PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO DE GENERACIÓN DE TRAYECTORIAS PARA LA EVASIÓN DE UN OBSTÁCULO PARA UN ROBOT MÓVIL

Tesis para optar el Título de Ingeniero Electrónico, que presenta el bachiller:

Bryan Ciro Gutiérrez Suárez

ASESOR: Gustavo Kato Ishizawa

Lima, febrero del 2012

RESUMEN

La robótica móvil ha buscado desde sus inicios la autonomía móvil en su desplazamiento hacia sus objetivos. Conforme fue evolucionando esta área de la robótica se desarrollaron diversas metodologías para hacer más eficiente el movimiento autónomo de los robots. Gran cantidad de estas metodologías se desarrollaron para movilizar al robot en un entorno con obstáculos no estables. Se obtuvieron buenos resultados a costa de la alta complejidad de sus algoritmos, así como la gran cantidad de sensores implementados en el robot y entorno. La presente tesis busca reducir la complejidad de algoritmos para calcular las trayectorias más cortas hacia el objetivo de recorrido de un robot móvil en un entorno con obstáculos estables (entorno estructurado).

Para la solución del problema se elige trabajar con un robot móvil de tracción diferencial. Se elaboró un programa en lenguaje C++/CLI con una interfaz gráfica de usuario (IGU) para poder, en esta, detallar información sobre el entorno y la posición final del robot y, también, para habilitar la comunicación usuario-robot. Posteriormente en el algoritmo principal del programa se hacen cálculos matemáticos para determinar la trayectoria que es más corta a recorrer con las condiciones especificadas previamente en la IGU. Con la trayectoria obtenida se generan las señales de control que son enviadas al robot móvil para que recorra dicha trayectoria.

Se hacen pruebas de la generación de trayectorias y seguimiento de las mismas con obstáculo y sin obstáculo. En los resultados de las pruebas experimentales se presentaron errores de precisión y se mejoraron con un arreglo correctivo de velocidades de los motores del robot. Con este ajuste los resultados obtenidos fueron los esperados en relación con la correcta generación de la trayectoria así como el seguimiento de la misma.

Se concluye que las velocidades a las que se desempeñan los motores deben ser corregidas para disminuir el error de precisión en las pruebas experimentales del sistema. Finalmente, la precisión del sistema depende de la longitud y forma de las trayectorias a seguir.

ÍNDICE

INTRODUCCIÓN:

La robótica móvil empezó a tener su apogeo en la industria en los 80s [4], comenzando a realizar sus tareas mediante un operador con la finalidad de reemplazar la mano de obra humana en altos peligros de labor. Se introdujeron robots que realicen tareas para innumerables operaciones como, por ejemplo, las siguientes: mapeado de áreas contaminadas, limpiado de superficies, detectores de corrosión en superficies metálicas, transporte de material nocivo, etc.

Es entonces que se requiere que todos estos robots, aparte de realizar eficientemente su función principal, sean lo suficientemente autónomos e inteligentes para movilizarse en su campo de operación de manera segura y eficiente, esquivando obstáculos y haciendo recorridos cortos en sus tareas. Para esto, posteriormente se diseñaron diferentes sistemas para crear un robot autónomo y con inteligencia de resolver problemas de desplazamiento en áreas estructuradas y no estructuradas. Para la robótica móvil se han desarrollado múltiples metodologías de diseño de sistemas para la movilización eficaz y autónoma de los robots móviles para diversas aplicaciones.

La presente investigación propone una solución para la generación y el seguimiento de trayectorias en un entorno estructurado con obstáculos conocidos y predefinidos con la finalidad de generar la trayectoria más corta en recorrido hacia su objetivo. Se expone la teoría básica de la estructura y principios de movimiento de los robots móviles así como el diseño e implementación del algoritmo seguido para la solución del problema. Se realizan y se documentan las pruebas experimentales del sistema desarrollado en el presente asunto de estudio.

CAPÍTULO 1: LOS ROBOTS MÓVILES DE TRACCIÓN DIFERENCIAL

1.1 Los robots móviles y sus configuraciones de locomoción

La robótica ha sido ampliamente aplicada a las industrias debido a que presenta las siguientes ventajas: aumento de la productividad, flexibilidad, alta calidad y mejora de la seguridad. Los robots móviles aparecen, inicialmente, como exploradores y transportadores en donde la mano de obra humana corre algún riesgo. En un inicio estos robots eran tele-operados para llegar a los objetivos indicados pero con el desarrollo de la robótica se han logrado implementar sistemas de navegación para los robots móviles de manera autónoma [1].

Los robots móviles se clasifican por el medio de movilización y el tipo de locomoción utilizado. Para el presente asunto de estudio se ha optado por usar un robot móvil con ruedas para llevar a cabo la tarea de evasión de obstáculos. A continuación se presentan las configuraciones más aplicadas actualmente a los robots móviles terrestres con ruedas.

La configuración de triciclo [2] consiste en una sola rueda de direccionamiento del robot y dos ruedas posteriores en un mismo eje de tracción. Esta configuración es bastante simple en cuanto a la estructura mecánica y el control electrónico del mismo. Entre sus principales inconvenientes se presentan la poca estabilidad debido al posicionamiento de su centro de gravedad y la perdida de tracción de sus ruedas causada por el direccionamiento hecho por una sola rueda provocando así dificultades en los cálculos de posicionamiento del robot.

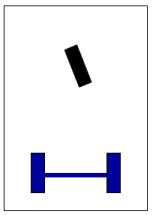

Figura 1.1: Configuración triciclo

La configuración de cuatriciclo o Ackerman [2] es cotidianamente usada en los automóviles. Esta configuración consiste de un sistema de direccionamiento Ackerman y un solo eje de tracción posterior. El sistema brinda una buena estabilidad al evitar el deslizamiento de las ruedas con lo cual los errores de odometría se reducen.

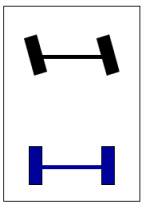

Figura 1.2: Configuración Ackerman

La configuración omnidireccional [2], es una de las más complejas pero a la vez la de mejor maniobrabilidad debido a que puede orientarse en cualquier dirección sin necesidad de pasar por un proceso de reorientación. Estos robots omnidireccionales por lo general se constituyen por ruedas suecas las cuales permiten el desplazamiento de la rueda en su plano perpendicular.

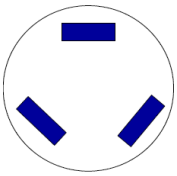

Figura 1.3: Configuración omnidireccional

La configuración diferencial se basa en la dirección y tracción del robot móvil realizado en un solo eje compuesto por dos ruedas independientes. Cada rueda posee su propio control de velocidad con lo cual se asegura el direccionamiento del robot. A continuación se explica de manera más detallada esta configuración.

1.2 El robot móvil de tracción diferencial

En el presente asunto de estudio se trabajará con un robot terrestre de tracción diferencial debido a su baja complejidad en implementación y que es comúnmente usado en la navegación de lugares con actividades humanas involucradas como almacenes u oficinas por su versatilidad.

Estos robots móviles están compuestos por un eje de tracción y dirección al mismo tiempo. Este eje está compuesto por dos motores independientes entre sí con ruedas acopladas; el direccionamiento del robot móvil se logra variando las velocidades de cada motor. Básicamente el robot puede realizar tres tipos de movimiento, los cuales son: en línea recta, en arco, y rotación (vuelta sobre su propio eje). Cada motor

conectado a cada rueda posee acoplado un codificador rotativo (encoder) para poder controlar la posición de estos robots mediante cálculos odométricos. Se entiende por odometría a la técnica que tiene por objetivo estimar la posición y orientación de un móvil a partir del número de vueltas dadas por sus ruedas (obtenidas por el número de cuentas encoder del motor).

La desventaja de esta configuración es que se necesita que las características de los motores sean completamente idénticos para que la odometría sea lo más precisa posible. Idealmente motores con especificaciones técnicas idénticas deberían tener un comportamiento idéntico pero en la práctica no es así. Es por esto que se deben hacer pruebas con cada motor para poder realizar los factores correctivos necesarios para asemejar los motores a lo ideal y reducir los errores odométricos. En la figura 1.4 se observa la distribución de las ruedas del robot móvil. La rueda denominada "ball caster" sirve como soporte de estabilidad mas no como direccionamiento.

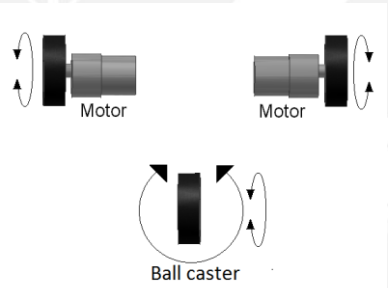

Figura 1.4: Configuración diferencial

1.3 Modelo cinemático del robot móvil de tracción diferencial

El modelo cinemático del robot móvil depende de las velocidades de cada rueda a las cuales denominaremos Vl y Vr como se observa en la figura 1.5. Debido a que el direccionamiento y velocidad del robot móvil está sujeto a las velocidades de sus ruedas se genera un momento angular con ω para cada velocidad constante de Vl y Vr. Este momento angular describe el radio de curvatura de arco que está siguiendo el robot en dicho instante. El modelo cinemático del robot móvil con respecto a las velocidades de sus ruedas se muestra en la figura 1.5 [2].

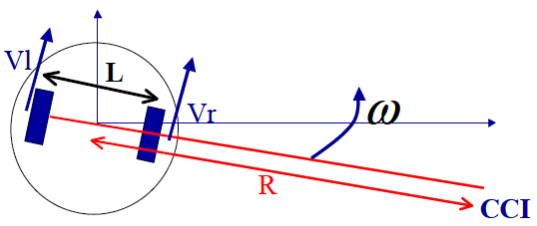

Figura 1.5: Modelo cinemático de la configuración diferencial

$$\omega = \frac{Vr - Vl}{L} \tag{1.1}$$

$$\omega \cdot \left(R - \frac{L}{2}\right) = Vr \tag{1.2}$$

$$\omega \cdot \left(R + \frac{L}{2}\right) = Vl \tag{1.3}$$

Se puede acotar de las ecuaciones anteriores que cuando Vl y Vr son diferentes entre sí el robot móvil sigue un arco de circunferencia (ecuaciones 1.2 y 1.3). Cuando Vl y Vr son idénticos el momento angular ω será igual a cero (ver ecuación 1.1), por consiguiente, el robot avanzará en línea recta. Cuando Vl= -Vr el robot girará sobre su propio eje en sentido antihorario.

Además es necesario saber el posicionamiento del robot en un instante determinado para lograr ejercer un control adecuado sobre este; por esto, se recurre a los cálculos odométricos en cada instante (frecuencia de muestreo). La odometría es una técnica bastante simple y reduce su error con altas frecuencias de muestreo. La debilidad de la odometría reside en que se basa en la acumulación temporal del movimiento lo cual en términos más simples inducen a acumular el error con la distancia recorrida. En la figura 1.6 se muestra el gráfico de cálculos para la odometría [3].

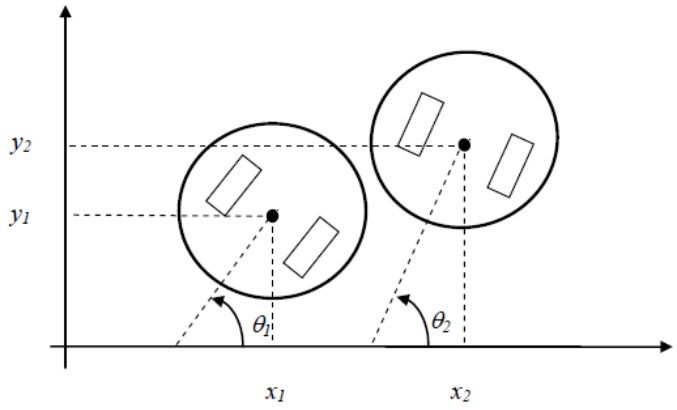

Figura 1.6: Gráfico de cálculos odométricos

$$x_2 = x_1 + \Delta x \quad (1.4)$$

$$y_2 = y_1 + \Delta y \quad (1.5)$$

$$\theta_2 = \theta_1 + \Delta\theta \quad (1.6)$$

$$\Delta x = \frac{\pi r(ContR + Cont\_L) \cdot \cos\theta_1}{E} \quad (1.7)$$

$$\Delta y = \frac{\pi r(Cont\_R + Cont\_L) \cdot \text{sen}\,\theta_1}{E} \quad (1.8)$$

$$\Delta\theta = \frac{2\pi r(Cont\_R - ContL)}{E.L} \quad (1.9)$$

Donde “r” es el radio de las ruedas en metros, E es la resolución de cuentas por revolución del encoder, L es la separación entre ruedas en metros, $(x_1; y_1)$ es el punto inicial conocido con orientación determinada por θ_1 y $(x_2; y_2)$ y θ_2 son del punto siguiente estimado, las muestras entre ambos puntos son tomadas con una frecuencia de muestro tal que cumpla con el criterio de Nyquist, es decir, debe ser mayor al doble de la frecuencia de cuentas (Cont_L y Cont_R) de los encoder. Estos cálculos se realizan frecuentemente durante todo el movimiento del móvil tan solo con la información sensorial recibida por los encoder y el conocimiento de posición y orientación del estado inicial del robot móvil.

CAPÍTULO 2: ALGORITMO DE GENERACIÓN DE TRAYECTORIAS PARA LA SOLUCIÓN DEL PROBLEMA

2.1. Estado del arte

2.1.1. Presentación del asunto de estudio

Desde la aparición de los robots móviles, sus diseños fueron básicamente para operaciones en las que el ser humano ponía en peligro su integridad física. Inicialmente fueron diseñados para ser guiados mediante distancias remotas y fueron catalogados como robots tele-operados. Por ejemplo, los robots usados por la NASA son controlados por un tele-operador vía radio en Marte [4].

En ciertas aplicaciones, conforme ha ido pasado el tiempo, se ha buscado que el movimiento del robot móvil sea autónomo y óptimo, con esto se trata de dejar de lado su dependencia con el tele-operador. Ejemplo de ello son los robots móviles en tiempo real y de trayectorias pre-programadas; ambos son autónomos y su selección depende la volatilidad del medio de operación a ser ubicados. En medios estructurados con obstáculos poco cambiantes los robots móviles resultan más eficientes debido a que pueden seguir las rutas más cortas según su programación. A diferencia de estos, los robots móviles en tiempo real son más eficientes en medios con obstáculos cambiantes.

La robótica móvil actual necesita de sistemas autónomos del movimiento de los robots [5], sistemas con los cuales el robot pueda ubicarse en un entorno y decidir una vía de movilización segura y eficiente. Para lograr esos objetivos se está usando sistemas inteligentes que utilizan redes neuronales artificiales. Se predice que en un futuro los automóviles tendrán incorporados estos sistemas con los cuales se facilitaría al conductor llegar al destino utilizando un sistema generador y seguidor de trayectorias en un espacio determinado eficientemente y sin colisiones.

Muchos de los sistemas inteligentes de generación y seguimiento de trayectorias frente a obstáculos de la robótica móvil en la actualidad están basados en el uso de las redes neuronales artificiales (RNA o ANN).

2.1.2. Estado de investigación

Existen diferentes metodologías aplicadas a los robots móviles autónomos. Una de ellas son las que programan sus trayectorias antes de actuar, es decir, que la generación de trayectorias se da antes de realizar cualquier movimiento. Este tipo de

metodología es muy útil en espacios donde los cuerpos obstáculos no son variables. Por otro lado existen los sistemas en tiempo real, los cuales están en constante actualización de los datos leídos por sus sensores.

Entre los sistemas en tiempo real están los sistemas inteligentes de navegación de robots móviles, los cuales están basados en el uso de redes neuronales artificiales (RNA) (figura 2.1). Las RNA funcionan con el principio de tratar de imitar en lo mayor posible a los sistemas nerviosos animales, es decir, consta de varias entradas (sensores), las cuales son transmitidas por una serie de ”n” capas neuronales hacia una sola salida o respuesta [6].

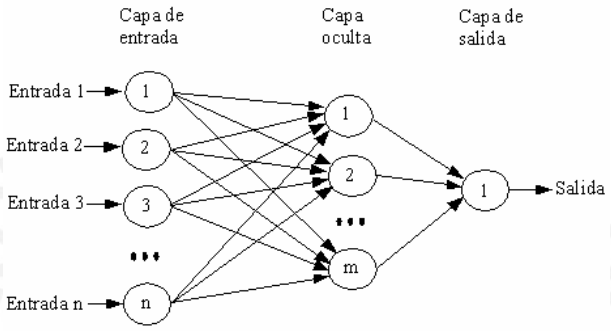

Figura 2.1: Esquema de una red neuronal artificial

El funcionamiento de estos robots consiste en la siguiente sucesión:

Sensar ➔ Mapear ➔ Generar un plan ➔ Actuar

Ejemplo de esto es el robot implementado en la Universidad Tecnológica de Pereira el cual está montado con sensores ultrasónicos (entradas), los cuales detectan la distancia hacia objetos próximos en ocho diferentes direcciones en un plano común (figura 2.2) [6].

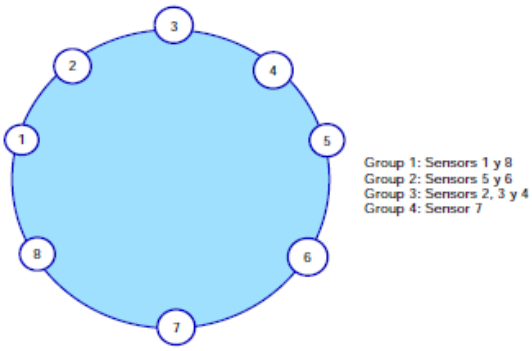

Figura 2.2: Distribución de sensores en el robot móvil

Las RNA le permitirán al robot móvil ubicarse en un espacio determinado tomando como entradas las señales que le envían los sensores mediante el conocimiento de la presencia de obstáculos y distancias a estos; esto le dará datos al robot para mapear y evaluar el entorno en el que se encuentra [7].

Una vez que el robot tiene un conocimiento del espacio en el que se ubica, procederá a realizar los cálculos para generar sus trayectorias en el espacio indicado. Este proceso se realiza constantemente con una frecuencia determinada. Finalmente con la dirección de movimiento elegida, el robot procederá a actuar y seguir la misma para llegar a acercarse a su objetivo [8].

De manera similar está el robot PIONEER-1 (el cual regenera su trayectoria cuando ocurren cambios en el entorno). Un ejemplo es el robot PIONEER-1 (figura 2.3) que está equipado con siete sensores ultrasónicos distanciados en 15 ° (figura 2.4) [5].

Figura 2.3: Robot PIONEER-1

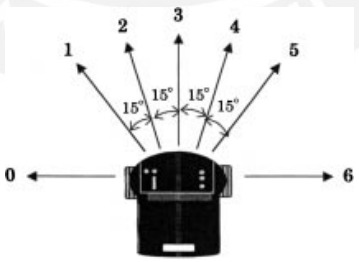

Figura 2.4: Localización de sensores ultrasónicos

El PIONEER-1 toma muestras de los sensores y grafica internamente un mapa local (figura 2.5)], el cual le brinda celdas libres y ocupadas en el mapa local. Como se puede observar en la figura 2.5, el mapa generado consta de pequeñas celdas –las celdas negras indican espacio ocupado y las blancas indican espacio vacío con esto

el robot consigue información en tiempo real de su entorno y busca una salida para llegar al punto objetivo [5].

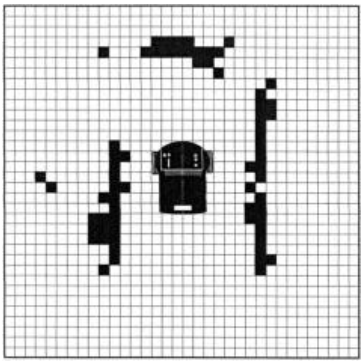

Figura 2.5: Generación de mapa local

Para la planificación y navegación de caminos en sistemas estructurados existe el algoritmo geométrico de los grafos de visibilidad, el cual tiene un conocimiento previo del entorno de trabajo. Los obstáculos están enmarcados por polígonos [10].

Mediante los grafos de visibilidad se puede obtener la interconexión entre los puntos de los obstáculos presentes en el mapa. Con esto al introducir el punto inicial y el final se genera una ramificación de puntos de visibilidad desde el punto inicial, pasando por los puntos de los polígonos hasta llegar al punto final. Así se obtiene una alta cantidad de caminos a seguir y mediante el algoritmo se determina el camino más corto a seguir. Recorre efectivamente el camino más corto hacia el final entre todos los caminos disponibles pero la desventaja de este método es que el recorrido se hace bordeando a los obstáculos presentes en el mapa lo cual disminuye el tiempo de recorrido.

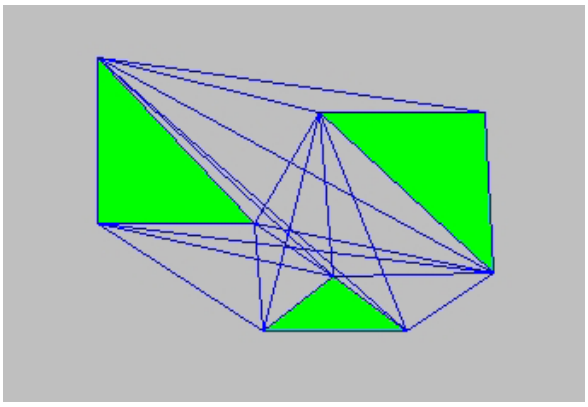

Figura 2.6: Grafo de visibilidad de tres obstáculos

2.1.3. Síntesis sobre el asunto estudiado

Actualmente se han implementado robots con ambos sistemas (tiempo real y programado) y, como se ha mencionado anteriormente, el sistema de generación de trayectorias de RNA muestra una alta eficiencia en superficies variantes. Sin embargo, los sistemas de programación de trayectorias ofrecen una solución más rápida y eficiente en distancia de recorrido en superficies estructuradas o invariantes.

El tema de estudio que se desarrollará será sobre un sistema de generación y navegación de trayectorias pre-programadas. Se entiende que el sistema a implementar contará con algoritmos de generación y seguimiento de trayectorias incorporados en una interfaz gráfica de usuario. La metodología de programación de trayectorias a priori abordada en el tema de estudio será eficiente para seguir distancias cortas en su trayectoria usando como principio los grafos de visibilidad y tratando de reducir el tiempo de recorrido al evitar el seguimiento del contorno de los obstáculos.

2.2. Modelo teórico

El sistema de generación de trayectorias pre-programadas requiere del uso de la interfaz gráfica de usuario (IGU) y de los algoritmos hechos en un lenguaje de programación; el lenguaje a usar en el asunto de estudio es el Visual C++/CLI. Se desarrollarán los algoritmos que incluirán el algoritmo generador de trayectorias que básicamente usa modelos matemáticos geométricos que serán necesarios para generar las trayectorias más eficientes del eventual recorrido del robot. De este primer algoritmo se obtendrá las señales de control con las cuales los motores del robot móvil podrán girar orientando al robot móvil hacia el objetivo.

La IGU será diseñada en Visual C++/CLI. Esta recibirá información del robot como la posición inicial y su posición final (esta es introducida por el usuario). Asimismo en la IGU el usuario tendrá la capacidad de ubicar gráficamente los obstáculos (representados por circunferencias de radio definible) que el robot deberá esquivar eventualmente. La generación de la trayectoria se realizará luego de la introducción de la posición final y obstáculos en la interfaz. Los algoritmos matemáticos calcularán la trayectoria y esta será mostrada en la IGU. El software programado generará las señales de control luego de la selección de la trayectoria generada a seguir. Las señales serán enviadas al microcontrolador del robot para que este pueda seguir la trayectoria indicada.

El robot móvil constará de tres ruedas: dos accionadas por motores DC, y una última posterior (ball caster) con la función de soporte. Para esta tarea se usará el micro-controlador ATMEGA8 de la familia de microcontroladores AVR8 de Atmel. Los datos generados por la IGU serán transmitidos mediante el puerto serial del computador directamente al microcontrolador del robot, para que finalmente este siga el recorrido. Las señales enviadas al robot constan básicamente de un arreglo de velocidades y distancias para cada motor. Con estos datos recibidos, el micro-controlador generará señales PWM (Pulse width modulation – Modulación por ancho de pulso) según el arreglo de velocidades así como también controlará las cuentas del encoder para controlar las distancias de desplazamiento. La PWM es una señal periódica con frecuencia predeterminada que tiene como objetivo controlar la potencia de dispositivos eléctricos mediante la variación de su ciclo de trabajo. En este caso concreto se usa para el control de velocidades de motores.

El microcontrolador enviará las señales PWM al Puente H (driver del sistema). El puente H está compuesto por un arreglo de transistores los cuales con señales lógicas pueden controlar el sentido de giro o accionamiento de motores. El puente H usado es dual (doble control) de la marca Toshiba.

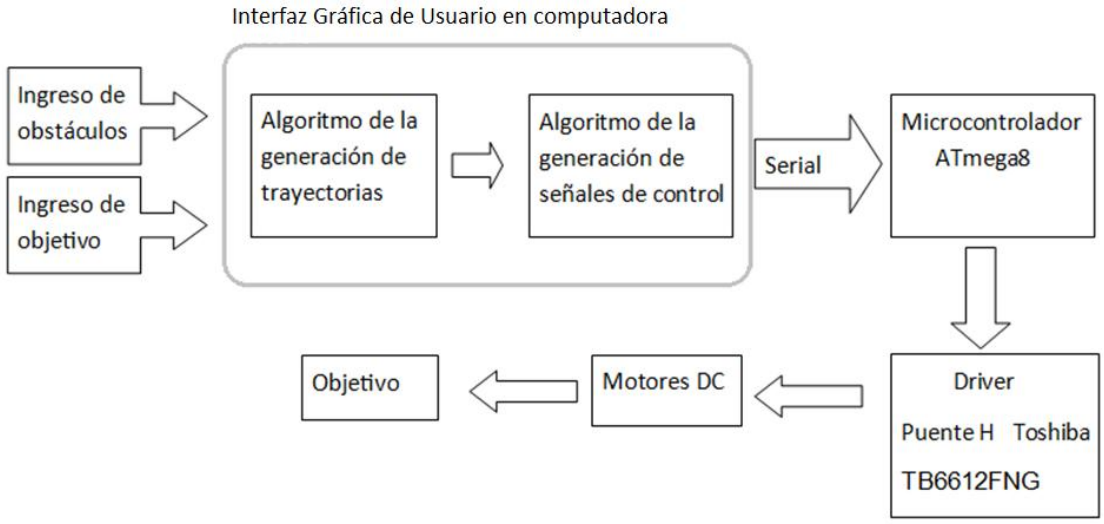

Figura 2.7: Diagrama de bloques del sistema

2.3. Algoritmo de generación de trayectorias

El algoritmo de generación de trayectorias implementado en Visual C++/CLI tiene como finalidad hallar la trayectoria más corta en distancia frente a los obstáculos para llegar al objetivo designado. Al finalizar el procedimiento del cálculo de trayectorias, este algoritmo debe mostrar por la IGU la gráfica de la trayectoria generada y, asimismo, almacenar los datos matemáticos de la trayectoria en un vector de programación, para convertirlos posteriormente a las señales de control entendibles por el microcontrolador del robot para el seguimiento de la ruta indicada.

El programa generará la trayectoria tomando como datos la posición inicial Pi (0;0), la posición final Pf (xf;yf) y los centros y radios de las circunferencias que serán considerados como obstáculos a esquivar por el algoritmo. Cabe resaltar que para la presente aplicación se asume que la dirección inicial que tiene el robot móvil es hacia el eje Y positivo del plano coordenado, es decir, el vector dirección inicial es de un valor vectorial unitario de (0;1).

El algoritmo generador de trayectorias consta de tres partes importantes, las cuales seguidas secuencialmente llegan a resolver el problema de la generación de trayectorias. Estas partes son las siguientes: a) detección de obstáculos y cálculo de puntos de paso, b) cálculo de los puntos adicionales de acomodamiento de curvas, c) el almacenamiento final de datos y gráfica de la trayectoria. A continuación se explica los tres pasos del algoritmo.

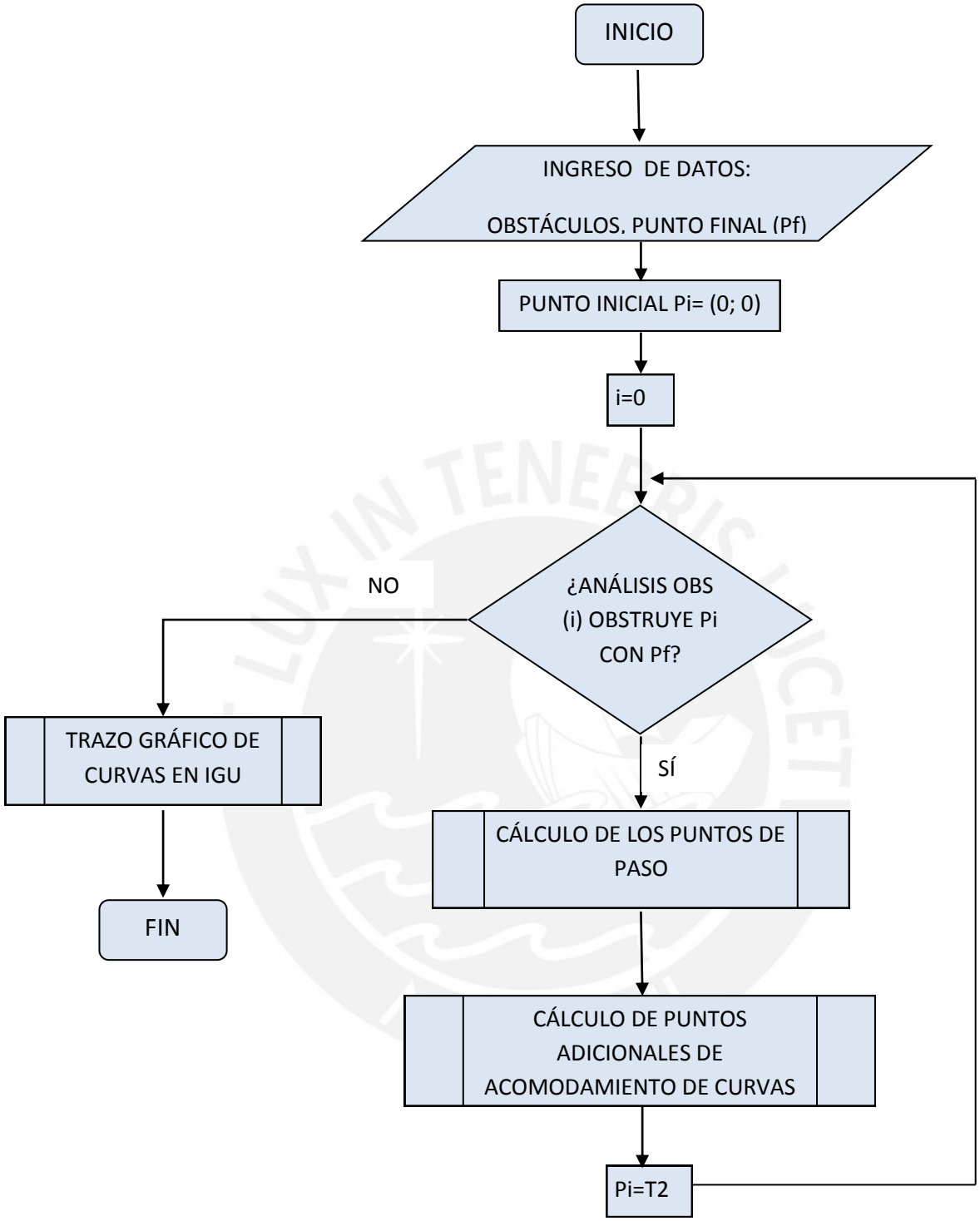

Figura 2.7: Diagrama de flujos del algoritmo generador de trayectorias

2.3.1 Detección de obstáculos y cálculo de los puntos de paso.

Una vez introducidos los parámetros a la IGU se procede a hacer el análisis de colisión o de detección de obstáculos. En primer lugar se necesita saber las dimensiones del robot móvil para hacer este cálculo. El robot móvil tiene forma circular con un diámetro L de 0.2 metros; este valor es importante para asegurar que los puntos de paso hallados posteriormente (por donde pasará el centro del robot) no estén dentro de una zona de peligro de colisión con el obstáculo. Para esto lo que se hace es representar el área de peligro de colisión mediante una nueva circunferencia de radio R que la englobe. El valor de R se denota en la siguiente expresión:

$$R = r + \frac{L}{2} + \sigma \tag{2.1}$$

Donde "r" es el radio del obstáculo, L es el diámetro del robot y σ es el espacio de tolerancia el cual se asume como 0.02 m. Posteriormente se procede a la detección de obstáculos. La figura 2.8 nos ilustra gráficamente de dónde se obtienen las siguientes expresiones para la detección de obstáculos.

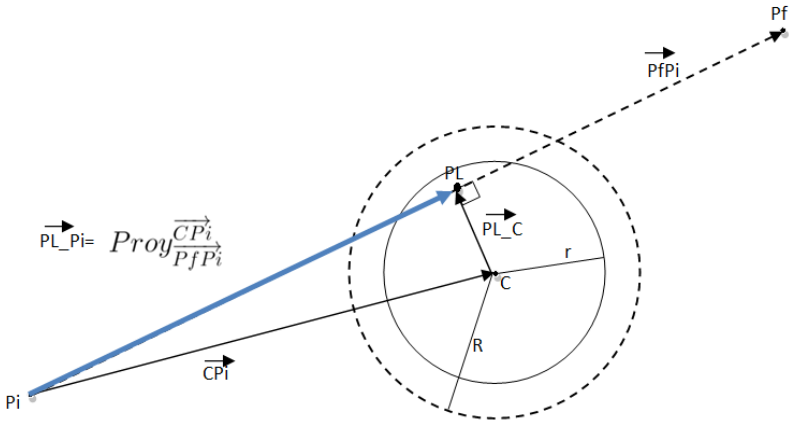

Figura 2.8: Gráfico para la detección de obstáculos

$$\overrightarrow{CPi} = C - Pi \tag{2.2}$$

$$\overrightarrow{PfPi} = Pf - Pi \tag{2.3}$$

$$\overrightarrow{PL_{Pi}} = Proy_{\overrightarrow{PfPi}}^{\overrightarrow{CPi}} \tag{2.4}$$

$$PL = Pi + \overrightarrow{PL_{Pi}} \tag{2.5}$$

$$\overrightarrow{PL_c} = PL - C \tag{2.6}$$

$$d = \left\|\overrightarrow{PL_C}\right\| \tag{2.7}$$

Donde Pi es el punto inicial, Pf es el punto final y “d” es la distancia entre el centro del obstáculo y el vector $\overrightarrow{PfPi}$. Si “d” es menor a R el programa automáticamente detecta obstáculo en el camino, en caso contrario este obstáculo no será tomado en consideración para el análisis de trayectorias.

Es necesario calcular los puntos de tangencia al obstáculo detectado para determinar los puntos de visibilidad por donde deberá pasar el robot móvil con la finalidad de hacer un recorrido más corto (en presencia de obstáculos, los puntos tangentes son considerados como los puntos de visibilidad). Habrá dos puntos posibles de tangencia, uno en dirección al vector $\overrightarrow{PL_c}$ y otro en dirección opuesta como se puede deducir de la figura 2.8. Para determinar qué punto de tangencia será el que asegure un camino aún más corto se utiliza la condicional $sin(\measuredangle(\overrightarrow{CP_i}; \overrightarrow{PL_c})) > 0$; si esta es afirmativa se calculará el punto de tangencia en dirección al vector $\overrightarrow{PL_c}$, en caso contrario se calculará en la dirección opuesta del mismo vector.

Una vez resuelta esta condicional se procede a calcular los puntos de tangencia T1 y T2 en la dirección designada. Se puede observar en la figura 2.9 un gráfico ilustrativo para el cálculo del punto de tangencia T1. Las siguientes expresiones son las necesarias para hallar el primer punto de tangencia.

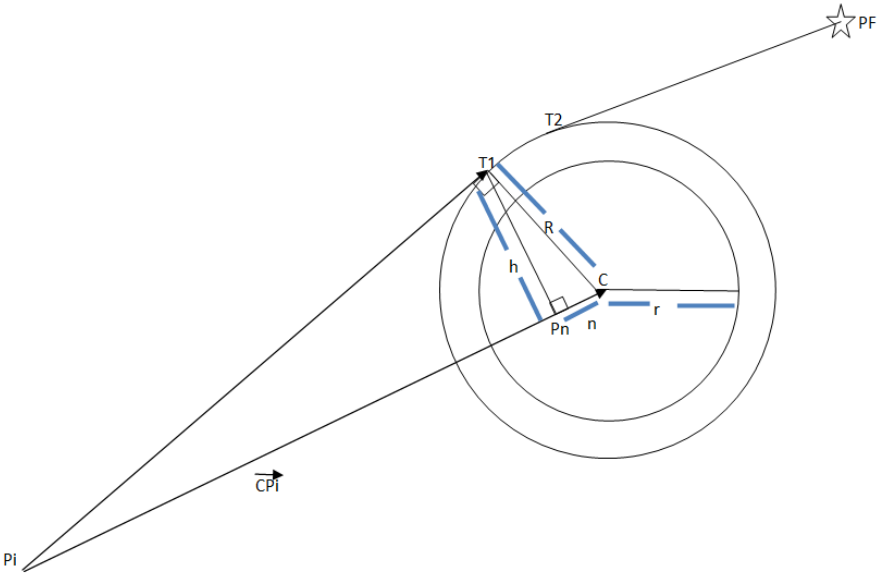

Figura 2.9: Gráfico para la ubicación del punto de tangencia T1

$$n = \left(R^2 \Big/ \left\|\overrightarrow{CPi}\right\|\right) \tag{2.8}$$

$$h = \sqrt{R^2 - n^2} \tag{2.9}$$

$$Pn = Pi + \overrightarrow{CPi} - n \cdot \vec{\mu}_{\overrightarrow{CPi}} \tag{2.10}$$

$$T1 = Pn \pm h.\vec{n}\{\vec{\mu}_{\overrightarrow{CPi}}\} \tag{2.11}$$

Donde $\vec{\mu}_{\overrightarrow{CPi}}$ es el vector unitario de $\overrightarrow{CPi}$ y $\vec{n}\{\vec{\mu}_{\overrightarrow{CPi}}\}$ es el vector perpendicular a $\vec{\mu}_{\overrightarrow{CPi}}$. Con las anteriores ecuaciones se logra hallar el punto de tangencia T1, el cual es el punto de tangencia entre el punto inicial Pi y la circunferencia de radio R.

El cálculo del punto T2 (punto tangencia entre la circunferencia y el punto final Pf) se hará de manera similar con la diferencia del cambio del vector $\overrightarrow{CPi}$ a $\overrightarrow{CPf}$ (vector entre el punto final Pf y el centro del obstáculo C). A continuación se muestra en la figura 2.10 un gráfico similar al anterior para explicar la obtención de las siguientes expresiones:

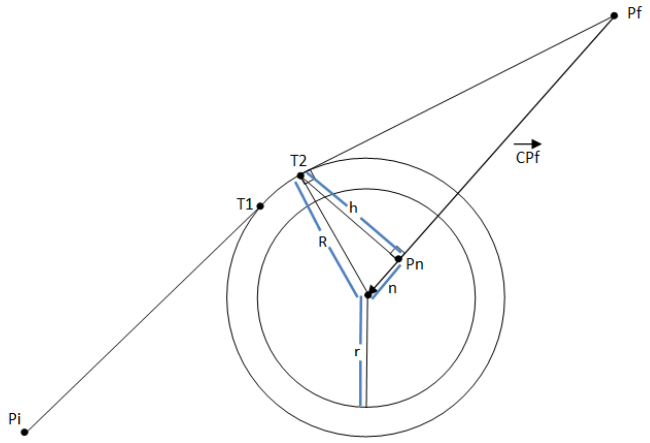

Figura 2.10: Gráfico para la ubicación del punto de tangencia T2

$$n = \left(R^2 \Big/ \left\|\overrightarrow{CPf}\right\|\right) \tag{2.12}$$

$$h = \sqrt{R^2 - n^2} \tag{2.13}$$

$$Pn = Pf + \overrightarrow{CPf} - n \cdot \vec{\mu}_{\overrightarrow{CPf}} \tag{2.14}$$

$$T2 = Pn \pm h.\vec{n}\left\{\vec{\mu}_{\overrightarrow{CPf}}\right\} \quad (2.15)$$

Una vez hallados los dos puntos de tangencia, T1 y T2, se han obtenido los puntos de paso para la evasión del obstáculo, los cuales serán registrados en un vector de programación del software como puntos de posición y vectores de dirección (estos vectores indican la dirección del robot móvil en su paso por dicho punto). Específicamente en este caso demostrativo se guarda los puntos de paso de la siguiente manera:

Tabla 2.1: Vector de almacenamiento de puntos de paso

Punto de posición	**Vectores dirección**
Pi	$\overrightarrow{Pi}$
T1	$\overrightarrow{T1}$
T2	$\overrightarrow{T2}$
Pf	$\overrightarrow{T3}$

Para la presente aplicación se ha escogido que el robot móvil tenga una dirección inicial hacia el eje Y positivo del plano coordenado con lo cual se sabe que $\overrightarrow{Pi}$ es igual a (0;1) tomando en consideración que los vectores dirección son unitarios. El vector unitario $\overrightarrow{T1}$ tendrá la dirección del vector $\overrightarrow{T1\_Pi}$ = T1-Pi y el vector $\overrightarrow{T2}$ tendrá la dirección del vector $\overrightarrow{PfT2}$ = Pf-T2. Finalmente el vector $\overrightarrow{T3}$ será igual al vector $\overrightarrow{T2}$ en el presente ejemplo analizado.

2.3.2 Cálculo de los puntos adicionales de acomodamiento de curvas

En la fase anterior del algoritmo de generación de trayectorias se han guardado una cantidad determinada de puntos de paso con sus respectivos vectores de dirección. Para algunos casos, que se detallarán a continuación, es necesario calcular puntos adicionales (los cuales serán insertados en el vector de programación de puntos de paso) para asegurar que el tramo de un punto a su consiguiente sea suavizado y asegure que se sigan los vectores dirección en ambos puntos del tramo. Para saber cuándo será necesario el cálculo de estos puntos se hace un análisis por cada dos

puntos consecutivos del vector de puntos de paso de la anterior fase. Se tienen los siguientes casos:

Caso 1: $\sin(\measuredangle(\overrightarrow{P1};\overrightarrow{P2P1})) < 0 \wedge \sin(\measuredangle(\overrightarrow{P2};\overrightarrow{P2P1})) < 0$

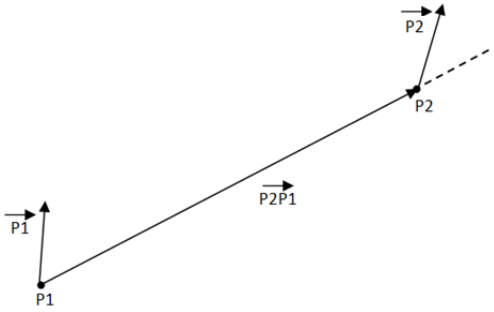

Figura 2.11: Caso Nº 1 del análisis de puntos consecutivos

Caso2: $\sin\left(\measuredangle(\overrightarrow{P1};\overrightarrow{P2P1})\right) > 0 \wedge \sin\left(\measuredangle(\overrightarrow{P2};\overrightarrow{P2P1})\right) > 0$

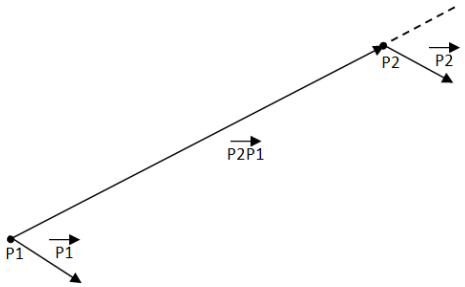

Figura 2.12: Caso Nº 2 del análisis de puntos consecutivos

Al cumplirse las anteriores dos condicionales (caso 1 y caso 2) entre dos puntos consecutivos de los puntos de paso previamente hallados se procede a calcular cual será el punto de inflexión (punto adicional) que une a los dos arcos de circunferencia (uno cóncavo y el otro convexo, no necesariamente en ese orden) los cuales se unen los dos puntos tratados inicialmente cumpliendo sus requerimientos de posición y dirección. En la figura 2.13 se explica gráficamente el procedimiento para hallar el punto de inflexión de los anteriores dos casos. Se utilizan las siguientes expresiones:

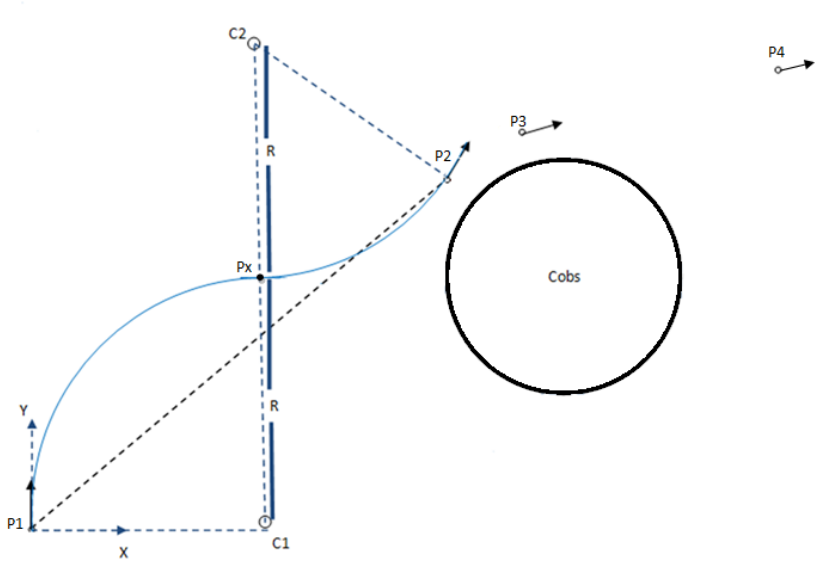

Figura 2.13: Gráfico para la ubicación del punto de inflexión Px

$$C1 = P1 \mp R \cdot \vec{n}_{\overrightarrow{P1}} \tag{2.16}$$

$$C2 = P2 \pm R \cdot \vec{n}_{\overrightarrow{P2}} \tag{2.17}$$

$$\|C2 - C1\| = \left\|\overrightarrow{C2C1}\right\| = 2R \tag{2.18}$$

Luego con esta anterior igualdad se deduce que R es:

$$R = \left| \frac{\sqrt{-a^2(d^2-4) + 2abcd - c^2(b^2-4)} - ab - cd}{b^2 + d^2 - 4} \right| \tag{2.19}$$

Donde,

$a = (P1_x - P2_x)$

$b = \mp\vec{n}\{\overrightarrow{\mu_{P1}}\}_x \mp \vec{n}\{\overrightarrow{\mu_{P2}}\}_x$

$c = (P1_y - P2_y)$

$d = \mp\vec{n}\{\overrightarrow{\mu_{P1}}\}_y \mp \vec{n}\{\overrightarrow{\mu_{P2}}\}_y$

En las anteriores expresiones símbolo de doble signo es el superior en el caso 1 o el inferior en el caso 2, en estas ecuaciones se ha considerado para simplificar el cálculo, que los arcos de circunferencia con centro C1 y C2 que unirán los puntos P1 y P2 serán del mismo radio R.

Una vez hallado el punto de inflexión se procede a insertarlo en el vector de puntos de paso de la fase anterior, posteriormente el vector de puntos de paso del tipo de la tabla 2.1 será modificado en un vector de tramos de la trayectoria. Este nuevo vector tiene un número indeterminado de tramos los cuales consisten de un arco de circunferencia o una recta que une dos puntos. Los parámetros de este nuevo tipo de vector son: punto inicial, punto final, dirección inicial, dirección final, radio de arco de curvatura de tramo y centro del arco de curvatura del tramo. En la tabla 2.2 se ilustra la composición de este nuevo vector suponiendo una trayectoria con cuatro puntos de paso P1, P2, P3 y P4 y sus respectivos radios y centros.

Tabla 2.2: Vector de almacenamiento de datos por tramos de trayectoria con tres tramos

Nº tramo	Punto inicial(x,y)	Punto final(x,y)	Dirección inicial(x,y)	Dirección final(x,y)	Radio de tramo	Centro de tramo(x,y)
1	P1	P2	$\overrightarrow{P1}$	$\overrightarrow{P2}$	R1	C1
2	P2	P3	$\overrightarrow{P2}$	$\overrightarrow{P3}$	R2	C2
3	P3	P4	$\overrightarrow{P3}$	$\overrightarrow{P4}$	R3	C3

Caso 3: $\sin\left(\measuredangle\left(\overrightarrow{P1};\overrightarrow{P2P1}\right)\right) < 0 \wedge \sin\left(\measuredangle\left(\overrightarrow{P2};\overrightarrow{P2P1}\right)\right) > 0$

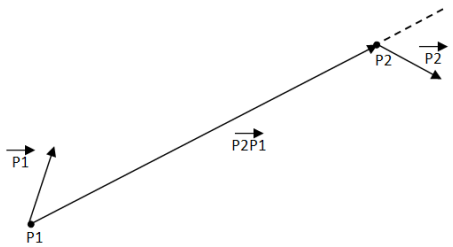

Figura 2.14: Caso Nº 3 del análisis de puntos consecutivos

Caso 4: $\sin\left(\measuredangle\left(\overrightarrow{P1};\overrightarrow{P2P1}\right)\right) > 0 \wedge \sin(\measuredangle(\overrightarrow{P2};\overrightarrow{P2P1})) < 0$

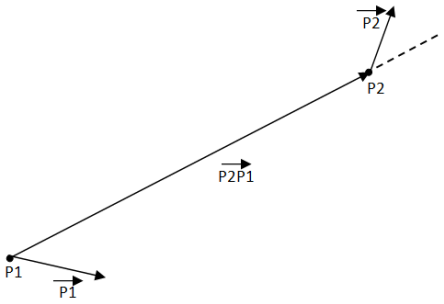

Figura 2.15: Caso Nº 4 del análisis de puntos consecutivos

En los casos 3 y 4 así como en los dos primeros casos se procede a calcular un punto adicional que asegure el suavizado de las curvas entre los dos puntos. En estos dos casos las curvas que unirían a los puntos P1 y P2 son un arco de circunferencia más una línea recta. En la figura 2.16 se observa un ejemplo del caso 4 del cual se pueden deducir las expresiones para el cálculo del punto adicional en ambos casos.

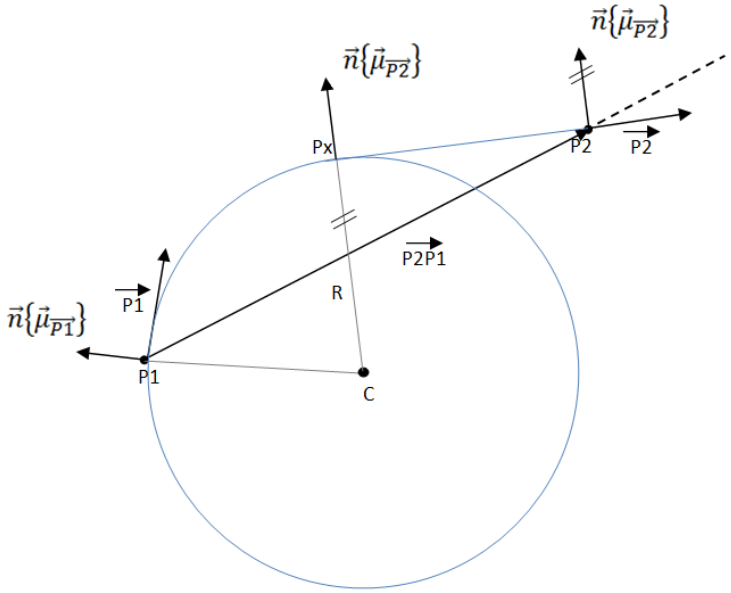

Figura 2.16: Gráfico para la ubicación del punto adicional Px

$$C = P1 - R \cdot \vec{n}\{\vec{\mu}_{\overrightarrow{P1}}\} \tag{2.20}$$

$$\overrightarrow{CP2} = C - P2 \tag{2.21}$$

$$R = \overrightarrow{CP2} \cdot \vec{n}\{\vec{\mu}_{\overrightarrow{P2}}\} \tag{2.22}$$

$$R = (P1 - R \cdot \vec{n}\{\vec{\mu}_{\overrightarrow{P1}}\} - P2) \cdot (\vec{n}\{\vec{\mu}_{\overrightarrow{P2}}\}) \quad (2.23)$$

$$Px = C + R \cdot \vec{n}\{\vec{\mu}_{\overrightarrow{P2}}\} \quad (2.24)$$

Con las anteriores expresiones se halla en primer lugar los valores de R y C para posteriormente hallar Px. Es bastante frecuente, sobretodo en este proyecto, que Px coincida con P2 con lo cual no habrá un punto adicional ni línea recta de tramo, sino un solo arco de circunferencia de radio R y centro C que una ambos puntos.

Caso 5: $\sin\left(\measuredangle\left(\overrightarrow{P1}; \overrightarrow{P2P1}\right)\right) = 0 \wedge \sin\left(\measuredangle\left(\overrightarrow{P2}; \overrightarrow{P2P1}\right)\right) = 0$

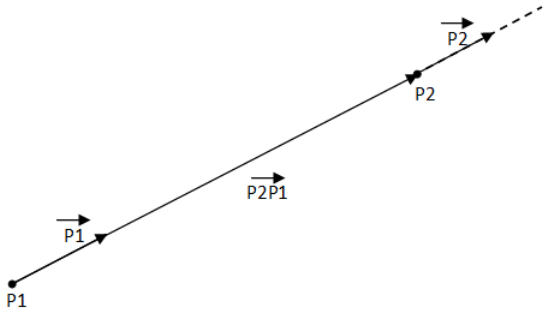

Figura 2.17: Caso Nº 5 del análisis de puntos consecutivos

En el caso 5 se presenta el caso más simple de todos puesto a que no hay que realizar cálculos de puntos adicionales ni cálculos de si centro de arco de recorrido debido a que en este caso el tramo es simplemente una línea recta entre ambos puntos. Se procede a guardar estos puntos, vectores dirección de los mismos, radio y centro del arco de recorrido como en los anteriores casos en el nuevo vector de tramos. Debido a que este caso es excepción a su ausencia de arcos de circunferencia, se le asignará valores al radio y al centro de circunferencia con "–1" para posteriormente ser identificado como línea recta por el programa.

2.3.3 Trazo de las curvas

Esta fase del algoritmo es solamente gráfico debido a que el último vector de tramos de la trayectoria (que será usado por el eventual algoritmo generador de señales de control) ya se guardó en la segunda fase. Se toma cada tramo de este vector y es enviado a una función en el VisualC++/CLI, la cual contiene el método de la clase Graphics, método DrawArc, el cual recibe los parámetros de cada tramo para dibujarlos y formar la trayectoria gráfica. En caso de recibir parámetro de centro y radio iguales a -1 (caso de la línea recta) se utiliza el método DrawLine para el dibujado línea.

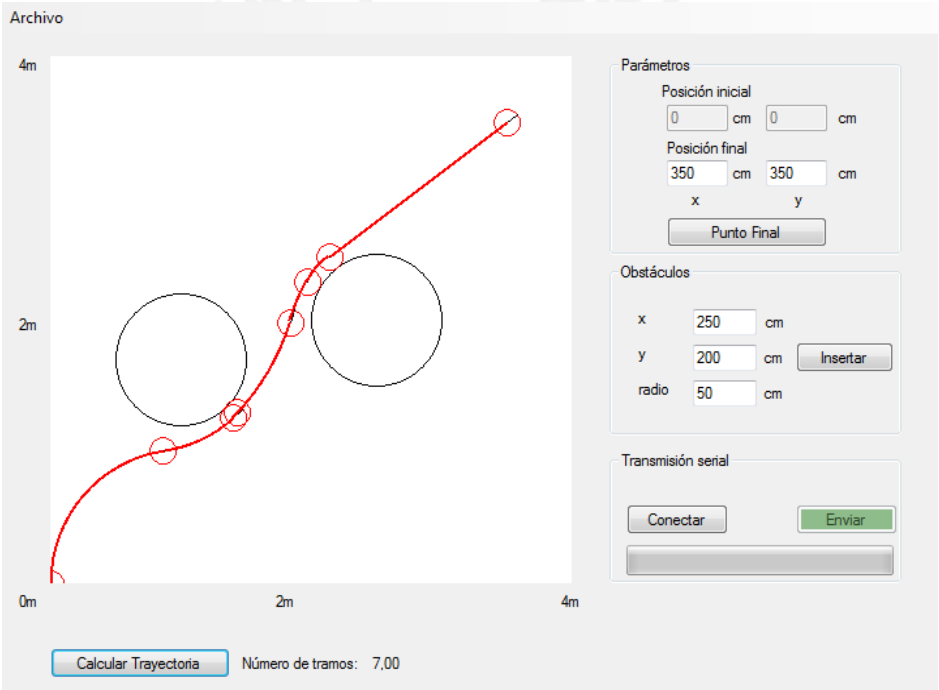

Figura 2.18: Funcionamiento de la IGU para la generación de trayectorias

CAPÍTULO 3: GENERACIÓN DE LAS SEÑALES DE CONTROL

En el anterior capítulo se trató el cálculo de trayectorias donde se obtiene los datos matemáticos de cada tramo de la trayectoria generada en un vector de programación. Este vector será de alta importancia para la generación de las señales de control al micro-controlador del robot, debido a que el vector contiene todos los datos de trayectoria que el robot móvil deberá seguir.

Se tienen los datos de cada tramo de la trayectoria, puntos de posición y vectores dirección iniciales y finales, así como radio de curvatura y centro de la curvatura. Con todos estos datos se pueden obtener las distancias y velocidades de recorrido que cada rueda debe recorrer por cada tramo para poder seguir la trayectoria generada de manera correcta. El algoritmo generador de señales de control consta de tres partes con las cuales se asegurarán la correcta recepción de las señales con robot móvil. Las partes son las siguientes:

a) Obtención de distancias de recorrido de las ruedas por tramo.

b) Obtención de velocidades de las ruedas por tramo.

c) Comunicación serial computadora-robot.

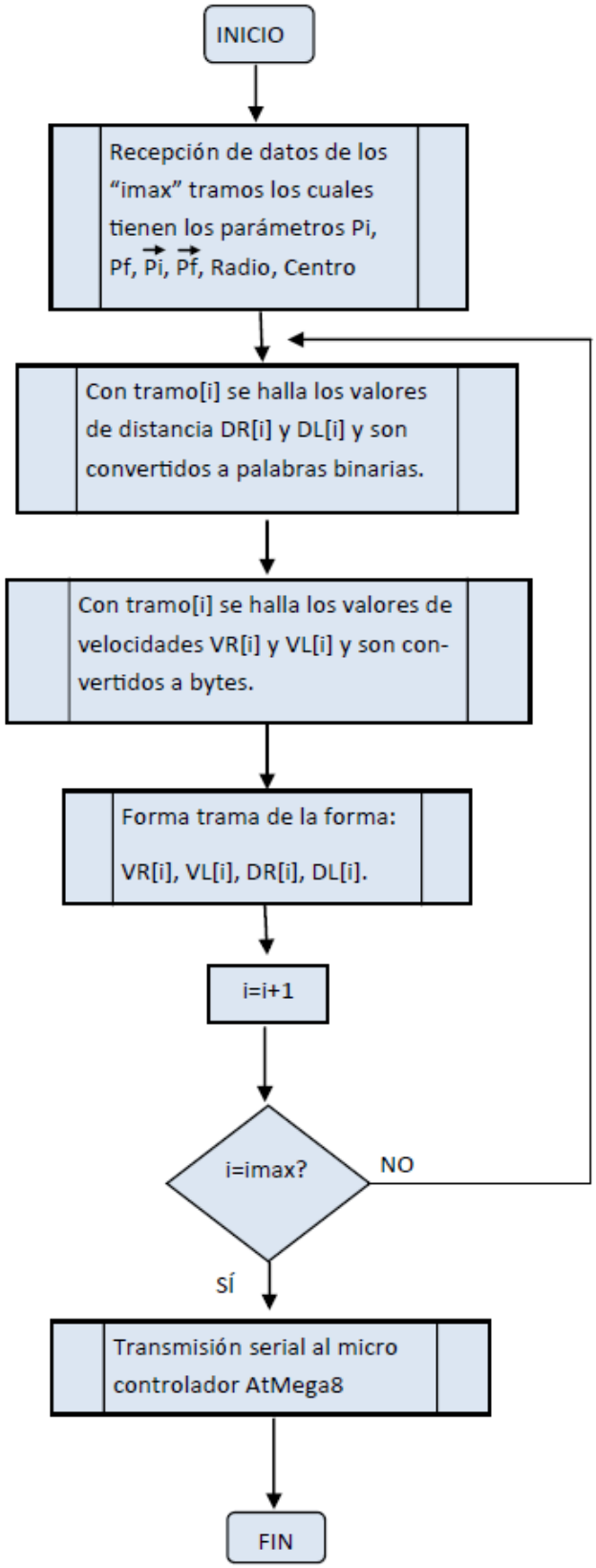

Figura 3.1: Diagrama de flujo de la generación de señales de control

3.1 Obtención de distancias de recorrido de las ruedas por tramo

Del algoritmo de generación de trayectorias se ha obtenido el vector con los datos de todos los tramos que forman la trayectoria a seguir. De aquí se deben de obtener las distancias que deberán recorrer las ruedas izquierda y derecha por cada tramo (o arco de circunferencia - línea) para asegurar que se reciban las distancias que lleven al objetivo. Para obtener las distancias que serán recorridas por el robot, primero se debe saber las dimensiones exactas del robot móvil.

3.1.1 El robot móvil y sus dimensiones

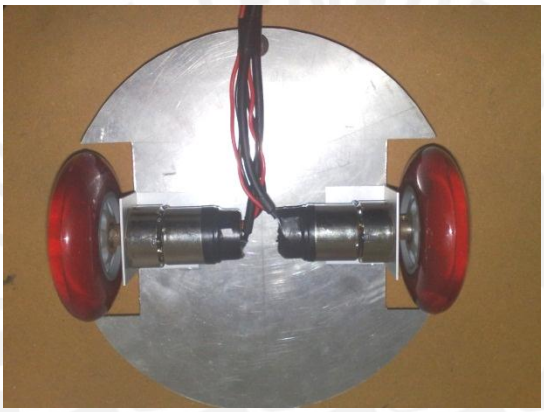

Figura 3.2: Robot móvil en vista superior

La figura 3.2 brinda una imagen clara del diseño del robot móvil. La distancia entre ambas ruedas L es 0.2 metros. Las ruedas usadas son unas ruedas de silicona comúnmente usadas en scooters y patines cotidianos, este material proporciona buena fricción con el suelo con lo cual se evitan los posibles deslizamientos de ruedas. El radio de estas ruedas R es 0.05 metros. La ecuación 3.1 muestra la relación entre la distancia recorrida S en metros con el número de cuentas del encoder C. [9]

$$S = 2\pi R \cdot \left(\frac{C}{E}\right) \tag{3.1}$$

Donde R es el radio de la rueda y E es la constante 624 (resolución del encoder por vuelta del eje)

3.1.2 Obtención de palabras de representación de distancias por tramos

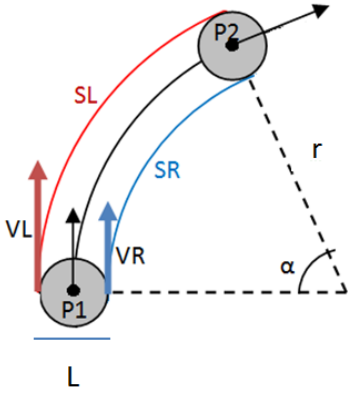

Figura 3.3: Gráfico para análisis de dirección de giro por tramo

De la figura 3.3 se puede obtener, por cada tramo, SL y SR, que son las distancias de la rueda izquierda y derecha respectivamente. La obtención de distancias (en metros) de ambas ruedas se deduce de las expresiones:

$$SL = \left(r \pm \frac{L}{2}\right) \cdot \alpha \tag{3.2}$$

$$SR = \left(r \mp \frac{L}{2}\right) \cdot \alpha \tag{3.3}$$

De donde "r" es el radio del arco de circunferencia del tramo, L es la distancia entre las ruedas y α es el ángulo de recorrido del arco de tramo. El signo superior se tomará si el recorrido es en dirección horaria o será el inferior en caso contrario. Del vector de tramos obtenido del algoritmo anterior no se puede obtener directamente α, este valor se deduce de dos datos de este vector los cuales son los vectores unitarios dirección en los puntos P1 y P2 ($\overrightarrow{P1}$ y $\overrightarrow{P2}$). La siguiente ecuación muestra la obtención del valor de α a partir del producto escalar de las normales de $\overrightarrow{P1}$ y $\overrightarrow{P2}$.

$$\alpha = \cos^{-1}\left(\overrightarrow{n_{P1}} \cdot \overrightarrow{n_{P2}}\right) \tag{3.4}$$

Ya obtenido el valor de α se obtiene los valores SL y SR de las ecuaciones 3.2 y 3.3 respectivamente. De los valores SR y SL (cuyas unidades son metros) se obtendrá los valores de C (ecuación 3.1), denotados por DistL y DistL(número de cuentas C obtenidas de la ecuación 3.1) respectivamente a las ruedas izquierda y derecha. DistL y DistL serán representados posteriormente en codificación binaria para ser enviados al robot móvil como parámetros distancia. De la ecuación 3.1, debido a la resolución del encoder representar cada distancia con un byte (256 valores o cuentas) daría una distancia máxima medible de 12.88 centímetros. Debido a que la aplicación funcionará

dentro de un campo cuadrado cartesiano de tres metros se decide representar a cada distancia con palabras de datos (16 bits) con las cuales se obtienen distancias máximas medibles de 32.99 metros, las cuales son útiles en el asunto de estudio.

3.2 Obtención de velocidades de las ruedas por tramos

Para que el robot móvil se desplace en cada tramo se necesita conocer las velocidades con las que cada rueda debe girar en cada tramo para asegurar que se siga la trayectoria designada. Para esto es necesario conocer el comportamiento de cada uno de los motores. La tabla 3.1 muestra las características de los motores DC utilizados

Tabla 3.1: Características del motor HG37D670WE12 - 052FH

Voltaje Nominal	7.2 V
Velocidad sin carga	160rpm
Reducción	1:52
Torque	100 oz-in (7.2 kg-cm)
Cables de motor	12" 20AWG
Cables de encoder de cuadratura	12" 20AWG (+5V, Gnd, Canal A, Canal B)
Resolución	624 Pulsos por revolución del eje
Diámetro del eje	6mm
Corriente sin carga	300mA
Máxima corriente (rotor bloqueado)	2.0A

3.2.1 Pruebas de velocidades de los motores DC sin carga y con carga

Las pruebas PWM de ambos motores frente a 256 valores diferentes de ciclo de trabajo para obtener la gráfica de velocidades de ambos motores como se aprecia en las figuras 3.4 y 3.5.

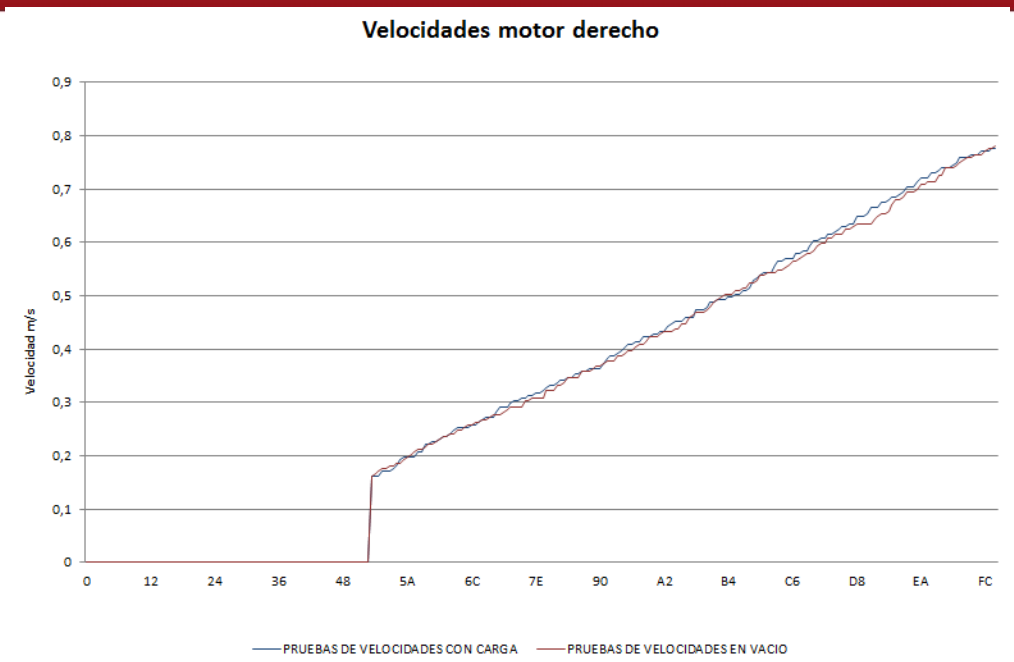

Figura 3.4: Gráfica Ciclo de trabajo vs. Velocidad de la rueda derecha

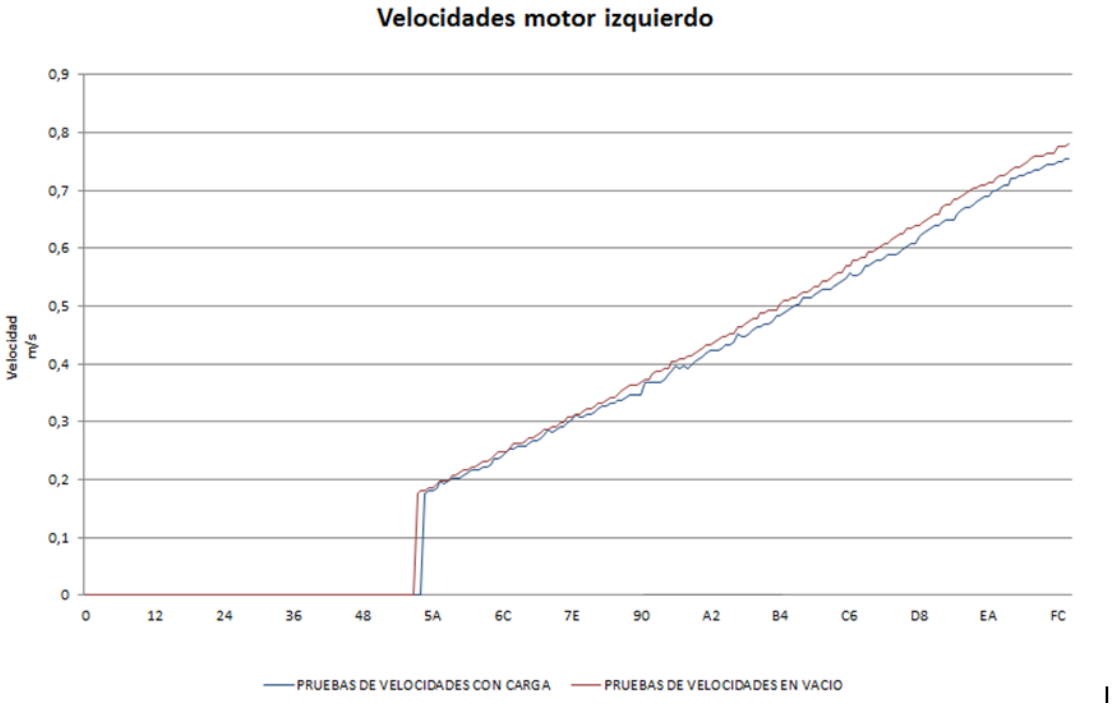

Figura 3.5: Gráfica Ciclo de trabajo vs. Velocidad de la rueda izquierda

De las pruebas de PWM, en vacío (sin carga) y con carga de ambos motores, realizadas se denota que la velocidad máxima que ambos motores pueden realizar es de 0.75m/s (simultáneamente) y la mínima es 0.18m/s. Debido a que las velocidades

más altas aumentan los errores de precisión, para la presente aplicación se establece que la velocidad máxima que cada rueda pueda desarrollar sea de 0.35 m/s.

3.2.2 Obtención de los bytes de representación de velocidades por tramos

Para obtener la velocidad que debe tener cada rueda por tramo se hace un análisis de sentido de giro (horario u antihorario). Según sea el resultado de este análisis se determina qué rueda lleva la velocidad más alta y con esto, mediante las ecuaciones 1.2 ó 1.3, se hallará de manera simple la velocidad de la rueda opuesta. A continuación el análisis de sentido de giro basado en la figura 3.3.

$$\sin(\measuredangle\overrightarrow{P1};\overrightarrow{P2}) > 0 \ \rightarrow VR = Vmax$$

$$\sin\left(\measuredangle\overrightarrow{P1};\overrightarrow{P2}\right) < 0 \ \rightarrow VL = Vmax$$

Donde Vmax es 0.35m/s. Los valores obtenidos de VR y VL tienen unidades en metros por segundo (m/s) y serán comparados con sus valores de ciclo de trabajo (en las gráficas de análisis de PWM) para obtener su respectivo valor binario entre 256 valores de las tablas de donde provienen estas gráficas.

3.3 Transmisión serial

Este paso es primordial para la comunicación entre la computadora y el robot. En los pasos anteriores se digitalizaron los valores de distancias y velocidades por tramos con los cuales la transmisión serial los enviará al robot.

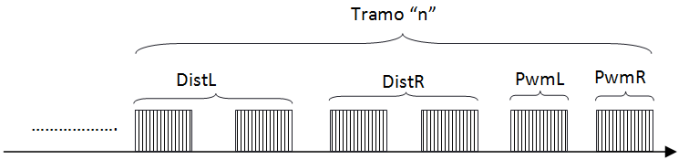

Figura 3.6: Trama enviada por comunicación serial

Apreciable en la figura 3.6 se tiene la trama a enviar por el puerto serial. La trama está subdividida en tramos (donde se almacenan los valores de velocidades y distancias de ambas ruedas por tramos de la trayectoria) los cuales serán recibidos por el robot

móvil y una vez terminada la transmisión de todos los datos de la trayectoria, el robot iniciará su movimiento. La conexión serial se configuró a 4800 baudios, 8bits de datos, 1 de parada, 1 bit de inicio y sin bit paridad.

CAPÍTULO 4: PRUEBAS DEL ALGORITMO Y RESULTADOS EXPERIMENTALES

4.1 Programación del robot móvil

En esta sección se documentan las pruebas realizadas al sistema del asunto de estudio. Antes de analizar los resultados experimentales se explica a continuación la arquitectura del robot móvil en programación.

El micro-controlador del robot móvil fue programado en lenguaje ensamblador usando el software VMLab. El programa se descargó al micro-controlador desde la computadora usando el grabador WinAVR con el AVRdude con la interfaz de usuario AVR8 Burn o Mat usando un módulo de programación USBasp.

El modo de trabajo del robot inicialmente es de recepción de datos. Está a la espera de datos entendibles por él y una vez que se reciben los datos por completo empieza su marcha hacia el objetivo. Esto mediante La aplicación de las señales PWM VL y VR recibidas y la constante comparación de distancias recorridas Cont_L y Cont_R con distancias por recorrer DL y DR en cada tramo de la trayectoria que se debe seguir. En la figura 4.1 se presenta el diagrama de flujos de la programación del micro-controlador.

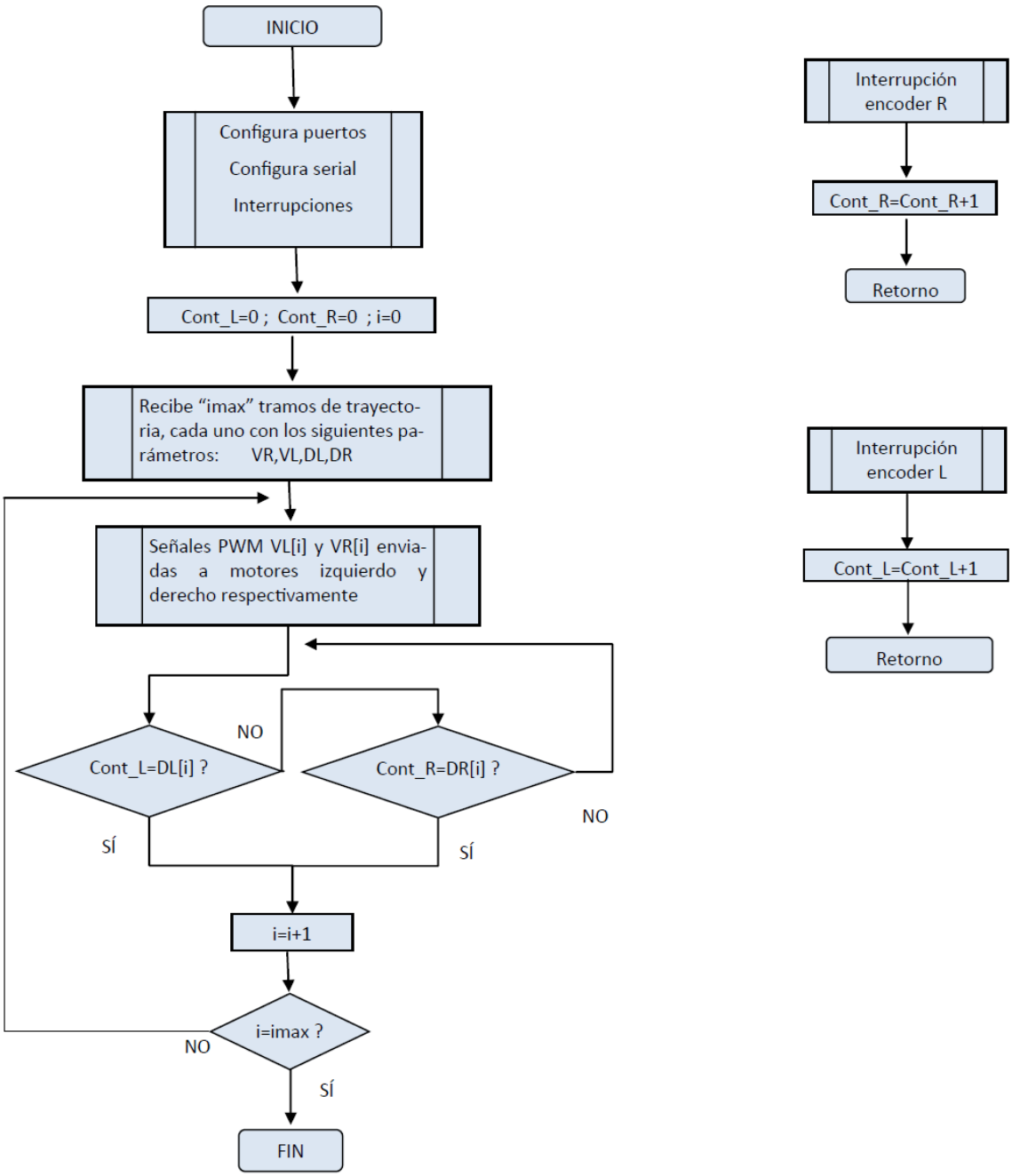

Figura 4.1: Diagrama de flujo del programa del micro-controlador ATmega8

4.2 Pruebas del programa de generación y seguimiento de trayectorias.

Una vez diseñado e implementado el sistema se procede a la fase de pruebas del mismo. Se hacen tres pruebas con el objetivo de observar y anotar el desempeño del sistema en cada una de estas pruebas. Se realizan tres tipos de mediciones iguales a cada una de las pruebas, estas son: mediciones con señales de velocidad PWM reales (de las pruebas de velocidades en vacío o sin carga) y mediciones con señales de velocidad PWM mejoradas (obtenidas de las pruebas de velocidades de los motores con carga).

Las tres pruebas a realizar son: trayectoria en línea recta, trayectoria hacia punto diagonal y trayectoria frente a obstáculos. Las pruebas son medidas con una regla de graduación en milímetros y los valores obtenidos son redondeados a centímetros.

4.2.1 Prueba 1: trayectoria en línea recta

En esta prueba se elige el punto final Pf (0,200cm) sin obstáculos. Esta prueba consiste de un tramo recto como se puede apreciar en la IGU de la figura 4.2

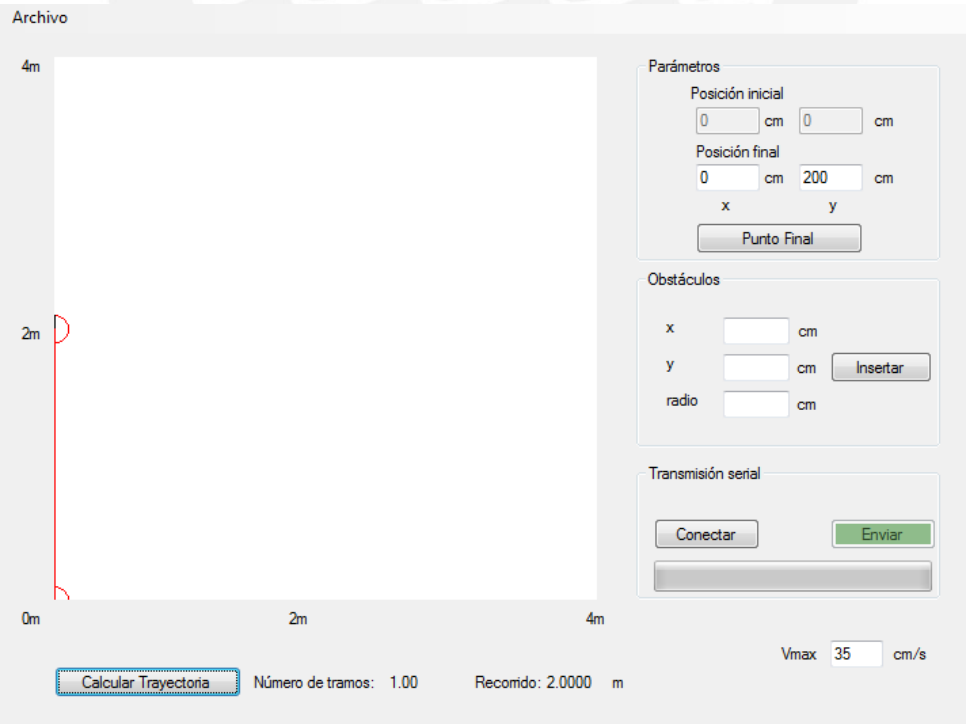

Figura 4.2: Trayectoria generada para la prueba 1

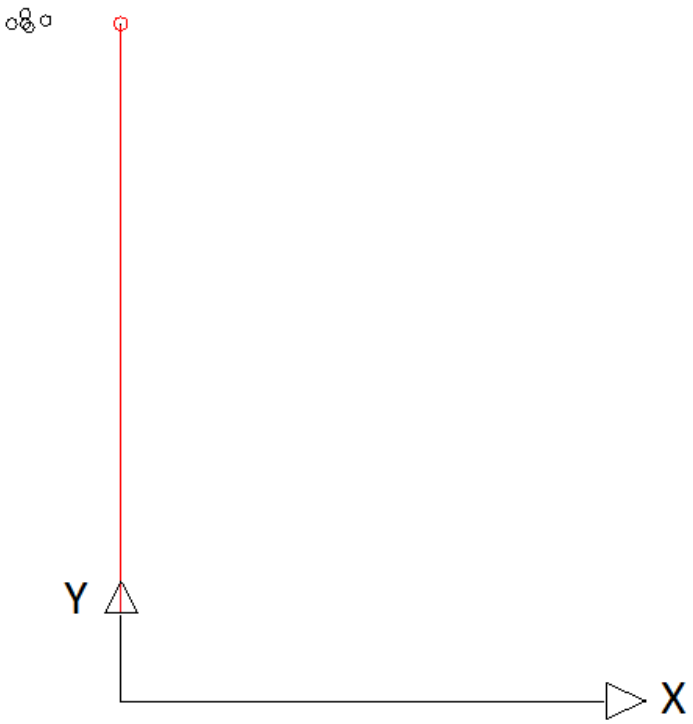

Figura 4.3: Resultados de seguimiento de la prueba 1 con data de velocidades reales

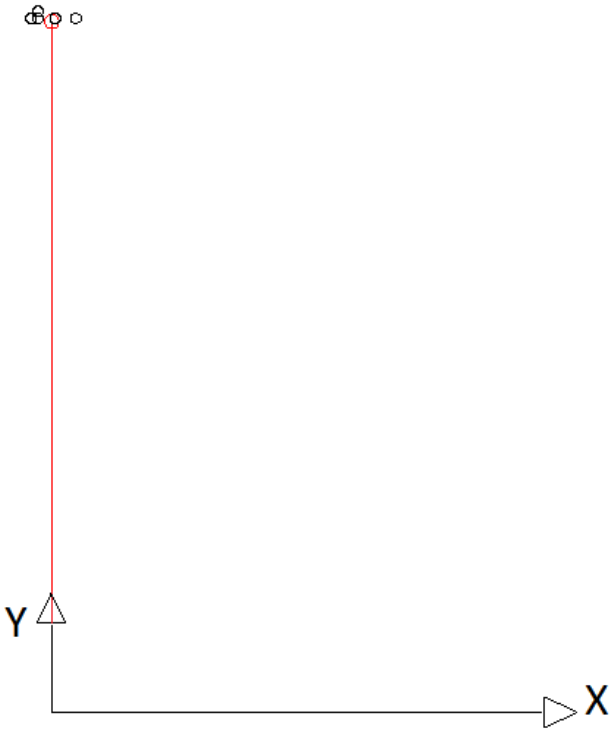

Figura 4.4: Resultados de seguimiento de la prueba 1 con data de velocidades mejorada

Del programa se obtiene que la distancia total de recorrido es 200 cm. Las pruebas a velocidades reales obtienen valores alejados hacia la izquierda del objetivo a diferencia de los resultados con las velocidades mejoradas. En la tabla 4.1 se muestra detalladamente los resultados de las pruebas así como el porcentaje de error obtenido en cada prueba.

Tabla 4.1: Resultados de pruebas para trayectoria en línea recta. Unidades en centímetros

	Prueba 1.1			**Prueba 1.2**			**Prueba 1.3**			**Prueba 1.4**			**Prueba 1.5**		
	X	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**
Programado	**0**	**200**	**0**	**0**	**200**	**0**	**0**	**200**	**0**	**0**	**200**	**0**	**0**	**200**	**0**
Real	-28	200	28	-27	199	27.02	-32	200	-32	-22	201	22.02	-28	203	28.16
	Error 14%			Error 13.51%			Error 16%			Error 11%			Error 14.08%		
Mejorado	-4	203	5	7	201	7.07	-6	201	6.08	-4	201	4.12	1	201	1.41
	Error 2.5%			Error 3.43 %			Error 3.04%			Error 2.06%			Error 0.71%		

El error se mide al comparar la diferencia de la distancia entre el punto de caída con el punto ideal con la distancia total ideal que debe recorrer el robot móvil, obteniendo así el porcentaje de error. El error de posicionamiento muestra que las pruebas mejoradas logran tener un margen de error bastante bajo comparado con las pruebas de velocidades reales. Debido a que la distancia de recorrido es 200cm y sólo hay un tramo el margen de error mejorado se mantiene bajo.

4.2.2 Prueba 2: trayectoria hacia punto diagonal

En esta prueba se elige el punto final Pf (200cm, 200cm) sin obstáculos. Esta prueba consiste de dos tramos como se puede apreciar en la IGU de la figura 4.5

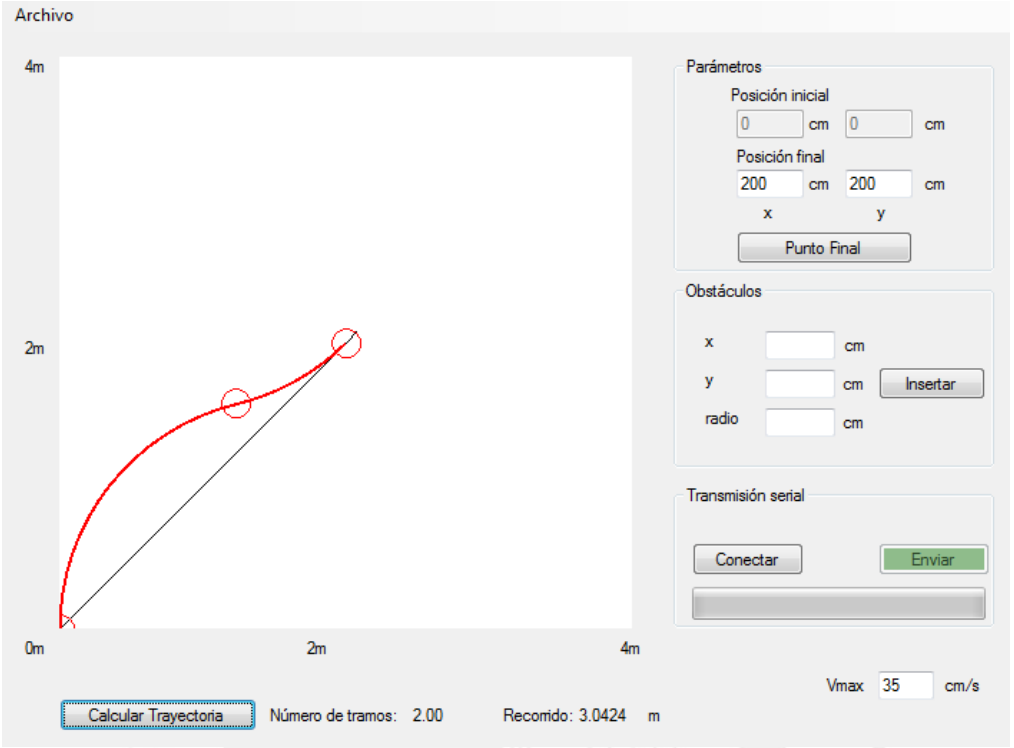

Figura 4.5: Trayectoria generada para la prueba 2

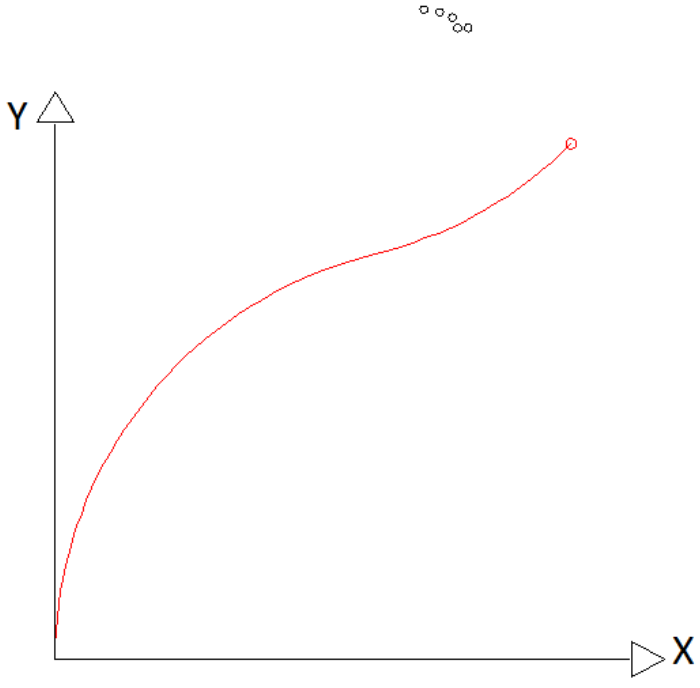

Figura 4.6: Resultados de seguimiento de la prueba 2 con data de velocidades reales

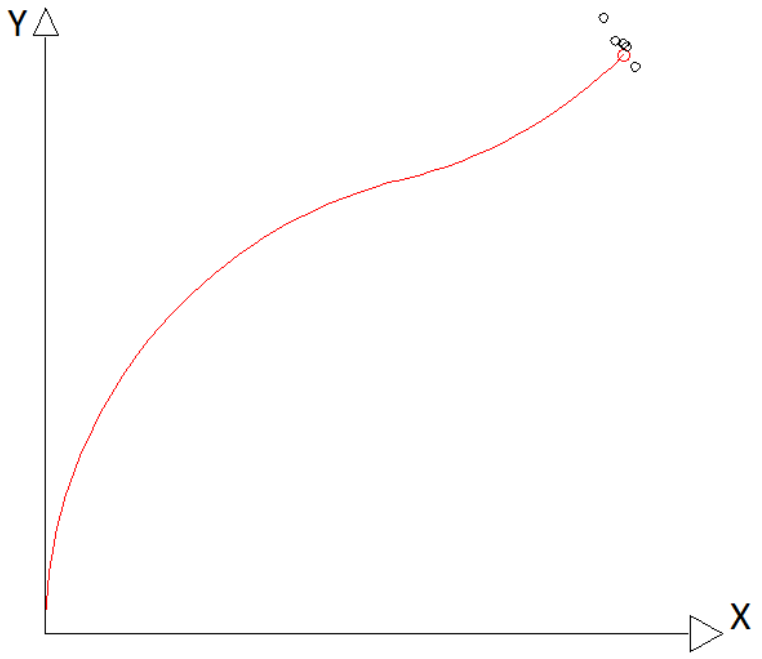

Figura 4.7: Resultados de seguimiento de la prueba 2 con data de velocidades mejorada

Del programa se obtiene que la distancia total de recorrido es 304.24 cm. Las pruebas a velocidades reales obtienen valores alejados hacia la izquierda del objetivo nuevamente a diferencia de los resultados con las velocidades mejoradas. En la tabla 4.2 se muestra detalladamente los resultados de las pruebas así como el porcentaje de error obtenido en cada prueba.

Tabla 4.2: Resultados de pruebas para trayectoria en diagonal. Unidades en centímetros

	Prueba 2.1			**Prueba 2.2**			**Prueba 2.3**			**Prueba 2.4**			**Prueba 2.5**		
	X	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**
Programado	**200**	**200**	**0**	**200**	**200**	**0**	**200**	**200**	**0**	**200**	**200**	**0**	**200**	**200**	**0**
Real	154	249	67.2	149	251	72.13	160	245	60.2	143	252	77.14	156	245	62,94
	Error 22.08%			Error 23.7%			Error 19.78%			Error 25.35%			Error 20.69%		
Mejorado	201	203	3.16	197	205	5.83	204	196	5.66	193	213	14.77	200	204	4
	Error 1.03%			Error 1.92%			Error 1.86%			Error 4.75%			Error 1.31%		

Se denota de la tabla 4.2 que el error aumenta ligeramente con respecto a los datos de la tabla 4.1 debido a la mayor longitud de recorrido y a que este presenta dos tramos (el cambio de velocidades entre tramos puede inducir a pequeños errores).

4.2.3 Prueba 3: trayectoria con obstáculo

En esta prueba se elige el punto final Pf (100cm, 100cm) con obstáculo con centro C (70cm,70cm) y radio de 20cm. Esta prueba consiste de cuatro tramos como se puede apreciar en la IGU de la figura 4.2.

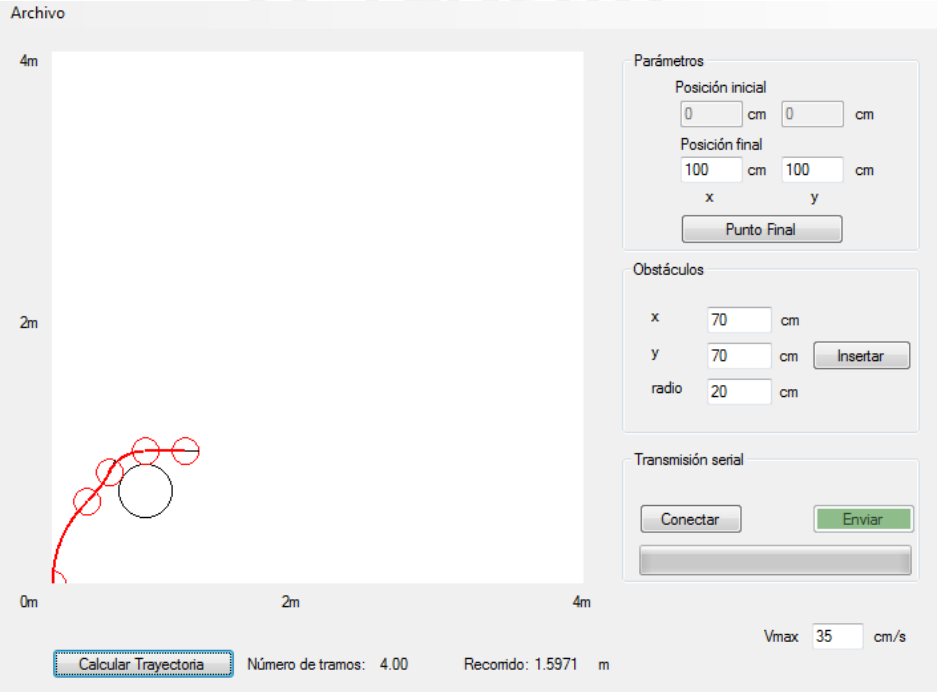

Figura 4.8: Trayectoria generada para la prueba 3

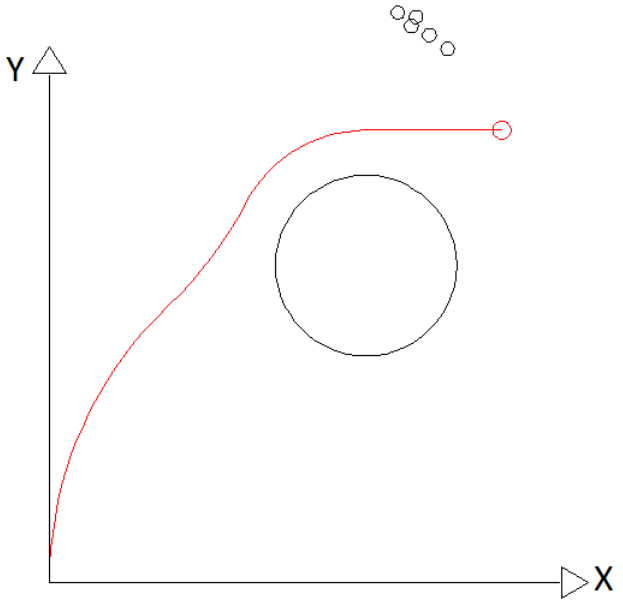

Figura 4.9: Resultados de seguimiento de la prueba 3 con data de velocidades reales

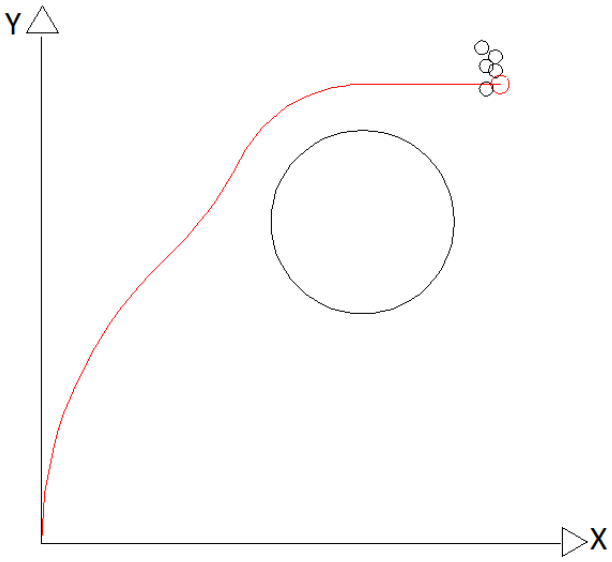

Figura 4.10: Resultados de seguimiento de la prueba 3 con data de velocidades mejorada

Tabla 4.3: Resultados de pruebas para trayectoria con obstáculo. Unidades en centímetros

	Prueba 3.1			Prueba 3.2			Prueba 3.3			Prueba 3.4			Prueba 3.5		
	X	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**	**X**	**Y**	**Δ**
Programado	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**	**100**	**100**	**0**
Real	81	125	31.4	84	121	26.4	80	123	30.48	88	118	21.63	77	126	34.7
	Error 19.7%			Error 16.5%			Error 19.05%			Error 13.52%			Error 21.69%		
Mejorado	97	104	5	96	108	8.94	99	103	3.16	99	106	6.08	97	99	3.16
	Error 3.12%			Error 5.59%			Error 1.97%			Error 3.8%			Error 1.97%		

De la tabla 4.3 se puede observar que el error de datos mejorados aumenta ligeramente con respecto a los datos de la tabla 4.1 y 4.2 aun cuando la distancia de recorrido es menos a la de los otros dos casos. Esto se debe a la acumulación de errores al momento de cambio de velocidad en cada tramo mayor.

Se puede comentar de los experimentos que las características físicas de cada motor varían frente a fuerzas presentes en los experimentos. El motor izquierdo pierde velocidad con respecto al derecho frente a cargas iguales distribuidas entre los motores (peso del robot móvil) provocando así el desviamiento hacia la izquierda notorias en todas las pruebas experimentales con parámetros reales (sin carga o en vacío).

CONCLUSIONES

1. Se concluye, a partir de los resultados obtenidos, que los parámetros de velocidad del sistema deben ser calculados de acuerdo a las condiciones físicas de funcionamiento (carga de los motores) para disminuir su error de precisión.

2. El error de exactitud del sistema aumenta ligeramente cuando el número de tramos a seguir en cada trayectoria aumenta. Este error se produce debido a la inercia de los motores, los cuales tardan unas fracciones de segundo en estabilizar sus velocidades a las deseadas en los tramos posteriores.

3. Aun cuando existen errores de exactitud en las pruebas realizadas se puede ver que los errores de precisión no son tan dispersos. Estos errores se deben a las condiciones físicas en el campo de experimentación como: a) dirección inicial del robot, b) desviaciones causadas por ranuras en la superficie y c) posibles pendientes ligeras en la superficie.

RECOMENDACIONES

1. Usar encoders de mayor resolución para los motores con la finalidad de obtener tablas de velocidades más precisas y, con esto, reducir el error de precisión de las pruebas realizadas.

2. Usar el robot móvil en superficies uniformes (sin ranuras ni pendientes) para disminuir las fuerzas externas incididas en las ruedas con lo cual se disminuye el error de precisión.

3. Asegurar el completo paralelismo entre ambas ruedas con la finalidad de reducir el error cumulativo que desvía al robot móvil de su trayectoria.

4. Usar un sistema de posicionamiento en tiempo real del robot móvil para conformar un sistema de control en lazo cerrado y corregir continuamente la posición del robot durante su movimiento.

5. Utilizar un sensor ultrasónico en la parte frontal del robot para detectar posibles variantes (nuevos obstáculos) en el campo de movimiento. Luego el robot móvil se detiene y reiniciará el cálculo de trayectorias desde aquel nuevo punto.

BIBLIOGRAFÍA

[1] Lopez, Ignacio, Sanz, Ricardo*: "Towards a Theory of General Autonomous Systems"*,UPM ASLab Seminars on Autonomous Systems I, 2007.

[2] Campion, G., Bastin, G and D'Andréa-Novel. *"Structural properties and classifications of kinematics anddynamics models of wheeled mobile robots"*. IEEE Transactions on Robotics and Automation. Vol. 12, No.1 February 1996.

[3] González Jiménez, J y Ollero Baturone, A. *"Estimación de la posición de un robot móvil Informática y Automática"*. Vol.29, pp.3-18. Asociación Española de Informática y Automática. Abril - 1996.

[4] (2005). *"History / Trivia on Robots. Science...Non-Fiction",* 185-186. Retrieved from Book Collection: Nonfiction database.

[5]Fujimori, A., Ogawa, Y., & Teramoto, M. (2000). *"Navigation of a Sonar-Equipped Mobile Robot with Real-Time Local Map Building"*. Proceedings of the Institution of Mechanical Engineers -- Part I -- Journal of Systems & Control Engineering, 214(5), 319-325. Retrieved from Academic Search Complete database.

[6] Rios, Luis H.; Bueno, Maximilianoy Sanchez, Santiago. (2008). *"Generación de trayectorias para un robot móvil empleando redes neuronales".* Revista "Scientia et Technica" Año XIV, Septiembre de 2008. Universidad Tecnológica de Pereira ISSN 0122-1701.

[7] Parhi, D., & Singh, M.. (2009). *"Real-time navigational control of mobile robots using an artificial neural network"*. Proceedings of the Institution of Mechanical Engineers: Part C Journal of Mechanical Engineering Science, *223*(C7), 1713-1725. Retrieved April 7, 2010, from ProQuest Science Journals. (Document ID: 1847506221).

[8] Su, L., Luo, C., & Zhu, F. (2009). *"Obtaining obstacle information by an omnidirectional stereo vision system"*. International Journal of Robotics & Automation, 24(3), 222-227. Retrieved from Computers & Applied Sciences Complete database.

[9] Hamidreza Chistas, *"Minimum Wheel-Rotation Paths for Differential Drive Mobile Robots Among Piecewise Smooth Obstacles"*

[10] M. de Berg.*" Computational geometry algorithms and applications".* Springer Verlag, 1987.