot.

Diametro nominal (mm)	D_n	32
Diametro nominal*(mm)	D_n *	31.9
Factor de conversion	C_m	0.166933
Factor de conversion*	C_m *	0.166327
Resolucion del encoder	C_e	6
Relacion de reduccion	n	100.37

Fuente 45. Propia

6.6.2 EXPERIMENTO UMBmark REDUCIDO

Figura 42. Experimento UMBmark



Fuente 46. Propia

Del experimento UMBmark en sentido horario y anti horario, se obtuvieron los siguientes resultados:

Del experimento UMBmark, se obtuvieron los siguientes resultados:

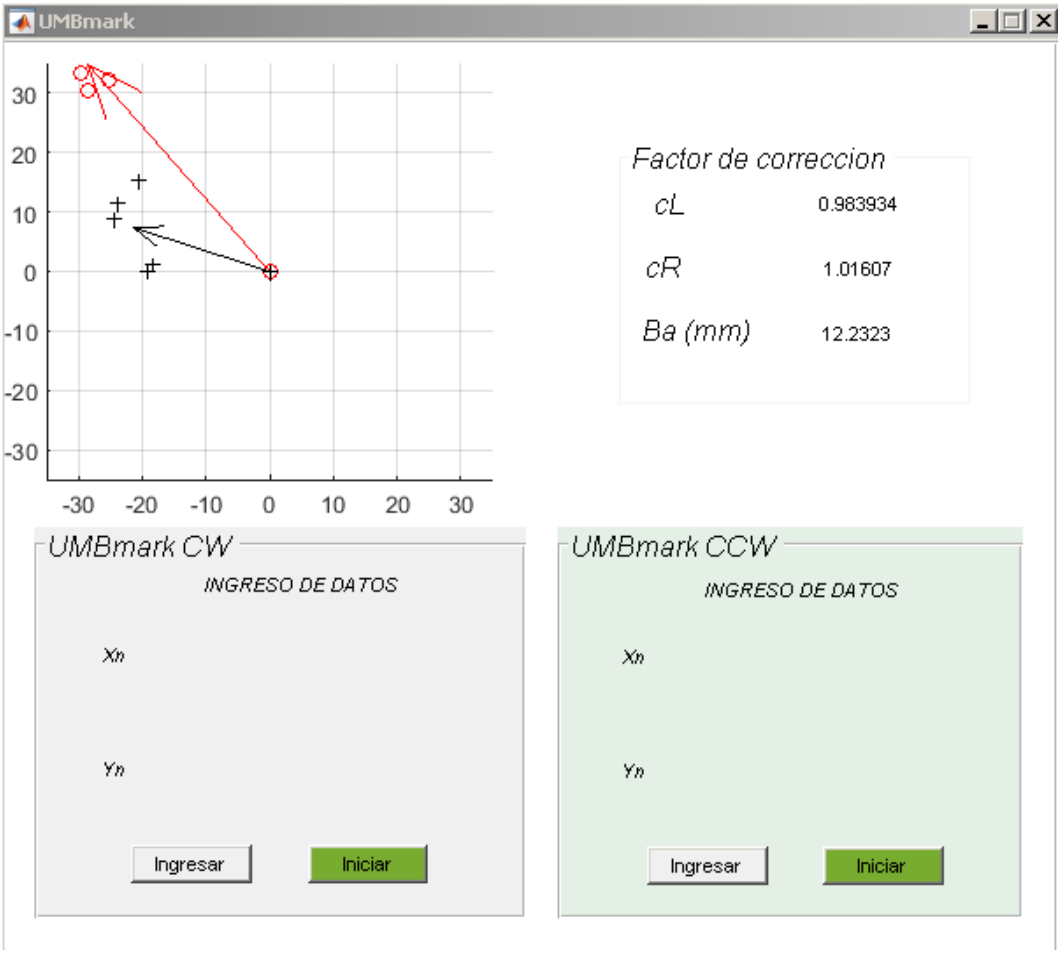
- **Experimento 1.**

Tabla 5. Experimento 1 UMBmark

	CW1			CWW1	
1	-28.6	30.5	1	-18.4	1.1
2	-25.2	32.2	2	-24.4	9
3	-36.1	38.5	3	-20.6	15.3

4	-29.7	33.4	4	-23.9	11.4
5	-23.1	39.6	5	-19.1	-0.1

Figura 43. Experimento 1 UMBmark reducido.



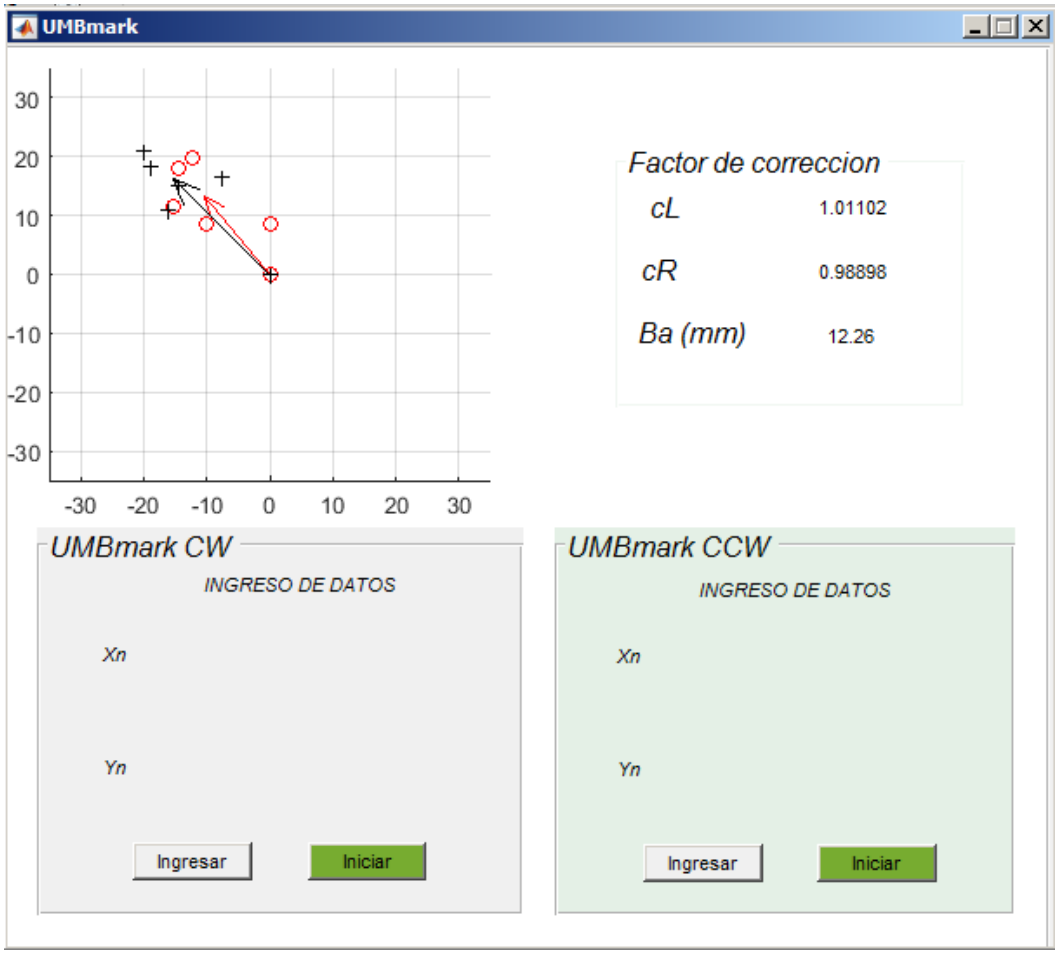
Fuente 47. Propia

- **Experimento 2.**

Tabla 6. Experimento 2 UMBmark

CW2			CWW2		
1	-14.4	18	1	-7.5	16.4
2	0.2	8.5	2	-20	21
3	-10.1	8.6	3	-19.1	18.4
4	-15.3	11.4	4	-16.1	10.9
5	-12.2	19.8	5	-14.5	15

Figura 44. Experimento 2 UMBmark reducido



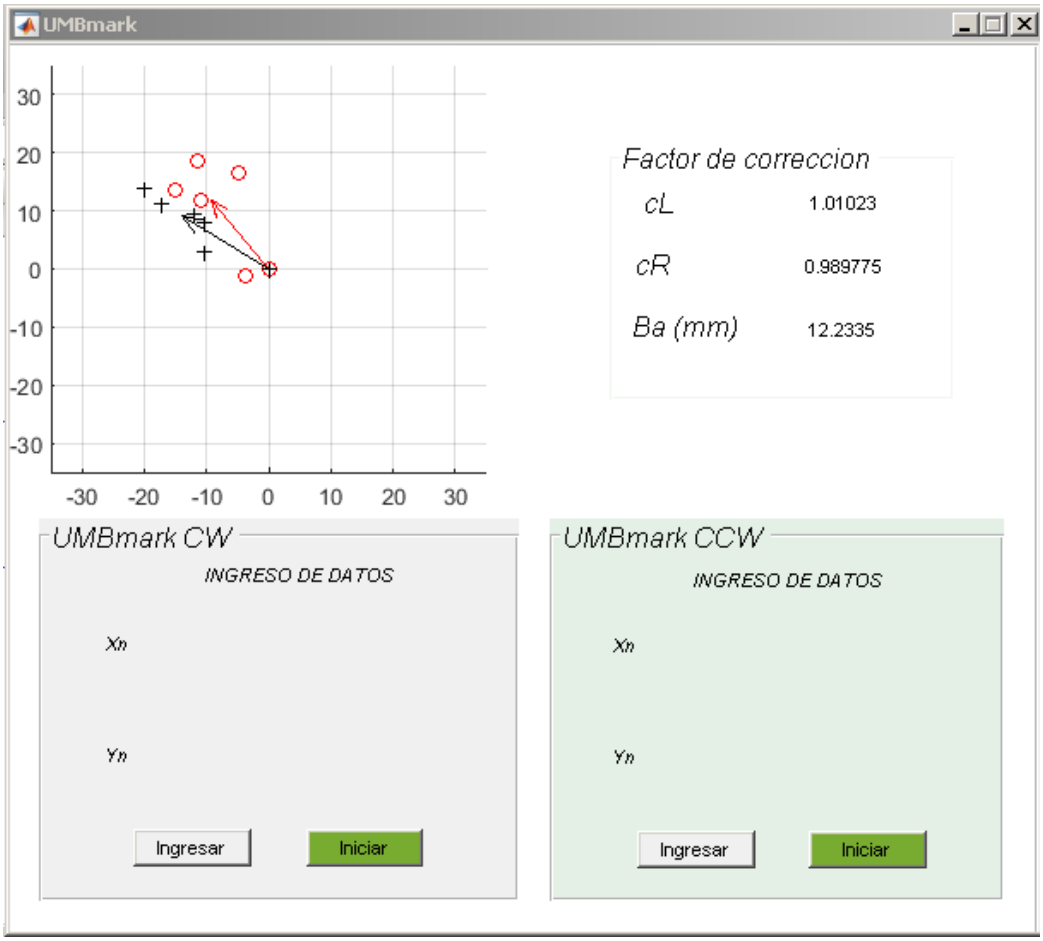
Fuente 48. Propia

- **Experimento 3.**

Tabla 7. Experimento 3 UMBmark

CW3			CWW3		
1	-11.5	18.6	1	-20	13.8
2	-11	11.8	2	-12	9.4
3	-4.9	16.5	3	-10.3	8.1
4	-15	13.7	4	-10.2	3.1
5	-3.8	-1.3	5	-17.1	11.3

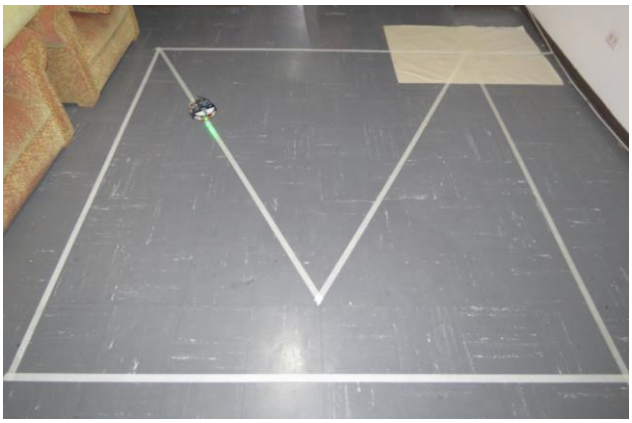
Figura 45. Experimento 3 UMBmark reducido



Fuente 49. Propia

6.6.3 EXPERIMENTO TRIANGULAR PATH REDUCIDO

Figura 46. Triangular Path Test



Fuente 50. Propia

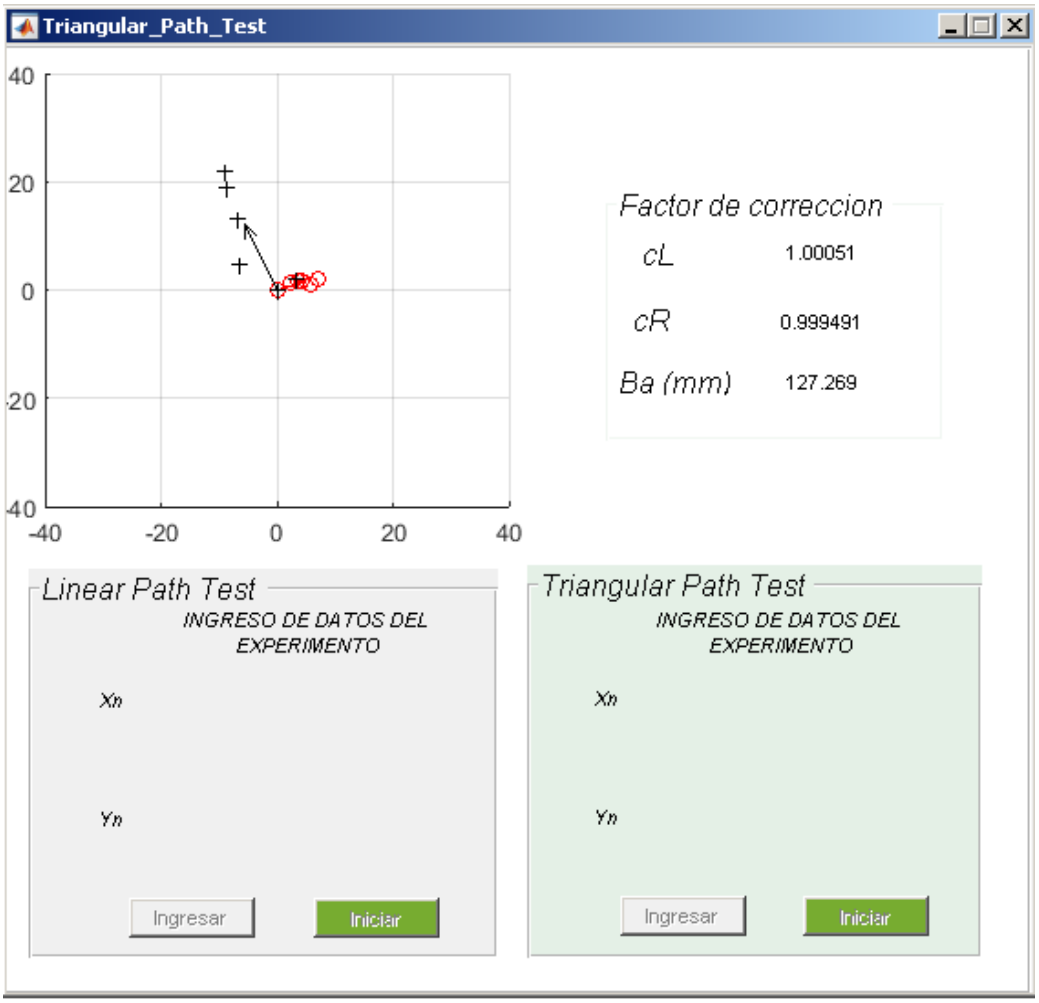
En el experimento Triangular Path Test no se tiene en cuenta el error de escalamiento. Inicialmente se ejecuta el Test en línea recta para corregir los errores de diámetro E_d y luego se ejecuta el recorrido triangular para corregir los errores de base E_b.

- **Experimento 1.**

Tabla 8. Experimento 1 Triangular Path

L1			TP1		
1	5.8	1	1	3.3	1.9
2	7.1	1.9	2	-6.9	13.2
3	3.6	1.8	3	-8.8	19
4	2.5	1.5	4	-9.1	22
5	4.4	1.8	5	-6.5	4.7

Figura 47. Experimento 1 Triangular Path Test reducido



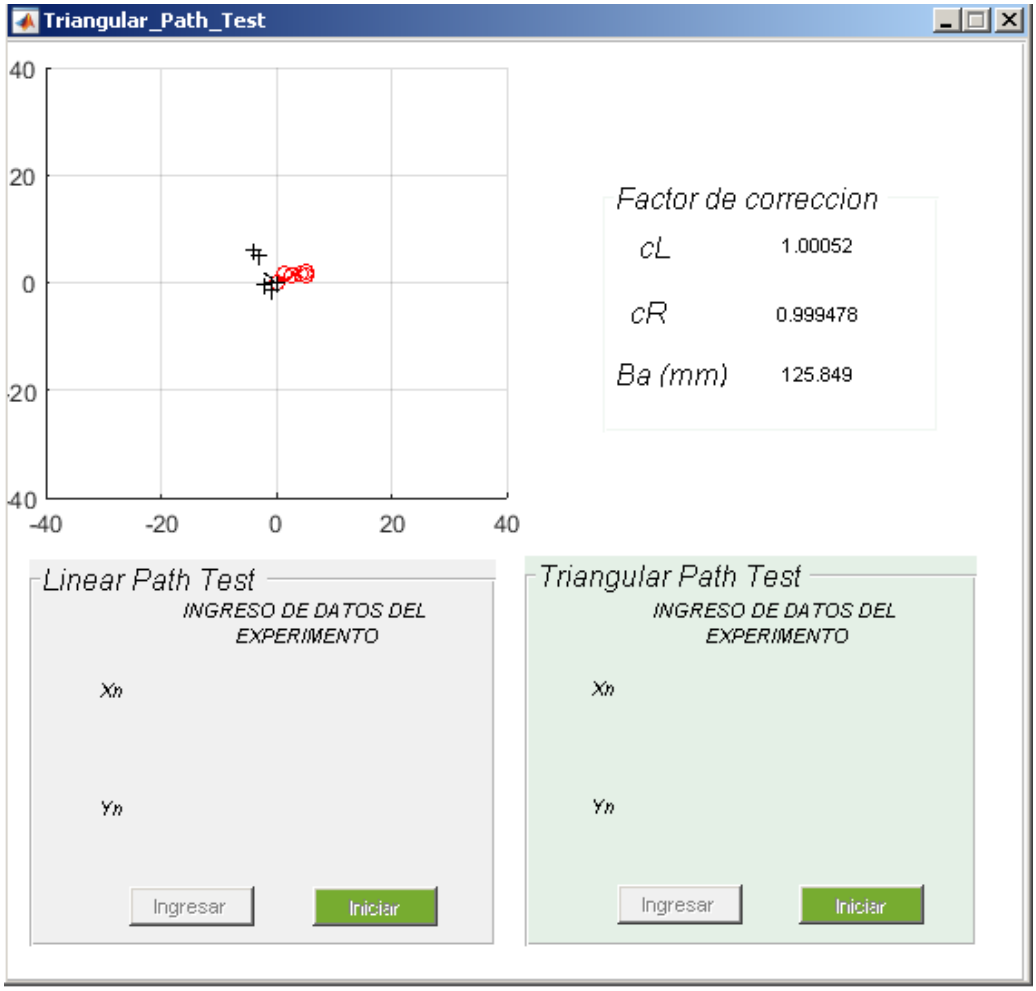
Fuente 51. Propia

- **Experimento 2.**

Tabla 9. Experimento 2 Triangular Path

	L2			TP2	
1	5.2	1.5	1	-2.9	5
2	4.2	1.6	2	-3.9	6.2
3	1.3	1.7	3	-0.9	-0.1
4	5.2	1.9	4	-2	-0.5
5	2.8	1.5	5	-0.9	-1.4

Figura 48. Experimento 2 Triangular Path Test reducido



Fuente 52. Propia

RESULTADOS

Tabla 10. Resultados

	Ba antes (mm)	Ba después (mm)	Corrección Ba	Erro Antes (cm)	ERROR después (cm)	Corrección Posición
UMBMARK	130	122.3	5.69%	33.77	15.84	53.09%
TRIANGULAR	130	127.6	2.15%	13.38	2.71	79.70%

7 CONCLUSIONES

Los robots diferenciales como *Henrytwo*, son muy susceptibles a los errores no sistemáticos por lo tanto, la trayectoria del robot se puede ver alterada por cualquier perturbación u obstáculo presente en el entorno de pruebas. Este problema se solucionó gracias a la implementación de un controlador PID, sin afectar la posterior medición y corrección de errores de odometría.

Inicialmente, se realizaron pruebas con el Test UMBmark y el Triangular Path un la distancia propuesta por *Johan Boreinstein de* $L = 4$ metros y con una medida de base inicial de 130mm (tomada del centro de cada llanta), como resultado los errores obtenidos fueron muy grandes por el orden de los 110 cm, por esta razón se tomó la decisión de caracterizar el robot y estudiar su comportamiento para implementar el *Test UMBmark reducido (TUR) y Triangular Path reducido (TPR).* Debido a los errores de construcción presentes en el robot se decidió tomar otra consideración; en lugar de tomar los 130 mm como base nominal, se tomó 125 mm que corresponde a la medida interna de cada llanta.

De los experimentos; del *TPR* se obtuvieron mejores resultados que del *TUR* (79.70% y 53.09% respectivamente), pero la proporción de errores corregidos por el *TUR* (17.93 cm) es mucho mayor que el TPR (10.67 cm). Aunque los resultados obtenidos del *TUR* están muy influenciados por los errores no sistemáticos, al momento de corregir los errores de base fueron mejores que el *TPR*, ya que el robot Henrytwo se aproxima demasiado con la corrección obtenida (122.3 mm), En cambio con el *TPR* se obtuvo una distancia de base (127.6 mm).

El robot de pruebas *Henrytwo* presentó mayor influencia de los errores de base E_b debido a los errores de ensamblaje observados en la figura 38 (página 66). A pesar de estas fallas en la construcción, se obtuvo muy buenos resultados al momento de realizar correcciones respectivas tanto en el *TUR* y el. En el caso de los errores de diámetro son muy pequeños debido a la inexistente deformación que presenta el caucho por desbalance en las cargas de cada llanta. El desbalance de carga no afecta la velocidad de cada llanta debido a la presencia del controlador PID implementado.

Los errores sistemáticos afectan en función del diámetro de las llantas y el *TUR* es un experimento que contempla este criterio tomado del *Triangular Path* test original. TUR es el test ideal para la corrección de errores en "*Henrytwo"*, sobre todo para la corrección en los errores de base E_b.

8 TRABAJOS FUTUROS

En la realización de los experimentos no se tuvo en cuenta la variable θ que corresponde al ángulo de salida y llegada del robot, el cual es considerado en la odometría, pero no es fundamental para la medición y corrección de los errores mediante la ejecución de los experimentos *UMBmark reducido y Triangular Path reducido*. La implementación de una brújula electrónica nos puede dar una referencia valida siempre y cuando la diferencia entre el punto de salida y llegada no sea demasiado grande.

En lo que concierne al ensamble y diseño del robot es posible mejorar en gran medida el chasis mediante un software de diseño, que permita ubicar los puntos de perforación y distribución de los componentes de forma precisa. En un entorno académico se puede lograr un muestreo exacto de forma remota en tiempo real, para establecer la posición exacta con respecto al punto de referencia inicial, esto es posible gracias a las ecuaciones provistas en el Test UMBmark.

En la interfaz gráfica para el cálculo, es posible realizar mejoras en la inserción de datos de corrección de odometría con tal de reducir a cero la reprogramación del robot en cada experimento. Al igual que incluir una opción donde se logre visualizar los datos insertados y/o modificar los datos allí almacenados.

9 BIBLIOGRAFIA

J.Borenstein and L.Feng. UmbMark: A Metod for Measurig, Comparing and Correct ing Dead-reckonin Errors in Mobile Robots. Universidad de Michigan: 1994.

2016. Arduino. *Arduino-Genuino.* [Online] Arduino, 2016. https://www.arduino.cc.

BORENSTEIN Johann, Correction of Systematic Odometry Errors in Mobile Robots. En IEEE (Oct. 1995).

J.Boreinstein, H. R. Everett, Where am I? Sensors and Methods for Mobile Robot Positioning. 1996.

Kuo, Benjamin. Sistemas de control automatico,7 ed. Madrid: Prentice Hall, 1996.

2016. Learn ! Do! ArduinoInfo.info. [Online] 2016. https://arduino-info.wikispaces.com/Arduino-PWM-Frequency.

LOPEZ, Shadai, calibración de robots de locomoción diferencial por el método de mínimos cuadrados empleando la técnica de suavizado y cartografía incremental, Pátzcuaro, 2011, 13-17p. Tesis (Maestro En Ciencias En Ingeniería Eléctrica). Universidad Michoa.

Mike Bosse, Marco Hutter, Martin Rufli, Davide Scaramuzza, (Margarita Chli, Paul Furgale), José "Autonomous Mobile Robots". {En línea}.{10 julio de 2015} disponible en internet :(https://edge.edx.org/courses/course-v1%3AETHZ%2BAMRx_Internal1_2015).

Ogata, Katsuhiko. Ingenieria de control moderna, 5 ed. Madrid: Prentice Hall, 2010.

2016. Pololu Robotics & Electronics. [Online] Pololu, 2016. https://www.pololu.com/.

SIEGWART, Roland. Introduction to Autonomous Mobile Robots. The MIT Press.

TANVEER Abbas, Measurement and Correction of Systematic Odometry Errors Caused by Kinematics.

THRUN, Sebastian. Probabilistic robotics.Edicion Borrador.1999-2000.

2016. Mathworks . [Online] Matlab, 2016. https://www.mathworks.com

ANEXO A

Código del robot

```
//CONTROL PID DE LA LLANTA DERECHA E IZQUIERDA



#define RH_ENCODER_A 3
#define RH_ENCODER_B 5
#define LH_ENCODER_A 2
#define LH_ENCODER_B 4

#define VERDE 10
#define AZUL  11
#define ROJO  12



// variables to store the number of encoder pulses
// for each motor
volatile long leftCount = 0;
volatile long rightCount = 0;



//Variables de odometría


float Constante = 0.1669339526; //constante hallada mediante ecuaciones de
Odometria
float DU_R = 0;
float DU_L = 0;
float DU_i = 0;
float DU_ = 0;
float D_TETA = 0;
float Distancia=400; //Distancia en milimetros
int b = 130; //Distancia entre llantas en milimetros
int Flag_Giro_D = 0;
int Flag_Giro_I = 0;
int Flag_Contador = 0;
int Selector_Exp;
int flag_Arranque = 0;

//Variables de tiempo

double Tiempo_s = 0;
int Flag_One = 1;
double Tiempo_Actual = 0;
double Muestreo = 0.015;//tiempo de muestro del controlador PID

//Variables de velocidad llanta derecha

double Conteo_Actual_D = 0;
double Conteo_D = 0;
double Pulsos_Min_D = 0;
float RPM_Enc_D = 0;
float RPM_Mot_D = 0;
float Frecuencia_D = 0;
```

```
//Variables de velocidad llanta izquierda

double Conteo_Actual_I = 0;
double Conteo_I = 0;
double Pulsos_Min_I = 0;
float RPM_Enc_I = 0;
float RPM_Mot_I = 0;
float Frecuencia_I = 0;

//Variables de control llanta derecha

double Error_D = 0; //Diferencia entre Sepoint(Set_D) y valor actual de velocidad
double Set_D = 14.47; //Setpoint llanta derecha
double U_D = 0;
double P_D = 0;
double I_D = 0;
double D_D = 0;
double pwm_D = 50;
double Kp_D = 2.74;   //Proporcional  llanta derecha
double Ki_D = 45.66;  //Constante integral llanta derecha
double Kd_D = 0.0411; //Constante derivativa llanta derecha
double Error_Pasado_D = 0;
double Error_Anterior_D = 0;
double ErrorD_D = 0;



//Variables de control llanta izquierda

double Error_I = 0;//Diferencia entre Sepoint(Set_I) y valor actual de velocidad
double Set_I = 15;
double U_I = 0;
double P_I = 0;
double I_I = 0;
double D_I = 0;
double pwm_I = 50;
double Kp_I = 2.20;//Proporcional  llanta izquierda
double Ki_I = 36.666;//Constante integral llanta izquierda
double Kd_I = 0.033;//Constante derivativa llanta izquierda
double Error_Pasado_I = 0;
double Error_Anterior_I = 0;
double ErrorD_I = 0;

//Variables de nivel de bateria

float Nivel = 0;


void setup() {


  TCCR2B = TCCR2B & B11111000 | B00000001;  //PWM a 31372.55 Hz Puertos 9 y 10
  TCCR4B = TCCR4B & B11111000 | B00000001;  //PWM a 31372.55 Hz Puertos 6,7 y 8

  Serial.begin(9600); //Puerto serial para monitoreo USB
  Serial1.begin(9600); //Puerto serial para comunicacion Bluetooth

  //Declaracion como entradas los puertos para el encoder izquierdo
  pinMode(LH_ENCODER_A, INPUT);
```

```
  pinMode(LH_ENCODER_B, INPUT);

  //Declaracion como entradas los puertos para el encoder derecho
  pinMode(RH_ENCODER_A, INPUT);
  pinMode(RH_ENCODER_B, INPUT);

  //Declaracion de puertos como salida para medicion de nivel de bateria

  pinMode(VERDE, OUTPUT);
  pinMode(AZUL , OUTPUT);
  pinMode(ROJO , OUTPUT);

  //Entrada analoga A0 para medicion del divisor de voltaje para el nivel de
bateria

  pinMode(A0, INPUT);

  attachInterrupt(0, leftEncoderEvent, CHANGE); //Interrupcion encoder izquierdo
  attachInterrupt(1, rightEncoderEvent, CHANGE); //Interrupcion endoder derecho

  Serial.println("Realizacion de test...");
}

void loop() {

  Nivel_Bateria();

  if (Serial1.available()) {


    Selector_Exp = Serial1.read();
    Serial.println(Selector_Exp);



    switch (Selector_Exp) {

      case 2:
       UMBmark_CW();
        break;
      case 3:
        UMBmark_CCW();
        break;
      case 4:
        Linear_Path_Test();
        break;
      case 5:
        Triangular_Path_Test();
        break;

    }
  }
}


void UMBmark_CW() {

  while ((Selector_Exp == 2) && (Flag_Contador <= 3))

  {
```

```
    Adelante_();
    Desplazamiento_();
    Nivel_Bateria();

    if (DU_i >= Distancia) //Si distancia recorrida es igual a 4 metros
    {
      Flag_Contador++;
      Inercia_Stop_Ad();
      Stop_General();
      rightCount = 0;
      leftCount = 0;
      DU_i = 0;
      DU_L = 0;
      DU_R = 0;
      Flag_Giro_I = 1;
      Flag_Giro_D = 1;
      if (Flag_Contador <= 4 ) {
        Giro_CW();

      }
    }
  }
}


void Giro_CW() {

  while ((Flag_Giro_D == 1) || (Flag_Giro_I == 1)) {

    analogWrite(7, pwm_D); // LLANTA DERECHA ADELANTE
    analogWrite(8, pwm_I); // LLANTA IZQUIERDA ATRAS
    PID();

    if ((abs(leftCount)) >= 612) {
      analogWrite(7, 0);
      leftCount = 0;
      Flag_Giro_I = 0;

    }

    if ((abs(rightCount)) >= 612) {
      analogWrite(8, 0);
      rightCount = 0;
      Flag_Giro_D = 0;
    }
  }

  Inercia_Stop_Izq();
  Stop_General();


  //***Reinicio de todas las variables***
  pwm_D = 0;
  pwm_D = 0;
  U_I = 0;
  U_D = 0;
  Error_Pasado_D = 0;
  Error_Anterior_D = 0;
  ErrorD_D = 0;
```

```
  Error_Pasado_I = 0;
  Error_Anterior_I = 0;
  ErrorD_I = 0;
  //Variables de velocidad llanta derecha

  Conteo_Actual_D = 0;
  Conteo_D = 0;
  Pulsos_Min_D = 0;
  RPM_Enc_D = 0;
  RPM_Mot_D = 0;
  Frecuencia_D = 0;

  //Variables de velocidad llanta izquierda

  Conteo_Actual_I = 0;
  Conteo_I = 0;
  Pulsos_Min_I = 0;
  RPM_Enc_I = 0;
  RPM_Mot_I = 0;
  Frecuencia_I = 0;


}

void UMBmark_CCW()
{
  while ((Selector_Exp == 3) && (Flag_Contador <= 3))
  {

    Adelante_();
    Desplazamiento_();
    Nivel_Bateria();

    if (DU_i >= Distancia)
    {
      Flag_Contador++;
      Inercia_Stop_Ad();
      Stop_General();
      rightCount = 0;
      leftCount = 0;
      DU_i = 0;
      DU_L = 0;
      DU_R = 0;
      Flag_Giro_I = 1;
      Flag_Giro_D = 1;
      if (Flag_Contador <= 4 ) {
        Giro_CCW();
      }
    }
  }
}

void Giro_CCW() {

  while ((Flag_Giro_D == 1) || (Flag_Giro_I == 1)) {

    analogWrite(6, pwm_D); // LLANTA DERECHA ADELANTE
    analogWrite(9, pwm_I); // LLANTA IZQUIERDA ATRAS
    PID();
```

```
    if ((abs(leftCount)) >= 612) {
      analogWrite(6, 0);
      leftCount = 0;
      Flag_Giro_I = 0;

    }

    if ((abs(rightCount)) >= 612) {
      analogWrite(9, 0);
      rightCount = 0;
      Flag_Giro_D = 0;
    }
  }

  Inercia_Stop_Der();
  Stop_General();


  //***Reinicio de todas las variables***
  pwm_D = 0;
  pwm_D = 0;
  U_I = 0;
  U_D = 0;
  Error_Pasado_D = 0;
  Error_Anterior_D = 0;
  ErrorD_D = 0;
  Error_Pasado_I = 0;
  Error_Anterior_I = 0;
  ErrorD_I = 0;

  //Variables de velocidad llanta derecha

  Conteo_Actual_D = 0;
  Conteo_D = 0;
  Pulsos_Min_D = 0;
  RPM_Enc_D = 0;
  RPM_Mot_D = 0;
  Frecuencia_D = 0;

  //Variables de velocidad llanta izquierda

  Conteo_Actual_I = 0;
  Conteo_I = 0;
  Pulsos_Min_I = 0;
  RPM_Enc_I = 0;
  RPM_Mot_I = 0;
  Frecuencia_I = 0;


}


void Linear_Path_Test() {
Stop_General();

  while ((Selector_Exp == 4)&&(Flag_Contador == 0))
  {
    Adelante_();
    Desplazamiento_();
```

```
    if (DU_i >= Distancia)
    {
      Inercia_Stop_Ad();
      Stop_General();
      rightCount = 0;
      leftCount = 0;
      DU_i = 0;
      DU_L = 0;
      DU_R = 0;
      Flag_Contador=1;

    }
  }
}

void Triangular_Path_Test() {

  while ((Selector_Exp == 5) && (Flag_Contador <= 2))
  {

    Adelante_();
    Desplazamiento_();

    if (DU_i >= Distancia)
    {
      Flag_Contador++;
      Inercia_Stop_Ad();
      Stop_General();
      rightCount = 0;
      leftCount = 0;
      DU_i = 0;
      DU_L = 0;
      DU_R = 0;
      Flag_Giro_I = 1;
      Flag_Giro_D = 1;
      if (Flag_Contador <= 3 ) {
        Giro_CCW_Triang();
      }
    }
  }
}

void Giro_CCW_Triang() {

  while ((Flag_Giro_D == 1) || (Flag_Giro_I == 1)) {

    analogWrite(6, pwm_D); // LLANTA DERECHA ADELANTE
    analogWrite(9, pwm_I); // LLANTA IZQUIERDA ATRAS
    PID();

    if ((abs(leftCount)) >= 815.5) {
      analogWrite(6, 0);
      leftCount = 0;
      Flag_Giro_I = 0;

    }

    if ((abs(rightCount)) >= 815.5) {
      analogWrite(9, 0);
      rightCount = 0;
```

```
      Flag_Giro_D = 0;
    }
  }

  Inercia_Stop_Der();
  Stop_General();


  //***Reinicio de todas las variables***
  pwm_D = 0;
  pwm_D = 0;
  U_I = 0;
  U_D = 0;
  Error_Pasado_D = 0;
  Error_Anterior_D = 0;
  ErrorD_D = 0;
  Error_Pasado_I = 0;
  Error_Anterior_I = 0;
  ErrorD_I = 0;

  //Variables de velocidad llanta derecha

  Conteo_Actual_D = 0;
  Conteo_D = 0;
  Pulsos_Min_D = 0;
  RPM_Enc_D = 0;
  RPM_Mot_D = 0;
  Frecuencia_D = 0;

  //Variables de velocidad llanta izquierda

  Conteo_Actual_I = 0;
  Conteo_I = 0;
  Pulsos_Min_I = 0;
  RPM_Enc_I = 0;
  RPM_Mot_I = 0;
  Frecuencia_I = 0;
}

void leftEncoderEvent() { // Interrupcion para el encoder izquierdo
  if (digitalRead(LH_ENCODER_A) == HIGH) {
    if (digitalRead(LH_ENCODER_B) == LOW) {
      leftCount++;
    } else {
      leftCount--;
    }
  } else {
    if (digitalRead(LH_ENCODER_B) == LOW) {
      leftCount--;
    } else {
      leftCount++;
    }
  }
}

void rightEncoderEvent() { //Interrupcion para el encoder derecho
  if (digitalRead(RH_ENCODER_A) == HIGH) {
    if (digitalRead(RH_ENCODER_B) == LOW) {
      rightCount++;
    } else {
```

```
      rightCount--;
    }
  } else {
    if (digitalRead(RH_ENCODER_B) == LOW) {
      rightCount--;
    } else {
      rightCount++;
    }
  }
}

void Nivel_Bateria() //determina el nivel de bateria
{
  Nivel = analogRead(A0);
  //Serial.println(Nivel);


  if (Nivel > 942) {

    //azul

    analogWrite(ROJO, 0);
    analogWrite(VERDE, 0);
    analogWrite(AZUL, 255);
  }

  else if ((Nivel > 901) && (Nivel < 942)) {

    //verde

    analogWrite(ROJO, 0);
    analogWrite(VERDE, 255);
    analogWrite(AZUL, 0);
  }

  else if (Nivel < 901) {

    //rojo

    analogWrite(ROJO, 255);
    analogWrite(VERDE, 0);
    analogWrite(AZUL, 0);
  }
}


void PID()// Controlador de los motores
{
  Nivel_Bateria();

  if (Flag_One == 1) {

    Tiempo_Actual = micros();
    Conteo_Actual_D = abs(rightCount);
    Conteo_Actual_I = abs(leftCount);
    Flag_One = 0;


  }
```

```
Tiempo_s = micros() - Tiempo_Actual;


if (Tiempo_s >= 15000) {

  //** PARAMETROS DE VELOCIDAD DERECHA
  Conteo_D = abs(rightCount) - Conteo_Actual_D;
  Frecuencia_D = Conteo_D / (Tiempo_s / 1000000); //Pulsos por segundo
  Pulsos_Min_D = Frecuencia_D * 60; //Pulsos por minuto
  RPM_Enc_D = Pulsos_Min_D / 6; //Revoluciones por minuto del encoder
  RPM_Mot_D = abs(RPM_Enc_D) * (0.0016605);

  Serial.println(RPM_Mot_D);

  //** PARAMETROS DE VELOCIDAD IZQUIERDA

  Conteo_I = abs(leftCount) - Conteo_Actual_I;
  Frecuencia_I = Conteo_I / (Tiempo_s / 1000000); //Pulsos por segundo
  Pulsos_Min_I = Frecuencia_I * 60; //Pulsos por minuto
  RPM_Enc_I = Pulsos_Min_I / 6; //Revoluciones por minuto del encoder
  RPM_Mot_I = abs(RPM_Enc_I) * (0.0016605);

  Flag_One = 1; //Habilito bandera para reiniciar el conteo de los 15ms

  Serial.println(RPM_Mot_I);

  //**Controlador llanta derecha

  Error_D = (Set_D - RPM_Mot_D);
  Error_Pasado_D = Error_D * Muestreo + Error_Pasado_D;
  ErrorD_D = (Error_D - Error_Anterior_D) / Muestreo;
  P_D = Kp_D * Error_D;
  I_D = Ki_D * Error_Pasado_D;
  D_D = Kd_D * ErrorD_D;
  U_D = P_D + I_D + D_D;

  pwm_D = map(U_D, 24, -24, 255, 0);


  if (pwm_D >= 255) // Restriccionde PWM para evitar que suba a mas de 255
  {
    pwm_D = 255;
  }
  else if (pwm_D <= 0)
  {
    pwm_D = 0;
  }


  //**Controlador llanta izquierda

  Error_I = (Set_I - RPM_Mot_I);
  Error_Pasado_I = Error_I * Muestreo + Error_Pasado_I;
  ErrorD_I = (Error_I - Error_Anterior_I) / Muestreo;
  P_I = Kp_I * Error_I;
  I_I = Ki_I * Error_Pasado_I;
  D_I = Kd_I * ErrorD_I;
  U_I = P_I + I_I + D_I;
```

```
    pwm_I = map(U_I, 24, -24, 255, 0);

    Error_Anterior_D = Error_D;
    Error_Anterior_I = Error_I;

    if (pwm_I >= 255) // Restriccionde PWM para evitar que suba a mas de 255
    {
      pwm_I = 255;
    }
    else if (pwm_I <= 0)
    {
      pwm_I = 0;
    }

  }

}

void Adelante_()// Activa motores hacia adelante
{
  analogWrite(7, pwm_D);  ///  LLANTA DERECHA ADELANTE
  analogWrite(9, pwm_I);  /// LLANTA IZQUIERDA ATRAS
  PID();
}


void Desplazamiento_() //Calcula el desplazamiento
{
  DU_R = Constante * rightCount;
  DU_L = Constante * leftCount;
  DU_i = (DU_R + DU_L) / 2;

}

void Inercia_Stop_Ad()//Elimina la inercia
{
  analogWrite(7, 0); //Iguala a cero para desactivar pwm llanta derecha
  analogWrite(9, 0); //Iguala a cero para desactivar pwm llanta izquierda
  delay(20);
  analogWrite(6, 255); //Activa pwm en sentido contrario
  analogWrite(8, 255); //Activa pwm en sentido contrario
  delay(5);
  analogWrite(6, 0); //Iguala a cero para desactivar pwm llanta derecha en
sentido contrario
  analogWrite(8, 0); //Iguala a cero para desactivar pwm llanta izquierda en
sentido contrario
  delay(1000);
}

void Stop_General() //Detiene ambos motores
{
  analogWrite(6, 0); //LLANTA IZQUIERDA ADELANTE
  analogWrite(7, 0); //LLANTA IZQUIERDA ATRAS
  analogWrite(8, 0); //LLANTA DERECHA ADELANTE
  analogWrite(9, 0); //LLANTA DERECHA ATRAS
  delay(1000);
}

void Inercia_Stop_Izq()//Elimina la inercia
{
```

```
  analogWrite(7, 0); // LLANTA IZQUIERDA  ATRAS
  analogWrite(8, 0); // LLANTA DERECHA ADELANTE
  delay(20);
  analogWrite(6, 255);
  analogWrite(9, 255);
  delay(5);
  analogWrite(6, 0);
  analogWrite(9, 0);
  delay(1000);
}
void Inercia_Stop_Der()//Elimina la inercia
{
  analogWrite(6, 0); // LLANTA IZQUIERDA  ADELANTE
  analogWrite(9, 0); // LLANTA DERECHA ATRAS
  delay(20);
  analogWrite(7, 255);
  analogWrite(8, 255);
  delay(5);
  analogWrite(7, 0);
  analogWrite(8, 0);
  delay(1000);
}
```

ANEXO B

Interfaz gráfica -Código de inicio

```
function varargout = Inicio(varargin)
% INICIO MATLAB code for Inicio.fig
%      INICIO, by itself, creates a new INICIO or raises the existing
%      singleton*.
%
%      H = INICIO returns the handle to a new INICIO or the handle to
%      the existing singleton*.
%
%      INICIO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INICIO.M with the given input
arguments.
%
%      INICIO('Property','Value',...) creates a new INICIO or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Inicio_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Inicio_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Inicio

% Last Modified by GUIDE v2.5 15-Sep-2016 15:02:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @Inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT



function Inicio_OpeningFcn(hObject, eventdata, handles, varargin)



handles.output = hObject;

guidata(hObject, handles);

axes(handles.axes1)
handles.imagen=imread('C:\Users\Henry dad\Dropbox\Proyecto Odometria Jhon
Calderon\Fotos HenryTwo\HenryTwo.bmp');
imagesc(handles.imagen)
axis off

function varargout = Inicio_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;


function pushbutton1_Callback(hObject, eventdata, handles) %inicio de
comunicacion del robot con el portatil

global bt;
instrhwinfo('Bluetooth','HC-05')
bt=Bluetooth('HC-05',1);
fopen(bt);

function pushbutton2_Callback(hObject, eventdata, handles) %Seleccion del
test UMBmark

%run('C:\Users\My dear dady\Dropbox\Proyecto Odometria Jhon
Calderon\GUI_HenryTwo\TURN_ON.m')
run('C:\Users\Henry dad\Dropbox\Proyecto Odometria Jhon
Calderon\GUI_HenryTwo\UMBmark.m')



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)%Seleccion del
test Triangular path test
%
run('C:\Users\Henry dad\Dropbox\Proyecto Odometria Jhon
Calderon\Trangular_Path\Triangular.m')
%run('C:\Users\My dear dady\Dropbox\Proyecto Odometria Jhon
Calderon\Trangular_Path\Triangular.m')
```

ANEXO C

Interfaz grafica- UMBmark

```
function varargout = UMBmark(varargin)
% UMBmark MATLAB code for UMBmark.fig
%      UMBmark, by itself, creates a new UMBmark or raises the existing
%      singleton*.
%
%      H = UMBmark returns the handle Serial.begin(9600); to a new
UMBmark or the handle to
%      the existing singleton*.
%
%      UMBmark('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UMBmark.M with the given input
arguments.
%
%      UMBmark('Property','Value',...) creates a new UMBmark or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before UMBmark_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to UMBmark_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help UMBmark

% Last Modified by GUIDE v2.5 15-Sep-2016 18:45:51JCA
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @UMBmark_OpeningFcn, ...
                   'gui_OutputFcn',  @UMBmark_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before UMBmark is made visible.
function UMBmark_OpeningFcn(hObject, eventdata, handles, varargin)

global vect_x
global vect_y
global n;
global n_;
global X_Z;
global Y_Z;
global X_Z_;
global Y_Z_;

handles.output = hObject;

%Configuracion inicial

set(handles.edit5,'String','')
set(handles.edit7,'String','')
set(handles.edit8,'String','')
set(handles.edit9,'String','')
set(handles.edit9,'String','')
set(handles.text12,'String','')
set(handles.text13,'String','')
set(handles.text18,'String','')
set(handles.edit8,'Enable','Inactive')
set(handles.edit8,'Visible','Off')
set(handles.edit9,'Enable','Inactive')
set(handles.edit9,'Visible','Off')
set(handles.pushbutton15,'Enable','off')
set(handles.pushbutton20,'Enable','off')

axis([-40 40 -40 40]) %Limite de los ejes

%Declaracion de variables iniciales

n=0;
n_=0;
vect_x=zeros(5);
vect_y=zeros(5);
Y_Z=0;
X_Z=0;
X_Z_=0;
Y_Z_=0;

guidata(hObject, handles);

function varargout = UMBmark_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;
```

```
function edit5_Callback(hObject, eventdata, handles) %Ingreso datos Xn
UMBmark CW


function edit5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles) %Ingreso datos Yn
UMBmark CW


function edit7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit8_Callback(hObject, eventdata, handles) %Ingreso datos Xn
UMBmark CCW


function edit8_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit9_Callback(hObject, eventdata, handles) %Ingreso datos Yn
UMBmark CCW


function edit9_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function pushbutton14_Callback(hObject, eventdata, handles) %Boton
"Ingresar" del UMBmark CW de la interfaz

grid on

global x_1;
global y_1;
global x;
global y;
global n;
global vect_x;
global vect_y;
global X_Z;
global Y_Z;
global X_Z_prom_CW;
global Y_Z_prom_CW;
global Raiz_CW;

hold on

vect_x=zeros(5); %Defino tamaño de los vectores de zeros en X
vect_y=zeros(5); %Defino tamaño de los vectores de zeros en Y
x_1=get(handles.edit5,'String'); %Captura informacion de las entradas de
los Edit Text de X
y_1=get(handles.edit7,'String'); %Captura informacion de las entradas de
los Edit Text de Y


 if isempty( x_1) || isempty(y_1) %Verifica que no existan casillas
vacias al momento de ingresar los datos

     msgbox('No deben existir campos vacios','Advertencia!') %Error
Emergente si la condicion de campos vacios se cumple

 else %Inicio de almacenamiento de vectores

     n=n+1; %Contador para el almacenamiento de vectores
     x=str2double(x_1); %Convirtiendo variables de String a Double
     y=str2double(y_1); %Convirtiendo variables de String a Double
     vect_x(n)=x; %Almacenando en el vector corrrespondiente
     vect_y(n)=y;%Almacenando en el vector corrrespondiente
     X_Z=X_Z+x; %Sumatoria de las coordenadas ingresadas
     Y_Z=Y_Z+y; %Sumatoria de las coordenadas ingresadas
     axis([-40 40 -40 40]) %Redefinicion de los ejes
     plot(vect_x, vect_y,'ro') %Grafico los puntos de las coordenadas
ingresadas
     hold on %Sostengo la informacion suministrada

 end
```

```
 if n>4; %Cuando se cumple la condicion,  finaliza el alamcenamiento de
vectores y ejecuta lo calculos pertinentes


   X_Z_prom_CW=X_Z/5; %Obtengo el promedio hallado de sumatoria de X
   Y_Z_prom_CW=Y_Z/5; %Obtengo el promedio hallado de sumatoria de Y
   compass(X_Z_prom_CW,Y_Z_prom_CW,'r')  %Grafico vector
   Raiz_CW=sqrt(((X_Z_prom_CW)^2)+((Y_Z_prom_CW)^2)) %Hallo longitud r
del vector
   msgbox('No es posible ingresar mas datos','Advertencia!')


   %Cambiamos las configuraciones una vez se han ingresados todos los
datos del primer test
   set(handles.edit5,'Enable','Inactive');
   set(handles.edit5,'Visible','Off');
   set(handles.edit7,'Enable','Inactive');
   set(handles.edit7,'Visible','Off');
   set(hObject,'Enable','Inactive');
   set(handles.pushbutton14,'Enable','Inactive');
   set(handles.pushbutton19,'Enable','Inactive');
   set(handles.edit8,'Enable','On');
   set(handles.edit8,'Visible','On');
   set(handles.edit9,'Enable','On');
   set(handles.edit9,'Visible','On');
   set(handles.pushbutton15,'Enable','On');
   set(handles.pushbutton20,'Enable','On');

   n=0;  % Reinicio variable de conteo de ingreso de datos


 end



function pushbutton20_Callback(hObject, eventdata, handles) %Iniciadores
del Test UMBmark CW

global bt;
fwrite(bt,3) %Enviamos dato mediante bluetooth



function pushbutton19_Callback(hObject, eventdata, handles) %Iniciadores
del Test UMBmark CCW

 global bt;
 fwrite(bt,2) %%Enviamos dato mediante bluetooth
```

```
function pushbutton15_Callback(hObject, eventdata, handles) %Boton
"Ingresar" del UMBmark CCW de la interfaz

grid on
global x_1_;
global y_1_;
global x_;
global y_;
global n_;
global vect_x_;
global vect_y_;
global X_Z_;
global Y_Z_;
global X_Z_prom_CCW;
global Y_Z_prom_CCW;
global X_Z_prom_CW;
global Y_Z_prom_CW;
global Raiz_CCW;
global Raiz_CW;
global Radio_C;
global Base_act
global Error_Base;
global Error_Diametro;
global cL;
global cR;
hold on


vect_x_=zeros(5);  %Defino tamaño de los vectores de zeros en X
vect_y_=zeros(5);  %Defino tamaño de los vectores de zeros en Y
x_1_=get(handles.edit8,'String');%Captura informacion de las entradas de
los Edit Text de X
y_1_=get(handles.edit9,'String'); %Captura informacion de las entradas de
los Edit Text de X


    if isempty( x_1_) || isempty(y_1_)

      msgbox('No deben existir campos vacios','Advertencia!')

    else

      n_=n_+1 %Contador para el almacenamiento de vectores
      x_=str2double(x_1_); %Convirtiendo variables de String a Double
      y_=str2double(y_1_); %Convirtiendo variables de String a Double
      vect_x_(n_)=x_; %Almacenando en el vector corrrespondiente
      vect_y_(n_)=y_; %Almacenando en el vector corrrespondiente
      X_Z_=X_Z_+x_; %Sumatoria de las coordenadas ingresadas
      Y_Z_=Y_Z_+y_; %Sumatoria de las coordenadas ingresadas
      axis([-35 35 -35 35]); %Redefinicion de los ejes
      plot(vect_x_, vect_y_,'k+'); %Grafico los puntos de las coordenadas
ingresadas
      hold on; %Mantengo las graficas visibles
```

```
    end

    if n_>4;

       X_Z_prom_CCW=X_Z_/5; %Obtengo el promedio hallado de sumatoria de
X
       Y_Z_prom_CCW=Y_Z_/5; %Obtengo el promedio hallado de sumatoria de
Y
       compass(X_Z_prom_CCW,Y_Z_prom_CCW,'k'); %Grafico vector
       Raiz_CCW=sqrt(((X_Z_prom_CCW)^2)+((Y_Z_prom_CCW)^2));%Hallando
raices de las dos componentes de las raices
       Error_Max=max(Raiz_CCW,Raiz_CW);%hallando valores maximos
      %alpha=((X_Z_prom_CW+X_Z_prom_CCW)*(180))/(-4*pi*2); %Hallando
angulo Alpha
      alpha=((Y_Z_prom_CW-Y_Z_prom_CCW)*(180))/(-4*pi*200);
       beta=((X_Z_prom_CW-X_Z_prom_CCW)*(180))/(-4*pi*200); %Hallando
angulo Beta
       Radio_C=(200/(2*(sin(beta/2)))); %Radio de curvatura (4.21)
       Error_Base=(90/(90-alpha)); %Hallando en error de la base nominal
       Base_act=Error_Base*12.25; %Hallando valor real de la base (4.26)
130 en mm
       Error_Diametro=(Radio_C+6.15)/(Radio_C-6.15); %Hallando error de
diametro (4.22).
       cL=2/(Error_Diametro+1); %factor de correccion llanta izquierda
       cR=2/((1/Error_Diametro)+1); %factor de correccion llanta derecha
       n_=0;  %Reinicio variable

      %Cambiamos las configuraciones una vez se han ingresados todos los
datos del segundo test

      msgbox('No es posible ingresar mas datos','Advertencia!')
      set(handles.edit8,'Enable','Inactive')
      set(handles.edit8,'Visible','Off')
      set(handles.edit9,'Enable','Inactive')
      set(handles.edit9,'Visible','Off')
      set(hObject,'Enable','Inactive')

      %Imprimo factores de correcciones los cuales van a ser ingresados
al
      %robot

      set(handles.text12,'string',cL)
      set(handles.text13,'string',cR)
      set(handles.text18,'string',Base_act)

 end
```

ANEXO D

ANEXO D

Interfaz Grafica -TRIANGULAR PATH TEST

```
function varargout = Triangular_Path_Test(varargin)
% TRIANGULAR_PATH_TEST MATLAB code for Triangular_Path_Test.fig
%      TRIANGULAR_PATH_TEST, by itself, creates a new
TRIANGULAR_PATH_TEST or raises the existing
%      singleton*.
%
%      H = TRIANGULAR_PATH_TEST returns the handle to a new
TRIANGULAR_PATH_TEST or the handle to
%      the existing singleton*.
%
%      TRIANGULAR_PATH_TEST('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TRIANGULAR_PATH_TEST.M with the given
input arguments.
%
%      TRIANGULAR_PATH_TEST('Property','Value',...) creates a new
TRIANGULAR_PATH_TEST or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Triangular_Path_Test_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Triangular_Path_Test_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Triangular_Path_Test

% Last Modified by GUIDE v2.5 15-Sep-2016 18:47:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Triangular_Path_Test_OpeningFcn,
...
                   'gui_OutputFcn',  @Triangular_Path_Test_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

function Triangular_Path_Test_OpeningFcn(hObject, eventdata, handles,
varargin)

global X_Z_L;
global Y_Z_L;
global n_L;
global vect_x_L
global vect_y_L;
global n_T;
global Y_Z_T;
global X_Z_T;


handles.output = hObject;

%Configuracion inicial

set(handles.edit7,'String','')
set(handles.edit8,'String','')
set(handles.edit5,'String','')
set(handles.edit6,'String','')
set(handles.text20,'String','')
set(handles.text21,'String','')
set(handles.text23,'String','')

set(handles.edit5,'Enable','Inactive')
set(handles.edit5,'Visible','Off')
set(handles.edit6,'Enable','Inactive')
set(handles.edit6,'Visible','Off')
set(handles.pushbutton5,'Enable','off')
set(handles.pushbutton6,'Enable','off')

axis([-40 40 -40 40])%Limite de los ejes

%Declaracion de variables iniciales

n_L=0;
vect_x_L=zeros(5);
vect_y_L=zeros(5);
Y_Z_L=0;
X_Z_L=0;
n_T=0;
X_Z_T=0;
```

```
Y_Z_T=0;

guidata(hObject, handles);


function varargout = Triangular_Path_Test_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;



function edit5_Callback(hObject, eventdata, handles)

function edit5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)


function edit6_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pushbutton5_Callback(hObject, eventdata, handles)

grid on

global x_1_T;
global y_1_T;
global x_T;
global y_T;
global n_T;
global vect_x_T;
global vect_y_T;
global X_Z_T;
global Y_Z_T;
global X_Z_prom_T;
global Y_Z_prom_T;
```

```
global alpha_T;
global Error_base_T;
global Base_Actual_T;
hold on


vect_x_T=zeros(5); %Defino tamaño de los vectores de zeros en X
vect_y_T=zeros(5); %Defino tamaño de los vectores de zeros en Y
x_1_T=get(handles.edit5,'String'); %Captura informacion de las entradas
de los Edit Text de X
y_1_T=get(handles.edit6,'String');  %Captura informacion de las entradas
de los Edit Text de Y


 if isempty( x_1_T) || isempty(y_1_T) %Verifica que no existan casillas
vacias al momento de ingresar los datos

    msgbox('No deben existir campos vacios','Advertencia!')%Error
Emergente si la condicion de campos vacios se cumple

 else %Inicio de almacenamiento de vectores

     n_T=n_T+1; %Contador para el almacenamiento de vectores
     x_T=str2double(x_1_T); %Convirtiendo variables de String a Double
     y_T=str2double(y_1_T); %Convirtiendo variables de String a Double
     vect_x_T(n_T)=x_T;  %Almacenando en el vector corrrespondiente
     vect_y_T(n_T)=y_T; %Almacenando en el vector corrrespondiente
     X_Z_T=X_Z_T+x_T; %Sumatoria de las coordenadas ingresadas
     Y_Z_T=Y_Z_T+y_T; %Sumatoria de las coordenadas ingresadas
     axis([-40 40 -40 40]); %Redefinicion de los ejes
     plot(vect_x_T,vect_y_T,'k+'); %Grafico los puntos de las coordenadas
ingresadas
     hold on %Sostengo la informacion suministrada

 end

 if n_T>4   %Cuando finaliza el alamcenamiento de vectores, ejecuta lo
calculos pertinentes

       X_Z_prom_T=X_Z_T/5%Obtengo el promedio hallado de sumatoria de X
       Y_Z_prom_T=Y_Z_T/5 %Obtengo el promedio hallado de sumatoria de Y
       compass(X_Z_prom_T,Y_Z_prom_T,'k'); %Grafico vector
       alpha_T=(X_Z_prom_T*180)/((-200)*pi) %Hallando angulo Alpha
       Error_base_T= 90/(90-alpha_T) %Hallando en error de la base
nominal
       Base_Actual_T=Error_base_T*125  %Hallando valor real de la base
(4.26) 130 en mm
       msgbox('No es posible ingresar mas datos','Advertencia!');
       n_T=0;
       set(handles.text23,'string',Base_Actual_T);

     set(handles.edit5,'Enable','Off')
```

```
     set(handles.edit5,'Visible','Off')
     set(handles.edit6,'Enable','Off')
     set(handles.edit6,'Visible','Off')
     set(handles.pushbutton5,'Enable','Off')
     set(handles.pushbutton6,'Enable','Off')

 end



function pushbutton6_Callback(hObject, eventdata, handles)
global bt;
 fwrite(bt,5)


function edit7_Callback(hObject, eventdata, handles)

function edit7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)

function edit8_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function pushbutton7_Callback(hObject, eventdata, handles)
grid on

global x_1_L;
global y_1_L;
global x_L;
global y_L;
global n_L;
global vect_x_L;
global vect_y_L;
global X_Z_L;
global Y_Z_L;
global X_Z_prom_L;
global Y_Z_prom_L;
global Beta_L;
```

```
global Radio_C_L;
global Error_Diametro_L;
global cL;
global cR;

 hold on


vect_x_L=zeros(5);
vect_y_L=zeros(5);
x_1_L=get(handles.edit7,'String');
y_1_L=get(handles.edit8,'String');


 if isempty( x_1_L) || isempty(y_1_L)

    msgbox('No deben existir campos vacios','Advertencia!')

 else %Inicio de almacenamiento de vectores

     n_L=n_L+1 %
     x_L=str2double(x_1_L);
     y_L=str2double(y_1_L);
     vect_x_L(n_L)=x_L;
     vect_y_L(n_L)=y_L;
     X_Z_L=X_Z_L+x_L;
     Y_Z_L=Y_Z_L+y_L;
     axis([-40 40 -40 40]);
     plot(vect_x_L,vect_y_L,'ro');
     hold on;

 end

 if n_L>4; %Cuando finaliza el alamcenamiento de vectores, ejecuta lo
calculos pertinentes

     X_Z_prom_L=X_Z_L/5; %Obtengo el promedio hallado de sumatoria de X
     Y_Z_prom_L=Y_Z_L/5; %Obtengo el promedio hallado de sumatoria de Y
     compass(X_Z_prom_L,Y_Z_prom_L,'r');  %Grafico vector
     Beta_L=(2*Y_Z_prom_L)/200
     Radio_C_L=(100/sin(Beta_L/2))
     Error_Diametro_L=(Radio_C_L-6.36)/(Radio_C_L+6.36)
     cL=2/(Error_Diametro_L+1); %factor llanta izquierda
     cR=2/((1/Error_Diametro_L)+1); %factor llanta izquierda



     msgbox('No es posible ingresar mas datos','Advertencia!')
     set(handles.edit7,'Enable','Inactive')
     set(handles.edit7,'Visible','Off')
     set(handles.edit8,'Enable','Inactive')
     set(handles.edit8,'Visible','Off')
     set(hObject,'Enable','Inactive')
```

```
    set(handles.pushbutton7,'Enable','off')
    set(handles.pushbutton8,'Enable','off')
    set(handles.edit5,'Enable','On')
    set(handles.edit5,'Visible','On')
    set(handles.edit6,'Enable','On')
    set(handles.edit6,'Visible','On')
    set(handles.pushbutton5,'Enable','On')
    set(handles.pushbutton6,'Enable','On')
    set(handles.text20,'string',cL)
    set(handles.text21,'string',cR)
  n_L=0;

 end


function pushbutton8_Callback(hObject, eventdata, handles)
global bt;
 fwrite(bt,4)
```