Planeación y seguimiento de trayectorias para un robot móvil

Aracely Yandún, Nelson Sotomayor, MSc.
Escuela Politécnica Nacional, Quito – Ecuador

***Resumen* – En este documento se presenta un breve estudio de los diferentes métodos para la planeación de trayectorias para robots móviles, haciendo énfasis en uno de estos métodos para ser desarrollado en LabVIEW e implementado en el robot móvil Robotino® de Festo. Se consideran entornos estructurados como fijos y conocidos, con lo cual el problema de planificación de movimientos se simplifica a que dadas una posición inicial y final, se desea generar una trayectoria óptima libre de obstáculos misma a ser seguida por el robot.**

***Índices* – Algoritmo A*, Diagramas de Voronoi, LabVIEW, Planeación de trayectorias, Robótica móvil, Robotino.**

I. INTRODUCCIÓN

La robótica móvil constituye una valiosa herramienta para el desarrollo de tecnologías para la creación de robots de navegación autónoma.
Para que el robot tenga la capacidad necesaria de ejecutar ciertas tareas a través de su movimiento es importante conocer previamente el espacio de trabajo, por ejemplo, para planear una ruta libre de colisión es importante conocer la ubicación de los obstáculos.
En la actualidad existe un gran número de métodos para la planeación y generación de trayectorias, desde algoritmos simples hasta lógicas de programación más complejas, por ende es importante seleccionar un método adecuado acorde a la aplicación requerida.

II. DESARROLLO DEL PROYECTO

A. Navegación en robots móviles

La navegación es la técnica de conducir un robot móvil mientras atraviesa un entorno para alcanzar un destino o meta sin chocar con ningún obstáculo [12].

Cuando se desconoce el entorno, el robot debe poseer la capacidad de reaccionar ante situaciones inesperadas, esto se logra a través de la percepción del entorno mediante el uso de sensores. Mientras que si se trata de un entorno conocido, el uso de los sensores se vuelve secundario y las tareas a seguir serían: planificar una trayectoria óptima libre de obstáculos, a partir de puntos de partida y llegada y obviamente que el robot pueda seguir y cumplir físicamente esta trayectoria.

Se considera a un robot como un objeto rígido al cual se le puede asociar un sistema de coordenadas móvil:

$$q = (p, \theta) \quad (1)$$

Donde p es la posición y θ la orientación. La localización del vehículo en un determinado instante de tiempo se encuentra definida por la relación existente entre el sistema de coordenadas global F_g en virtud del cual está definido todo el entorno de trabajo y su sistema de coordenadas locales asociado F_r [13].

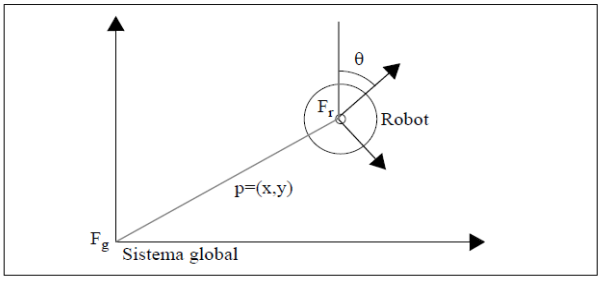

Fig 1. Sistema de coordenadas global, y sistema local asociado al robot

B. Métodos de planificación de trayectorias

1) *Grafos de visibilidad:* Para la generación del grafo, este método introduce el término de visibilidad, según el cual, define dos puntos del entorno como *visibles* si y solo sí se pueden unir mediante un segmento rectilíneo que no intercepte ningún obstáculo. En otras palabras, el segmento definido debe yacer en el espacio libre del entorno C_l [13].
Se consideran como nodo del grafo a la posición inicial, la posición final y todos los vértices de los obstáculos, siendo el grafo el resultado de la unión de los nodos visibles, tal como se muestra en la Fig. 2.

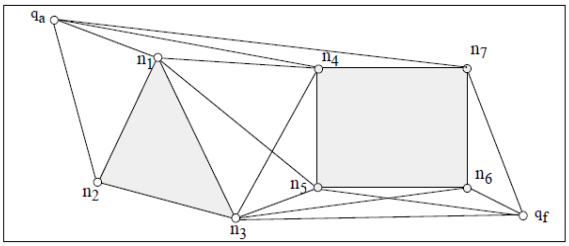

Fig 2. Grafo de visibilidad en un entorno de dos obstáculos, tomado de [13]

A través de un algoritmo de búsqueda de grafos se escoge la ruta más óptima que una la posición inicial con la final.

2) *Diagramas de Voronoi:* Se encuentran entre las más importantes estructuras en geometría computacional, este diagrama codifica la información de proximidad entre elementos [15].

A. Yandún, es Ingeniera de Obra en Microcircuits Cía. Ltda., Quito-Ecuador, (e-mail: aracely.yandun@yahoo.com).
N. Sotomayor, es Jefe del Departamento de Automatización y Control Industrial y Profesor Principal T/C en la Facultad de Ingeniería Eléctrica y Electrónica de La Escuela Politécnica Nacional, Quito-Ecuador, (e-mail: nelson.sotomayor@epn.edu.ec).

Los diagramas de Voronoi se definen como una proyección del espacio libre del entorno en una red de curvas unidimensionales yacientes en dicho espacio libre. Formalmente se definen como una retracción (Janich, 1984) con preservación de la continuidad. Si el conjunto C_l define las posiciones libres de obstáculos de un entorno, la función retracción RT construye un subconjunto C_v continuo de C_l [13].

$$RT(q): C_l \rightarrow \frac{C_v}{C_v} \subset C_l \quad (2)$$

Existe un camino desde una posición inicial q_a hasta una posición final q_f, libre de obstáculos, si y solo si existe una curva continua desde $RT(q_a)$ hasta $RT(q_f)$.

La principal idea de la construcción del diagrama de Voronoi es ampliar al máximo la distancia que existe entre el robot y los obstáculos, por tanto, el diagrama resulta el lugar geométrico de las configuraciones que se encuentran a la misma distancia de los obstáculos más próximos del entorno.

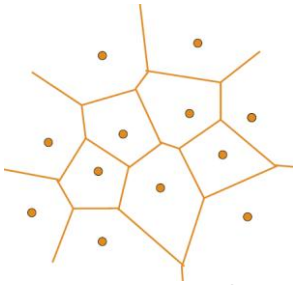

Fig 3. Diagrama de Voronoi de obstáculos representados como puntos, tomado de [17]

Para una representación más real se consideran a los obstáculos como polígonos, debido a que físicamente un obstáculo no es un punto. El diagrama de Voronoi estará compuesto por dos tipos de segmentos: rectas y parábolas.

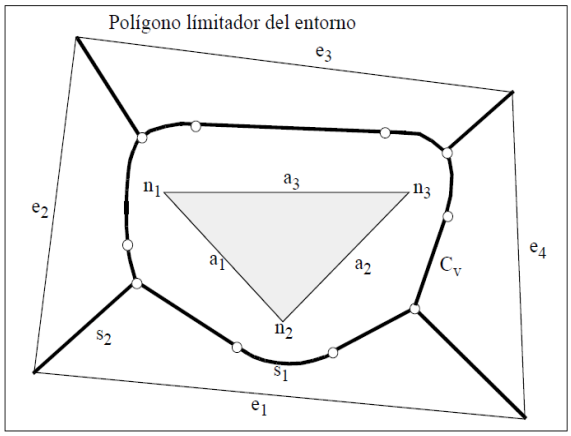

Fig 4. Retracción del espacio libre en un diagrama de Voronoi, tomado de [13]

El lugar geométrico de las configuraciones que se hallan a la misma distancia de dos aristas de dos obstáculos distintos será una recta, y si se trata de una arista y un vértice, será una parábola.

En la Fig. 4 se puede observar el diagrama de Voronoi C_v representado por las líneas gruesas, el entorno se encuentra delimitado por un polígono de aristas $\{e_1, e_2, e_3, e_4, e_5\}$ y un obstáculo triangular de vértices $\{n_1, n_2, n_3\}$ y aristas $\{a_1, a_2, a_3\}$. Claramente se pueden apreciar los dos tipos de segmentos que componen el diagrama de Voronoi, el segmento s_1 (parabólico) corresponde al lugar geométrico de los puntos equidistantes entra el vértice n_2 y la arista e_1. Así como el segmento s_2 (rectilíneo) corresponde al lugar geométrico entre las aristas e_1 y e_2.

Al igual que los grafos de visibilidad, este método también trabaja en entornos totalmente conocidos y con obstáculos modelados mediante polígonos. Sin embargo, también existen versiones para la utilización del mismo con obstáculos inesperados (Meng, 1988) [13].

3) *Roadmap Probabilístico (PRM):* Consiste en generar un número n de configuraciones libres de colisión de forma aleatoria y uniforme en toda el área de trabajo.

Se prosigue a conectar cada uno de los nodos con sus nodos más cercanos según una métrica que depende del número de objetos en el entorno de trabajo y finalmente se aplica un algoritmo que obtenga la ruta más óptima, en este caso el algoritmo A*.

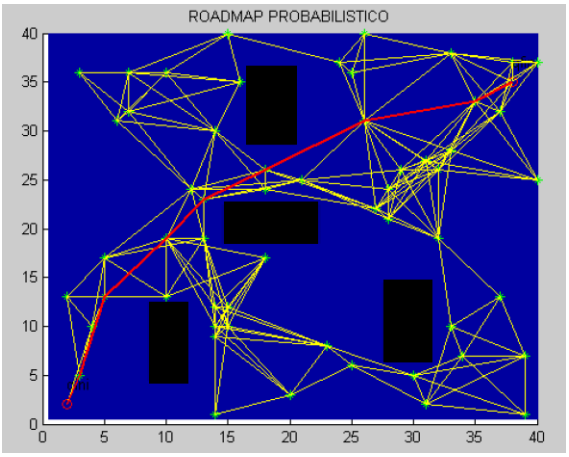

Fig 5. Enlaces válidos en la etapa de conexión y camino entre q_{ini} y q_{fin} mediante el algoritmo A*, tomado de [16]

4) *Modelado del Espacio Libre:* Igualmente en este método los obstáculos se los representa como polígonos. La planificación se lleva a cabo a través de los CRG, cilindros rectilíneos generalizados, y al igual que Voronoi, con el uso de los CRG se pretende que el robot se mueva lo más alejado de los obstáculos. La ruta será una configuración de CRG interconectados, tal que la configuración inicial o de partida se encuentre en el primer cilindro de la sucesión y la configuración final en el último cilindro.
El proceso para construir un CRG será el siguiente:

- Cálculo del eje del CRG, se define como la bisectriz del ángulo α formado por el corte de las rectas que contienen las aristas ${}^1a_i$ y ${}^2a_j$ que cumplen con las condiciones antes mencionadas.

- Por ambos lados de dichas aristas se construyen rectas paralelas al eje, con origen en los vértices de las aristas implicadas y con extremo señalado por la proyección del primer obstáculo que corta el eje.

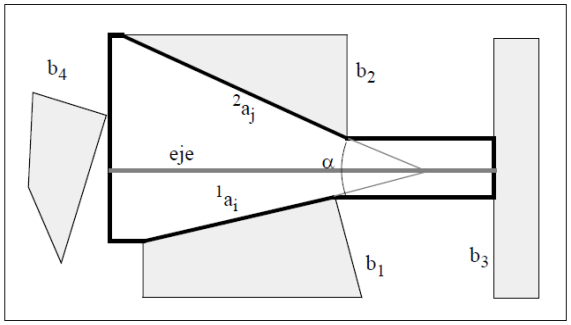

Fig 6. Construcción de un CRG, tomado de [13]

Repitiendo este proceso se construye una red CRG en el entorno del robot que modela el espacio libre del mismo. El robot navegará por el eje del cilindro, en el cual se encuentran anotadas para cada punto el rango de orientaciones admisibles. El paso de un CRG a otro se produce siempre y cuando sus ejes intercepten y la intersección del rango de orientaciones admisibles en el punto de corte de ambos ejes no sea nulo [13].

5) *Descomposición en celdas:* En este método no se encuentra una sucesión de segmentos, sino una sucesión de celdas, por lo que es necesaria la construcción de un grafo de conectividad encargado de definir la ruta además de la descomposición de celdas.

Dentro de los métodos basados en descomposición en celdas, el método más sencillo es el de descomposición trapezoidal. Se construyen rectas paralelas al eje Y a partir de los vértices de cada elemento del entorno, estas rectas quedan delimitadas por el corte con las líneas de los elementos del entorno.

El grafo de conectividad se construye a través de la unión de los puntos medios de las rectas definidas, como se observa en la Fig. 7.

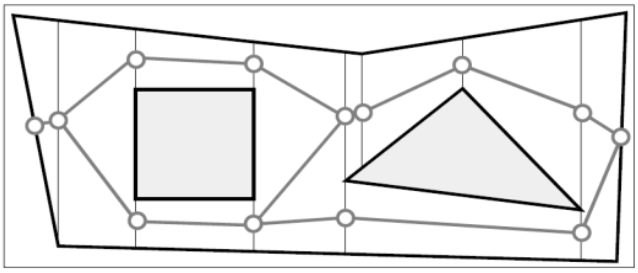
Fig 7. Grafo de conectividad de una descomposición trapezoidal, tomado de [13]

6) *Campos Potenciales:* Este método está basado en técnicas reactivas de planificación, esta técnica se centra en la planificación local en entornos desconocidos.

Esta teoría considera al robot como una partícula bajo la influencia de un campo potencial artificial. La función potencial *U* en un punto *p* del espacio euclídeo, consiste en la composición de un potencial atractivo $U_a(p)$ que atrae el robot a la posición destino y un potencial repulsivo $U_r(p)$ que lo hace alejarse de los obstáculos. Tal que:

$$U(p) = U_a(p) + U_r(p) \quad (3)$$

El potencial artificial $U(p)$ influye en la fuerza artificial $F(p)$, tal que:

$$F(p) = -\nabla U(p) \quad (4)$$

Así mismo la fuerza $F(p)$ está compuesta por una fuerza de repulsión y una fuerza de atracción:

$$F(p) = F_a(p) + F_r(p) \quad (5)$$

Así, la navegación basada en campos potenciales se basa en llevar a cabo las siguientes acciones [13]:

i. Calcular el potencial $U(p)$ que actúa sobre el vehículo en la posición actual *p* según la información recabada de los sensores.
ii. Determinar el vector fuerza artificial $F(p)$.
iii. En virtud del vector calculado construir las consignas adecuadas para que los actuadores del vehículo hagan que éste se mueva según el sentido, dirección y aceleración dadas por $F(p)$.

La iteración del ciclo anterior constituye una navegación reactiva basada en campos potenciales. El potencial de atracción debe ser función de la distancia euclídea a la posición destino, mientras más cerca esté el robot, el potencial debe disminuir su influencia. Por otro lado, el potencial repulsivo debe solo influir el momento en que el robot se encuentra demasiado cerca de los obstáculos. En la posición destino, es necesario que la suma de los dos potenciales sea nula.

El problema que existe en este tipo de métodos son los mínimos locales, que son lugares donde el potencial resulta nulo pero no se trata de la posición final. Evitar estos problemas implica definir ciertas funciones potenciales que eviten la aparición de mínimos locales, por ejemplo, modelados mediante círculos.

Para la implementación del presente proyecto se seleccionó a los diagramas de Voronoi como el método de planeación de trayectorias para ser desarrollado, debido a que en comparación a los otros métodos mencionados, este posee menos desventajas y además presenta la mayor seguridad en la ruta del robot al considerar a la misma lo más alejada de los obstáculos.

C. Generación de caminos

El camino se lo construye en base a la planificación de la ruta y debe estar libre de obstáculos, la importancia de un camino con buenas propiedades se basa en la capacidad del seguidor de caminos para ejecutar la navegación con el menor error posible. La función del generador es convertir una ruta en un camino, llevar al robot de una posición inicial a una final, de tal manera que se elimine la restricción de omnidireccionabilidad inherente a la definición de ruta.

El camino se define como la discretización de una curva continua que interpola ciertos puntos elegidos de la ruta calculada por el planificador. Por tanto, el problema de la definición de un camino con buenas propiedades pasa por la construcción de la función camino adecuada que las posea.

Las características buscadas son aquellas que hacen posible el seguimiento del camino especificado según el comportamiento cinemático y dinámico del vehículo [13].

Para la generación de la ruta en este proyecto, se hace uso de un algoritmo denominado A*.

1) *Algoritmo A*:* El algoritmo de búsqueda A* se clasifica dentro de los algoritmos de búsqueda en grafos. Presentado por primera vez en 1968 por Peter E. Hart, Nils J. Nilsson y Bertram Raphael, el algoritmo A* encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo [19].

El problema de algunos algoritmos de búsqueda es que se guían en exclusiva por la función heurística, la cual puede no indicar el camino de coste más bajo, o solo se guían por el coste real de desplazarse de un nodo a otro. Es por ello, que un buen algoritmo de búsqueda informada, debería tener en cuenta ambos factores: el valor heurístico de los nodos y el coste real del recorrido, de esta manera se obtiene al algoritmo A*.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

Fig 8. Ejemplo de aplicación del algoritmo A*, tomado de [19]

Una heurística es una función matemática $h'(n)$ definida en los nodos de un árbol de búsqueda que sirve como una estimación del coste del camino más económico de un nodo dado hasta el nodo objetivo.

El algoritmo A* utiliza una función de evaluación:

$$f(n) = g(n) + h'(n) \quad (6)$$

Donde:

- $h'(n)$ representa el valor heurístico del nodo n a evaluar, desde el nodo actual hasta el final.
- $g(n)$ representa el coste real del camino recorrido para llegar a dicho nodo.

D. Robotino®

Robotino® es un sistema de robot móvil de alta calidad con accionamiento omnidireccional. Cuenta con 3 unidades de accionamiento que permiten movimientos en todas las direcciones: adelante, atrás y lateralmente. Adicional a esto, el robot puede girar sobre un punto.

Además Robotino® se encuentra equipado con una webcam que permite visualizar una imagen de cámara en vivo y una serie de sensores analógicos para mediciones de distancia, es el caso de los sensores binarios para protección de colisiones y sensores digitales para detectar la velocidad real.

Adicionalmente puede conectarse actuadores y sensores adicionales en el Robotino® a través de una interfaz de entradas/salidas.

Robotino® consiste en un PC embebido con una tarjeta compact flash, en esta se han instalado el sistema operativo Linux así como algunas aplicaciones de demostración. Estas aplicaciones pueden ejecutarse directamente desde el panel de control del Robotino®.

Fig 9. Robot móvil Robotino® de Festo

Para programar el Robotino® en un PC se utiliza el software Robotino®View, este software es capaz de transmitir señales de manera inalámbrica al controlador del motor, así como visualizar, cambiar y evaluar valores de los sensores. La programación se la puede hacer incluso durante el funcionamiento real. Otras alternativas de programación del Robotino® son APIs Linux, C++, LabVIEW y Matlab.

Debido a las altas prestaciones que aportan al sistema la necesaria "inteligencia", a Robotino® se lo considera autónomo.

E. Desarrollo del software

Para la programación de la aplicación presentada en el presente proyecto se utilizó LabVIEW, software desarrollado por National Instruments, debido a que proporciona todas las herramientas requeridas para el desarrollo del proyecto.

LabVIEW es un entorno de programación gráfica usado para desarrollar sistemas sofisticados de medida, pruebas y control a través de íconos gráficos y cables que parecen un diagrama de flujo. Ofrece una integración con miles de dispositivos de hardware y brinda cientos de bibliotecas integradas para análisis avanzado y visualización de datos.
Para la implementación del programa fue necesario hacer uso de tool kits adicionales como:

- LabVIEW Robotics Module
- MathScript RT Module
- Robotino LabVIEW driver.

El desarrollo del programa tiene varias etapas, como primer paso se tiene la adquisición de datos, es decir el ingreso del mapa de entorno del robot al LabVIEW, después se tiene una etapa de procesamiento de estos datos donde se obtienen los puntos que conforman los bordes de los obstáculos pertenecientes al mapa de entorno. Se halla el diagrama de Voronoi en base a los puntos hallados y seguido se procede a discriminar los puntos del diagrama generado que no corresponden a lo requerido, por último, en base al diagrama de Voronoi final se obtiene la trayectoria óptima a través del

algoritmo A* y la misma es descargada en la plataforma móvil Robotino® de Festo.

Fig 10. Diagrama de bloques de la programación

A continuación se presenta una breve explicación de la programación en cada etapa.

1) *Mapa de entorno:* Este proyecto se aplica para mapas de entorno conocidos, los obstáculos son representados como polígonos, por lo que se ingresan entonces las coordenadas de los puntos de los vértices correspondientes a cada polígono (obstáculo), incluyendo los límites del mapa en un array. Se ingresan los obstáculos representados como coordenadas de puntos debido a que más adelante se hará uso de la función *voronoi* perteneciente al MathScript Node y la misma trabaja solo con puntos.

Ingresados los vértices de los polígonos, se rota todos los puntos ingresados un cierto ángulo despreciable, debido a que en la obtención de trayectorias, realizada más adelante a través del algoritmo A*, se generaban grandes problemas el momento en que se dispone cierta cantidad de puntos cuya componente en y era la misma en todos ellos, es el caso del borde del mapa de entorno, por lo que al rotar un ángulo despreciable los puntos se elimina esta restricción.

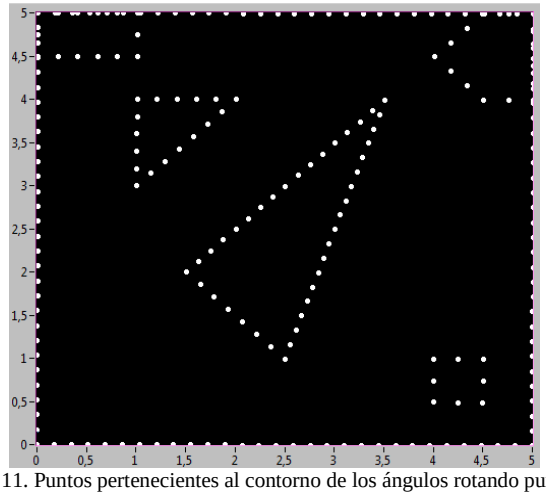

Fig 11. Puntos pertenecientes al contorno de los ángulos rotando puntos

Una vez realizado este proceso, se creó un subVI para obtener más puntos pertenecientes al contorno de todos los obstáculos, de esta manera se obtienen más datos para hallar un apropiado diagrama de Voronoi.

2) *Diagrama de Voronoi:* A partir de los datos del mapa de entorno, se dispone de un vector para coordenadas de los puntos en x y otro para y, entonces se ingresan los mismos en un MathScript Node del LabVIEW y se hace uso de la función *voronoi*, misma que obtiene diagrama de Voronoi en base a datos como puntos.

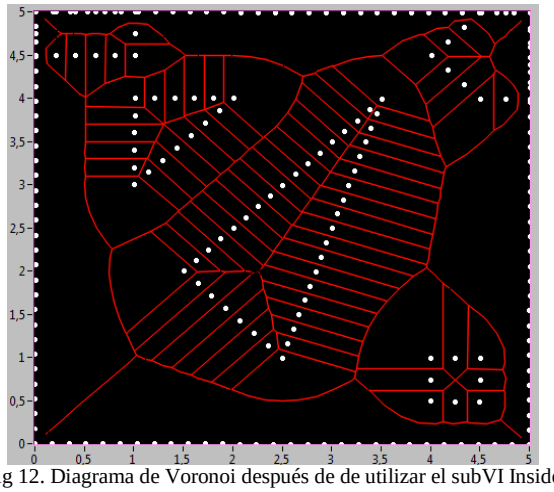

Fig 12. Diagrama de Voronoi después de de utilizar el subVI Inside

Se hace uso de 2 subVIs para hallar el diagrama de voronoi del mapa de entorno ingresado con el objetivo de discriminar puntos que se encuentran ya sea fuera de los límites del mapa de entorno o dentro de los obstáculos.

Como se muestra en la Fig. 12 se puede apreciar claramente que una vez realizada la primera discriminación, el diagrama de Voronoi está compuesto solo por puntos que se encuentran dentro de los límites del entorno.

El diagrama de Voronoi obtenido por el momento, no es el que se requiere para obtener la ruta más segura para el robot, claramente se puede observar que se considera una ruta segura a rectas que pertenecen a los obstáculos, por esta razón es necesario discriminar estos puntos de estas rectas a través de un segundo subVI.

Se obtiene entonces el diagrama de Voronoi final (líneas de color rojo), tal cual se muestra en la Fig. 13, éste representa la ruta más segura por la cual el robot va a desplazarse dentro de todo el ambiente.

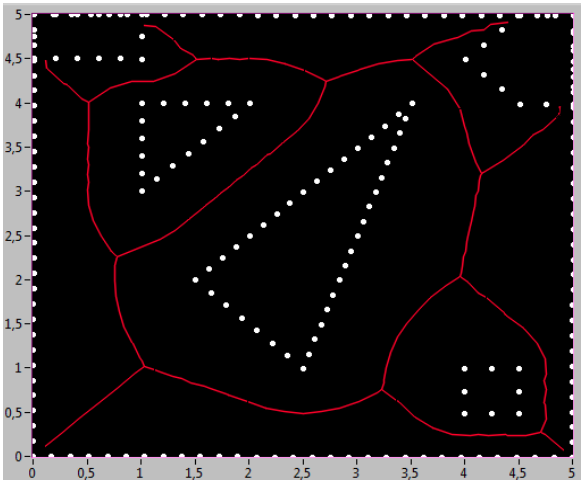

Fig 13. Diagrama de Voronoi del mapa de entorno actual

3) *Generación de la trayectoria:* Para la obtención de la trayectoria se sigue el esquema que se muestra en la Fig. 14.

Fig 14. Diagrama de programación para la generación de la trayectoria

Para esta etapa de la programación se hace uso de los VI's pertenecientes al NI LabVIEW Robotics Module, mismos que nos permiten crear el mapa de entorno, ingresar los diferentes nodos pertenecientes al diagrama de Voronoi, los puntos de partida y llegada, luego aplicar el algoritmo A* para obtener la trayectoria más óptima y finalmente obtener los nodos de la ruta a seguir por el robot.

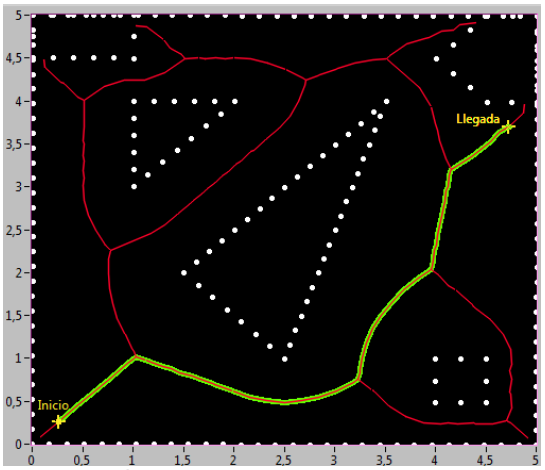

Fig 15. Trayectoria óptima acorde al mapa de entorno basada en diagramas de Voronoi

En la Fig. 15 se puede observar que la trayectoria más óptima desde los puntos de partida y llegada, indicados en el mapa, es la que se muestra de color verde.

4) *Programación de Robotino®:* Una vez que se obtenida la trayectoria óptima a seguir, se debe descargarla en la plataforma móvil Robotino® para que físicamente el robot siga esta trayectoria y de esta manera comprobar los algoritmos implementados.

Los datos con los que se dispone están en un array, mismo que contiene los puntos pertenecientes a las rectas que conforman la trayectoria a seguir por el robot.
Para que el robot pueda seguir la trayectoria, se utilizó la lógica de movimientos presentada en la Fig. 16.

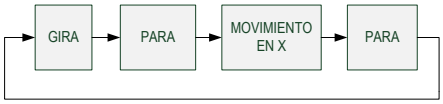

Fig 16. Lógica de movimientos de Robotino

Como primer paso es necesario que el robot gire hasta tomar la posición de la recta, una vez que el robot se encuentra orientado de frente, se mueve la distancia necesaria para completar el seguimiento de la recta y el robot vuelve a girar para tomar posición de la siguiente recta a seguir. Se repite la misma lógica hasta recorrer todas las rectas que conforman la trayectoria.

Es necesario conocer entonces: ángulo y sentido de giro y la distancia a recorrer.

Para las condiciones de programación tanto para el sentido de giro como para el ángulo. Se utilizó una lógica de desplazamiento y rotación de ejes de coordenadas teniendo como nuevas referencias x' y y' tal como se observa en la Fig. 17. Se obtuvo que:

- Si x' es de valor *positivo*, el ángulo de giro será α.
- Si x' es de valor *negativo*, el ángulo de giro será 180º- α.
- Si y' es de valor *positivo*, el sentido de giro será anti horario.
- Si y' es de valor *negativo*, el sentido de giro será horario.

Siendo $P_2(x', y')$ el segundo punto que pertenece a la recta a seguir, y el primer punto $P_1(0,0)$. Para hallar α, se utiliza la ecuación:

$$\alpha = \left| \tan^{-1}\left(\frac{y' - 0}{x' - 0}\right) \right| \quad (7)$$

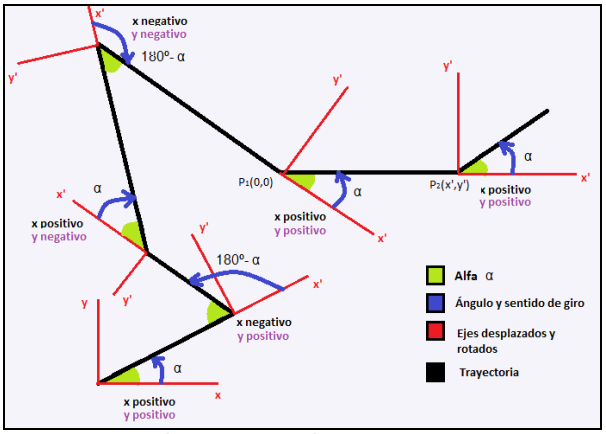

Fig 17. Lógica para hallar ángulos y sentido de giro

La distancia requerida para el movimiento en x se la calcula a través de la distancia euclídea:

$$d.e. = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)} \quad (8)$$

Para programar el hardware del Robotino® a través del LabVIEW, se hace uso de una librería exclusiva que se muestra en la Fig. 18, misma que contiene todos los VI's necesarios para programar al robot según la aplicación requerida.

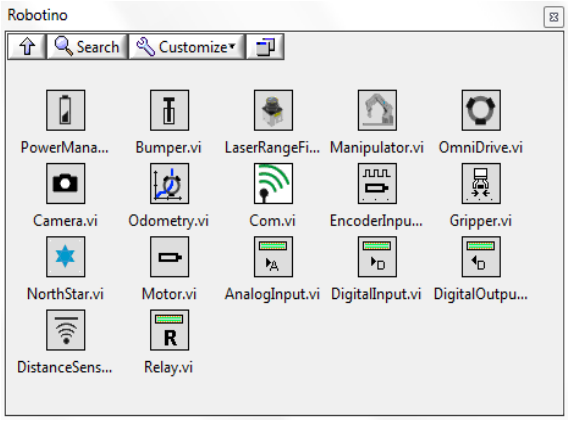

Fig 18. Robotino LabVIEW driver

F. *Pruebas y resultados*

1) *Prueba 1:* El mapa de entorno ingresado, contiene 5 obstáculos de varias formas y tamaños, el diagrama de Voronoi y la trayectoria obtenida a seguir por el Robotino® se muestra en la Fig. 19.

Es importante mencionar, que el mapa de entorno ingresado no es real, la idea de esta prueba es comprobar que el robot se desplace no sólo en la trayectoria generada sino además, los mismos puntos y distancias correspondientes.

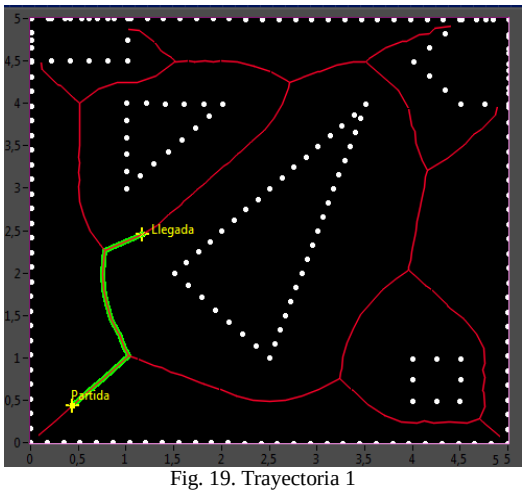

Fig. 19. Trayectoria 1

Se colocó un marcador en el eje del robot y como se observa en la Fig.20, el robot siguió al menos la forma correspondiente a la trayectoria requerida.

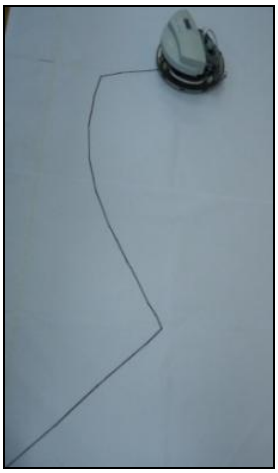
Fig 20.Trayectoria seguida por el robot

Para análisis de errores, se tomaron 6 puntos a lo largo de la trayectoria, tanto teóricos como prácticos y se obtuvieron los resultados mostrados en la Tabla 1.

Como se puede observar en la Tabla I los errores cometidos tanto en el eje x como en el eje y no sobrepasan el 7%, la mayoría de ellos no llegan ni al 5% de error por lo que se puede decir que el error de posición cometido en el seguimiento de esta trayectoria es relativamente pequeño y se puede considerar a los resultados obtenidos como satisfactorios.

Se puede justificar los errores cometidos no solo por posibles fallos en el programa o por la respuesta del hardware del robot a través de posibles problemas en la comunicación WLAN, sino también a errores producidos en las medidas dependiendo de la exactitud en que el robot fue colocado en su posición y orientación inicial, además se debe considerar la manera en que se tomaron las medidas respecto a un sistema de referencia.

TABLA I
RESULTADOS PRUEBA 1

	DATOS PRÁCTICOS		DATOS TEÓRICOS		ERRORES	
	X [m]	Y [m]	X [m]	Y [m]	X [%]	Y [%]
Punto 1	0,35	0,35	0,35	0,34	0	2,9
Punto 2	0,60	0,60	0,60	0,59	0	1,7
Punto 3	0,55	0,675	0,56	0,69	1,8	2,2
Punto 4	0,34	1,12	0,36	1,20	5,5	6,7
Punto 5	0,33	1,84	0,35	1,82	5,7	1,1
Punto 6	0,72	2,05	0,74	2,03	2,7	1

2) *Prueba 2:* El mapa de entorno ingresado, es exactamente el mismo de la prueba anterior, con 5 obstáculos de varias formas y tamaños, por ende, el diagrama de Voronoi generado será el mismo.

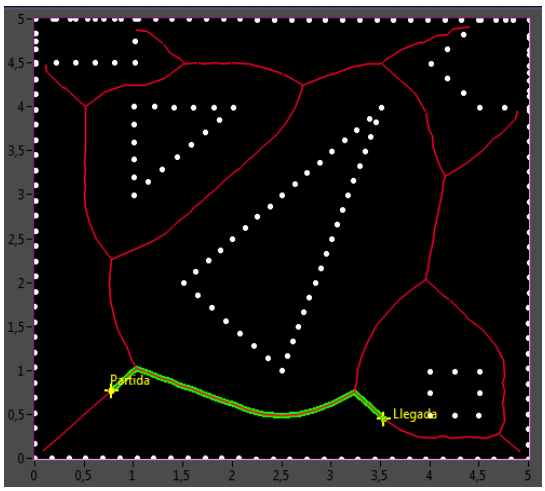

Fig 21. Trayectoria 2

A diferencia de la prueba anterior, en esta prueba se escogen distintos puntos de partida y llegada, por lo que la trayectoria a seguir por el Robotino® tiene una forma diferente a la presentada en la *prueba 1*. Se puede observar esta trayectoria en la Fig. 21.

De igual manera, en la Fig. 22 se puede observar la trayectoria seguida por el Robotino® y si se la compara a ésta con la generada en LabVIEW (Figura 21) se puede notar la similitud en cuanto a la forma de ambas trayectorias.

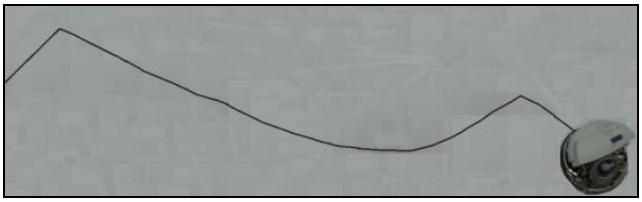
Fig 22. Trayectoria 2

Para análisis de errores se tomaron 6 puntos a lo largo de la trayectoria, se obtuvieron los resultados que se muestran en la Tabla II.

TABLA II
RESULTADOS PRUEBA 2

	DATOS PRÁCTICOS		DATOS TEÓRICOS		ERRORES	
	X [m]	Y [m]	X [m]	Y [m]	X [%]	Y [%]
Punto 1	0,25	0,25	0,25	0,25	0	0
Punto 2	0,615	0,10	0,61	0,10	0,82	0
Punto 3	1,20	-0,16	1,20	-0,15	0	6,7
Punto 4	1,90	-0,29	1,89	-0,27	5,3	9,3
Punto 5	2,38	-0,06	2,41	-0,05	1,2	20
Punto 6	2,75	-0,32	2,76	-0,33	0,36	3

En cuanto a los errores generados, se puede observar en la Tabla II que el error máximo cometido es del 20%, pero téngase en cuenta que es en el *punto 5* en el que la coordenada en y tiene un valor de 0,05 [m] teóricamente y en la práctica se obtuvo 0,06 [m], es decir 1 [cm] de error, lo cual se considera despreciable debido a que no significa que el robot se movió 6 [cm] en lugar de 5 [cm] sino que después de girar cierto ángulo y haber recorrido cierta distancia llegó a esas coordenadas, como por ejemplo en este mismo punto en la coordenada en x se tienen 3 [cm] de diferencia entre valor práctico y teórico, pero las coordenadas al ser de alto valor generan un error del 1,2%. En cuanto a los demás errores obtenidos, se los considera bajos por lo que el resultado de la trayectoria seguida se considera satisfactorio.

Así mismo se justifica los errores cometidos debido a la posible inexactitud de la posición de partida del robot, así como la toma de datos respecto a un sistema de referencia elegido.

3) *Prueba 3:* El mapa de entorno ingresado para esta prueba es el del laboratorio de Microprocesadores, el cual comprende un contorno rectangular de 6,97x7,6 (m), posee 6 obstáculos de forma rectangular mismos que son de diferentes tamaños, el diagrama de Voronoi y trayectoria generada por la HMI a seguir por el Robotino® en base a los puntos de partida y llegada deseados se muestran en la Fig 23.

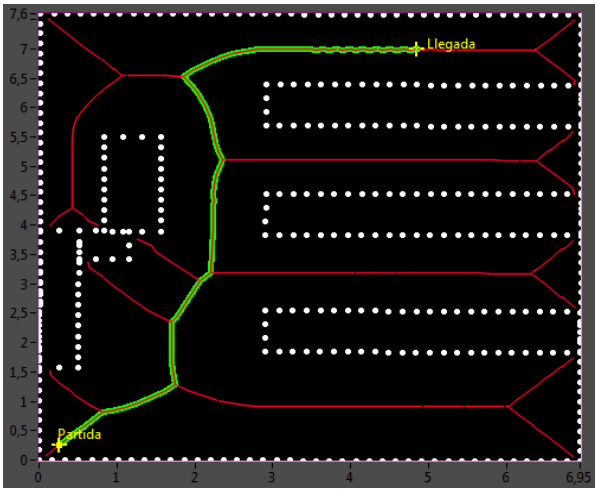

Fig 23. Trayectoria 3

Para análisis de errores, se tomaron 9 puntos a lo largo de la trayectoria, tanto teóricos como prácticos y se obtuvieron los resultados mostrados en la Tabla III.

TABLA III
RESULTADOS PRUEBA 3

	DATOS PRÁCTICOS		DATOS TEÓRICOS		ERRORES	
	X [m]	Y [m]	X [m]	Y [m]	X [%]	Y [%]
Punto 1	0,81	0,79	0,806	0,80	0,5	1,3
Punto 2	1,76	1,26	1,77	1,285	0,6	1,9
Punto 3	1,71	2,39	1,725	2,406	0,9	0,7
Punto 4	2,17	3,13	2,14	3,17	1,4	1,3
Punto 5	2,18	4,13	2,197	4,16	0,8	0,7
Punto 6	2,32	5,06	2,309	5,077	0,5	0,3
Punto 7	1,74	6,44	1,874	6,523	7,1	1,3
Punto 8	2,84	6,94	2,82	7,0	0,7	0,9
Punto 9	4,71	7,05	4,86	7,0	3,1	0,7

Se puede observar en la Fig. 24 la trayectoria real seguida por el robot en base a los puntos obtenidos prácticamente.

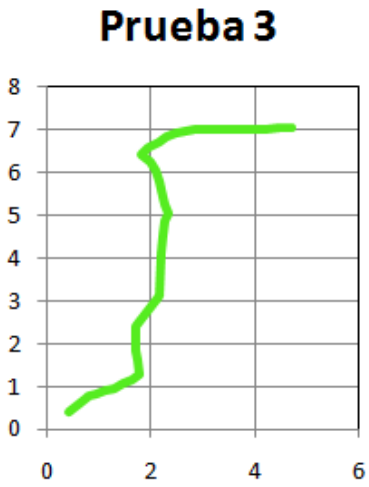

Fig 24. Trayectoria 3 en la práctica

El gráfico obtenido en Excel no cuenta con todos los puntos de la trayectoria seguida por el robot, es por esto que visualmente no presenta una similitud exacta en cuanto a la forma de la trayectoria, pero si muy parecida.

En cuanto a los errores se puede observar en la Tabla III que son pequeños la mayoría de ellos alrededor del 1%, solo en un punto se tiene un error de 7% pero éste no produce un error en los puntos posteriores de la trayectoria por lo que los resultados obtenidos en esta prueba se los considera satisfactorios. Para la justificación de errores, se debe mencionar que el laboratorio en la realidad no es perfectamente rectangular, por lo que desde un inicio ya se produce cierto error. Además de los errores mencionados anteriormente.

III. CONCLUSIONES

- La planeación y generación de trayectorias es una herramienta clave para la navegación autónoma de robots móviles debido a la necesidad de poseer una trayectoria óptima libre de obstáculos.

- Existen varios métodos para la planeación y generación de trayectorias por lo que es importante saber elegir que método se va a utilizar dependiendo de las características de la aplicación a desarrollar.

- Diagramas de Voronoi llega a ser uno de los mejores métodos de planeación de trayectorias debido a la seguridad que presenta la trayectoria al considerarse a esta lo más alejada de los obstáculos, así como presentar menos desventajas en comparación a otros métodos existentes.

- El toolkit LabVIEW Robotics Module proporcionó grandes facilidades en cuanto a la programación, puesto que posee una gran variedad de VI's que permitieron la creación del mapa de entorno así como la generación de una trayectoria óptima.

- El algoritmo de búsqueda A* cumple una función importante dentro de la generación de la trayectoria debido a que éste es el encargado de encontrar el camino de menor coste entre los nodos de partida y llegada.

- Robotino® es una plataforma educativa de alta calidad y plenamente funcional con accionamiento omnidireccional. Dispone de varios elementos como sensores analógicos, sensores binarios, cámara, encoders, un controlador de altas prestaciones, entre otros, que en conjunto proporcionan al sistema la necesaria "inteligencia" para hacer de Robotino® un robot autónomo.

- Dependiendo de la aplicación, el robot móvil Robotino® permite la programación de su hardware acorde a las necesidades del proyecto que se esté desarrollando

- La comunicación vía wireless presenta grandes ventajas respecto a otros tipos de comunicación para las HMI ya que elimina el medio físico de comunicación (cable). No obstante presenta desventajas, una de ellas es la interferencia que puede presentarse al recibir y enviar datos.

IV. RECOMENDACIONES

- Es importante ingresar con precisión las coordenadas de los obstáculos para de esta manera evitar desde ya errores iniciales, de igual manera, la colocación del robot en el punto de partida así como la orientación del mismo debe ser lo más precisa posible, ya que de lo contrario, este error ocasiona desvíos de la trayectoria original.

- Se recomienda que las baterías del Robotino® se encuentren 100% cargadas debido a que, de no estarlo, se podrían tener problemas y fallos en el funcionamiento de los motores lo que ocasionaría desvíos en la trayectoria.

- De tener problemas con la comunicación, se recomienda cambiar el número de puerto del *ImagePort*. Debe tenerse en cuenta que la red del Robotino® es una red pública sin clave de acceso, por lo que cualquier persona podría conectarse a la misma y posiblemente causar problemas en el funcionamiento del robot con la HMI.

- El programa podría ser adaptado para descargase en un robot más simple con la finalidad de no subutilizar las capacidades del Robotino®, debido a que lo único que se está controlando del robot móvil son los motores y los encoders, así como el sistema de comunicación inalámbrica.

- A pesar de que diagramas de Voronoi es uno de los métodos más completos, presenta ciertas desventajas, por lo que se podría optimizar la planeación y generación de trayectorias combinando a éste método con propiedades de otro método.

- El movimiento del robot se lo realiza solo en un sentido, en x , al ser de accionamiento omnidireccional se podría hacer que el movimiento sea en x y en y , de esta manera se podría dar una continuidad más precisa en el seguimiento del camino, pero a la vez la lógica de programación y obtención de las velocidades acorde a la trayectoria se vuelve más compleja.

V. AGRADECIMIENTOS

Al Laboratorio de Control y Sistemas por permitir el uso de sus instalaciones así como facilitar el robot Robotino® para el desarrollo del presente proyecto.

VI. REFERENCIAS

[1] A. Barrientos, L. Peñin y otros, "Fundamentos de Robótica," Segunda Edición, Editorial McGra-Hill. España, 2007
[2] F. Torres, J. Pomares, P. Gil, S. Puerta, R. Aracil, "Robots y Sistemas Sensoriales," Editorial Pearson Educación SA, Madrid 2002
[3] J. Lara, O. Loor, "Control de una plataforma robótica bípeda," EPN, Quito 2006
[4] Historia del arte de la Robótica, "Móviles o vehículos robot," Abril 2009, http://robotik-jjlg.blogspot.com/2009/04/moviles-o-vehiculos-robot.html
[5] A. Sánchez, C. López, "Diseño de un robot hexápodo, Hardware y Software de Control," Escola Universitària Politécnica de Vilanova i la Geltrú, 2005
[6] O. González, Bricogeek, "Hexapod Phoenix: El robot araña," Marzo 2008, http://blog.bricogeek.com/noticias/diy/video-hexapod-phoenix-el-robot-arana/
[7] Roboserv, "Robucar," 2010, http://www.roboserv.net/robucar
[8] CFIE de Valladolid, "Robots móviles: diseño," Febrero 2002, http://cfievalladolid2.net/tecno/cyr_01/robotica/movil.htm
[9] V. Muñoz, G. Gil, A. García, "Modelo Cinemático y Dinámico de un Robot Móvil Omnidireccional," Universidad de Málaga, 2003
[10] L. Gracia, "Modelado Cinemático y Control de Robots Móviles con ruedas," Universidad Politécnica de Valencia, 2006
[11] Manual Robotino®, FESTO, 2007
[12] M. Fernández, D. Fernández D, C. Valmaseda, "Planificación de trayectorias para un Robot Móvil," Universidad Complutense, Madrid 2009
[13] A. Ollero, "Planificación de trayectorias para Robots Móviles," Universidad de Málaga, 1995
[14] J. Latombe, "Robot Motion Planning," Kluwer Academic Pulishers, 1991
[15] E. Rodríguez, "Diagramas de Voronoi," Cinvestav-Tamaulipas, 2010
[16] J. Borja de los Santos, "Planificación de Trayectorias – El algoritmo PRM," España, 2007
[17] L. Ortega, "El Diagrama de Voronoi," Universidad de Jaen, España, 2010

[18] La Canción de Malapata, "El Diagrama de Voronoi," Febrero 2010, http://lacanciondemalapata.blogspot.com/2010/02/el-diagrama-de-voronoi-i.html
[19] Wikipedia, "Algoritmo de búsqueda A*," diciembre 2010, http://es.wikipedia.org/wiki/Algoritmo_de_b%C3%BAsqueda_A*
[20] M. De Verg, O. Cheong, M. Van Kreveld, M. Overmars, "Computational Geometry: Algorithms and Applications," Third Edition, Berlin, 2008

VII. BIOGRAFÍAS

Aracely Yandún, nació el 23 de Septiembre de 1986 en la ciudad de Quito – Ecuador, su educación primaria la realizó en la Escuela Ciudad de Cuenca en donde obtuvo la distinción de Abanderada del Pabellón Nacional. Sus estudios secundarios los realizó en el Colegio Experimental de Señoritas Simón Bolívar en el cuál se incorporó como Bachiller en Ciencias Básicas especialidad Físico - Matemático, obteniendo la mención de Portaestandarte del Colegio, además de ser reconocida por sus distinciones extracurriculares en concursos de ciencias exactas y ortografía. Obtiene su título en Ingeniera en Electrónica y Control en la Escuela Politécnica Nacional en el año 2011. Fue residente de obra de las instalaciones electrónicas del nuevo Hospital San Francisco de Quito del IESS. Actualmente trabaja como Ingeniera de obra en Microcircuits Cía. Ltda.
Áreas de interés: Control industrial y automatización, Domótica, Robótica, Ecología y Medio Ambiente.
(aracely.yandun@yahoo.com)

Nelson Sotomayor, nació en Quito-Ecuador el 9 de Septiembre de 1971. Realizó sus estudios secundarios en el Instituto Nacional Mejía. Se graduó en la Escuela Politécnica Nacional como Ingeniero en Electrónica y Control en 1999. Obtuvo su título de Magíster en Ingeniería industrial en junio del 2006 en la Escuela Politécnica Nacional. En septiembre del 2008 como becario del Gobierno de México y la Agencia de Cooperación Internacional del Japón (JICA), participó en el IV Curso Internacional de Robótica Aplicada, en el Centro Nacional de Actualización Docente CNAD ubicado en México DF. Actualmente desempeña el cargo de Profesor Principal T/C en el Departamento de Automatización y Control Industrial de la Escuela Politécnica Nacional. Adicionalmente es Jefe del Departamento de Automatización y Control Industrial.
Áreas de interés: robótica móvil, informática y redes, microcontroladores, automatización y control industrial.
(nelson.sotomayor@epn.edu.ec)