UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

NAVEGACIÓN AUTÓNOMA DE UN ROBOT MÓVIL

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO ELÉCTRICO ELECTRÓNICO

PRESENTA:

GUILLERMO HERRERA RAMÍREZ

DIRECTORA DE TESIS:

ING. GLORIA MATA HERNÁNDEZ

MÉXICO D.F., MAYO 2014

DEDICATORIA

A Dios por el aprendizaje del día a día, las alegrías vividas y por darme la fuerza necesaria para enfrentar las adversidades.

A mis padres Benigna Ramírez Pardo y Celedonio Herrera Avilés, por el apoyo brindado a lo largo de todo este camino, por estar conmigo en mis éxitos y en mis fracasos, por su paciencia, cariño y entrega. Han sido un ejemplo de vida para mí.

A Claudia, por permitirme estar a su lado, por apoyarme en los momentos difíciles y darme la chispa para seguir adelante, por compartir sueños y aventuras, por su amor, paciencia y por creer en mí.

A mis hermanas Celina y Diana, por su compañía, por su apoyo, por las vivencias y experiencias juntos.

A mis sobrinos Sebastián y Camila por la gran alegría que trajeron a la familia.

A mi primo Eduardo por ser un hermano y compañero de aventuras.

A Güero, Mini y todos ellos, siempre nobles y fieles.

¡Lo logramos!

AGRADECIMIENTOS

A la Universidad Nacional Autónoma de México por permitirme tener el honor de formarme dentro de sus aulas y por alimentar mi sed de conocimiento.

A la Ing. Gloria Mata por la oportunidad de presentar este trabajo en varios lugares, por el apoyo y las facilidades brindadas a lo largo de la elaboración de este proyecto.

A los profesores que a lo largo de la carrera me ayudaron en mi formación académica.

A todos mis amigos a lo largo de la carrera por todos esos grandes momentos vividos en la facultad.

No siempre puedes conseguir lo que quieres, pero si lo intentas podrías encontrar lo que necesitas

(The Rolling Stones)

Contenido

Justificación

Los robots en la actualidad son parte fundamental en las actividades de la vida moderna del ser humano, desde procesos de manufactura (que hacen posibles la fabricación de teléfonos inteligentes, computadoras portátiles y autos modernos), hasta actividades de investigación espacial en otro planeta, pasando por aplicaciones en la medicina e incluso en el entretenimiento; los robots están presentes directa o indirectamente en la vida de las personas

Debido a ello, la investigación en robótica es de especial interés ya que permite obtener tecnología útil para resolver problemas o hacer más eficientes algunos procesos. De igual manera, el desarrollo de la robótica tiene impacto en el crecimiento tecnológico de un país; países avanzados en este campo como Alemania ó Japón exportan este tipo de tecnología a otros países debido a su utilidad así, en el caso de México donde aún falta seguir invirtiendo en investigación en este ámbito, podría permitir en un futuro una dependencia menor a la importación de soluciones tecnológicas al poder desarrollar tecnología propia.

La investigación asociada a la disciplina de robots, y de manera particular a los móviles, han adquirido gran relevancia; y aun cuando ha habido importantes avances, sigue siendo un tema de investigación y gran interés. Las aplicaciones orientadas al uso de robots móviles involucran aspectos que van desde el diseño mecánico hasta su control, maniobrabilidad, ubicación, interacción y sincronización con otros dispositivos similares.

Este contexto motivó a realizar una aplicación en robótica móvil para contribuir al desarrollo en la investigación de esta área. En el caso particular de este trabajo, para lograr la navegación autónoma de un robot móvil diferencial se presenta un esquema que permite cumplir con el objetivo propuesto, el cual puede servir de referencia para otro tipo de robots móviles e incluso para ser aplicado a una escala mayor, por ejemplo: un robot móvil que se desplace en todo un edificio o una planta de manufactura, o para la robotización o navegación autónoma de automóviles.

Objetivo

Implementar un sistema que permita a un robot móvil con direccionamiento diferencial navegar de forma autónoma en un entorno o espacio de trabajo con obstáculos, de manera tal que el robot pueda recorrer la ruta más corta posible entre una posición inicial y una final. La posición y orientación iniciales al igual que la posición de meta o final son conocidas por el robot. El espacio o entorno que se debe atravesar es un área con obstáculos que forman diferentes posibles caminos hasta la meta.

Capítulo 1

Introducción a la robótica

1.1 Antecedentes históricos.

La idea de crear una máquina que pueda desempeñar tareas útiles en beneficio de las actividades del hombre siempre ha estado en la imaginación del ser humano, a lo largo de la historia de la humanidad ha habido varios inventos que han tratado de lograr esto de una u otra manera.

El desarrollo en la tecnología a partir de la Edad Moderna hizo posible lo que eventualmente se denominó como robot (Figura 1-1 y Figura 1-2), antes de existir esta palabra se denominaba como autómata a las máquinas automáticas puramente mecánicas.

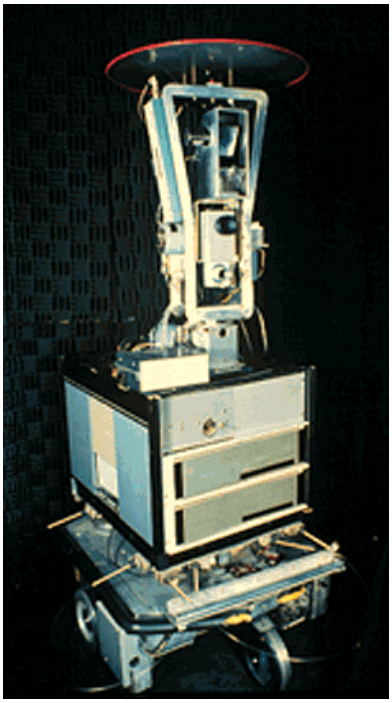


Figura 1-1. Robot Shakey, uno de los primeros robots móviles autónomos

El termino robot fue usado por primera vez en la obra de teatro R.U.R (Rossum´s Universal Robots) escrita por el dramaturgo checo Karel Capek, estrenada en 1921, el termino robot deriva de la palabra *robota* que en checo y muchas lenguas eslavas significa trabajo o labor (figuradamente trabajo duro). Al extender el uso de la palabra robot surgió la palabra *robótica* para describir este campo de estudio, este término fue acuñado por el escritor de ciencia ficción Isaac Asimov.

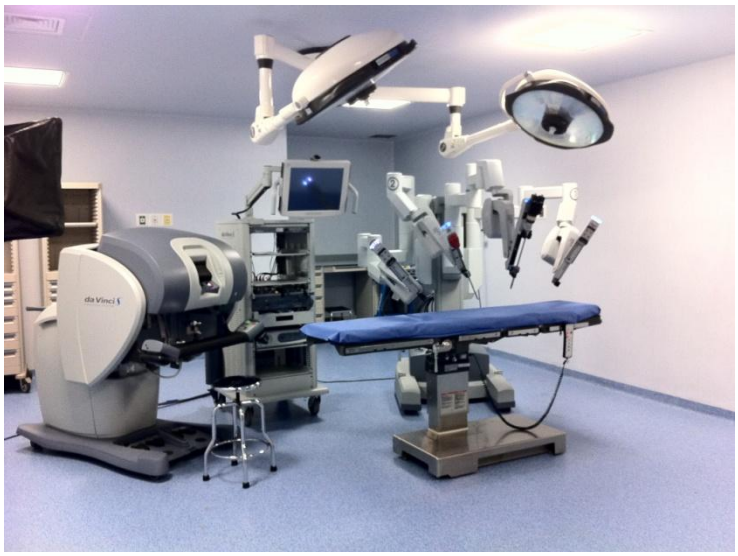


Figura 1-2. Robot da Vinci, un robot quirúrgico que permite cirugías menos invasivas

1.2 Clasificación de robots

El éxito de la robótica se debe principalmente a la gran variedad de tareas que los robots pueden realizar, por otro lado la movilidad que tiene un robot influye directamente en estas tareas. Desde el punto de vista del estudio en ingeniería de la movilidad de los robots, estos pueden clasificarse (como se observa en la Figura 1-3) en dos grupos: los brazos robóticos ó manipuladores y los robots móviles.

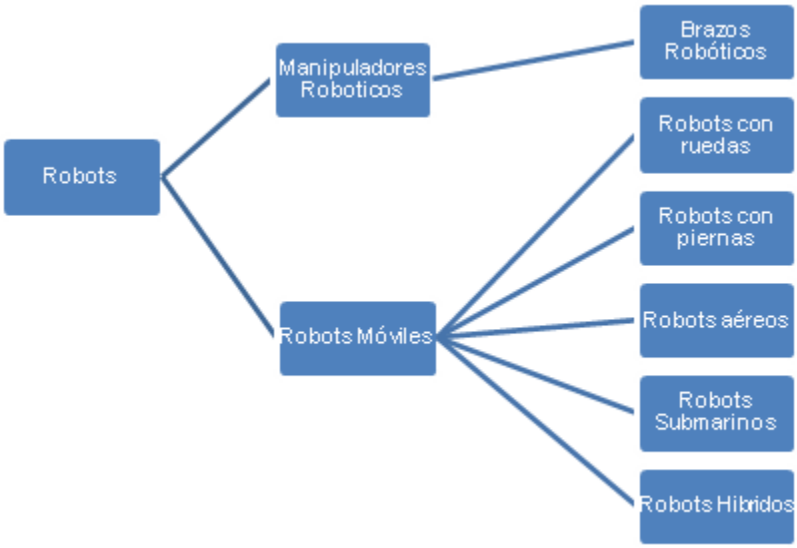


Figura 1-3. Clasificación de robots según su movilidad

1.2.1 Robots manipuladores

De acuerdo con la de definición del *Robot Institute of America*, un robot industrial es un manipulador programable multifuncional diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos variados, programados

para la ejecución de distintas tareas. Esta definición se debe al hecho de que la mayoría de los robots industriales (Figura 1-4) son esencialmente brazos articulados.

Figura 1-4. Brazo robótico. Este tipo de robots se usan muy a menudo en la industria automotriz

El éxito de los brazos robóticos o manipuladores se debe a que pueden realizar tareas repetitivas (como por ejemplo dentro de una línea de producción) con una rapidez y exactitud por encima de las capacidades humanas. Estas tareas pueden ser: pintado, soldado superficial de componentes electrónicos, soldado de piezas automotrices (ver Figura 1-5), etc.

Figura 1-5. Manipulador robótico

A pesar del éxito que gozan los robots manipuladores en la industria estos tienen la desventaja de que su estructura mecánica esta fija por uno de sus extremos lo cual los confina a una zona de trabajo en particular sin poder trabajar y moverse fuera de dicha zona.

1.2.2 Robots móviles

A diferencia de los robots manipuladores, los robots móviles (ver Figura 1-6) pueden moverse de un lugar a otro dentro de un área de trabajo para la cual están diseñados, lo cual le permite al robot poder realizar tareas en diferentes ubicaciones, esta movilidad a su vez incrementa la autonomía del robot al permitirle moverse por sí mismo.

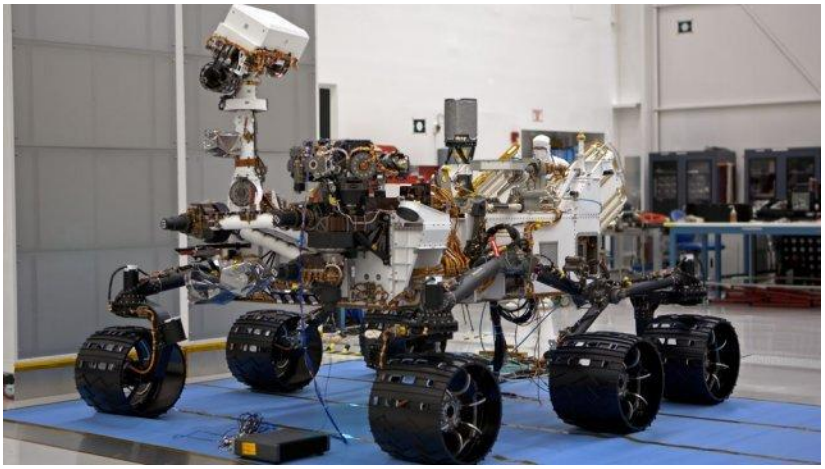


Figura 1-6. El robot Curiosity diseñado para explorar la superficie de Marte es un ejemplo de un robot móvil

La gracia de los robots móviles radica en la autonomía de movimiento que poseen, para lograr dicha autonomía un robot debe tener la capacidad de reconocer el entorno en el que se encuentra a través de la correcta interpretación de la información recibida a través de los sensores con los que cuenta, así mismo el robot móvil debe poder convertir dicha información en instrucciones que serán ejecutadas por los actuadores para lograr la correcta realización de su tarea específica.

1.3 Tipos de locomoción de robots móviles

El primer reto que existe en el diseño de un robot móvil es el mecanismo de locomoción que le permitirá moverse de un lugar a otro. Existe una gran variedad de formas de lograr dicho movimiento, hay robots que pueden caminar, brincar, correr, deslizarse, nadar, volar, y rodar usando ruedas. La mayoría de estas formas de locomoción son inspiradas en la forma que se mueven los seres vivos, la única excepción es la rueda actuada la cual es un invento del hombre que logra una alta eficiencia en superficies lisas y duras. Cabe señalar que la locomoción en un robot puede verse como complemento de la manipulación.

1.3.1 Robots móviles con piernas

Este tipo de locomoción se caracteriza por tener un conjunto de puntos discretos de contacto entre el robot y el suelo, esto hace que la calidad del suelo entre estos puntos de

contacto no sea de interés, así mismo permite que el robot que camina sea capaz de cruzar por ejemplo un agujero, por lo que la principal ventaja de este tipo de locomoción es la maniobrabilidad en terrenos muy irregulares. Las principales desventajas de este tipo de robots son: una mayor complejidad mecánica y un mayor consumo de energía.

La configuración de seis piernas es la más popular de estos robots debido a la estabilidad que presentan al caminar, además de un control menos complejo en comparación a configuraciones con menos piernas. Aún así cada vez es más común el desarrollo de sistemas de dos piernas, estos robots intentan imitar el sistema de locomoción del hombre (ver Figura 1-7), uno de los principales desafíos de estos robots es el balance de la estructura al permanecer de pie y al caminar, además de que cada pierna debe tener la capacidad de soportar el peso completo del robot, todo esto hace necesario la implementación de sistemas de control complejos.

Figura 1-7. El robot Asimo es un robot bípedo que emula el sistema de locomoción humana

1.3.2 Robots móviles con ruedas

Las ruedas son el mecanismo de locomoción más usado en los robots móviles y el más eficiente en terrenos con superficies duras y planas, además el balance de la estructura no suele ser un problema en el diseño de estos robots ya que normalmente están diseñados para mantener todas sus ruedas en contacto con el suelo (ver Figura 1-8), tres ruedas son suficientes para garantizar un balance estable. Debido a que el balance no es un problema en este caso, el diseño de los robots con ruedas puede enfocarse en las tareas de maniobrabilidad y control. En caso de tener menos de tres ruedas o de que alguna de las ruedas no esté en contacto con el suelo el balance es un problema a resolver. Entre sus desventajas están el deslizamiento en las ruedas debido al movimiento, y el hecho de que la locomoción por ruedas es poco eficiente en superficies blandas.

Figura 1-8. El Robot DaNI (1.0) es un robot diseñado para investigación en robótica móvil

1.3.3 Robots móviles submarinos y aéreos

Este tipo de robots surgen de la evolución de vehículos piloteados o teleoperados por el hombre tales como helicópteros, submarinos (Figura 1-9), y pequeños aviones, normalmente son usados para tareas de inspección, investigación y toma de datos en lugares remotos o de difícil acceso.

Figura 1-9. Serafina [13], es un robot submarino diseñado para recabar datos en el océano

1.4 Tipos de ruedas y configuraciones de robots móviles con ruedas

La elección del tipo de ruedas para un robot móvil está fuertemente ligada al arreglo o geometría de las ruedas en el robot dando diferentes tipos de configuraciones de robots móviles con ruedas. Existen cuatro tipos básicos de ruedas usadas en los robots móviles (Figura 1-10), las cuales difieren ampliamente en su cinemática, y tienen un impacto directo en la cinemática total del robot móvil.

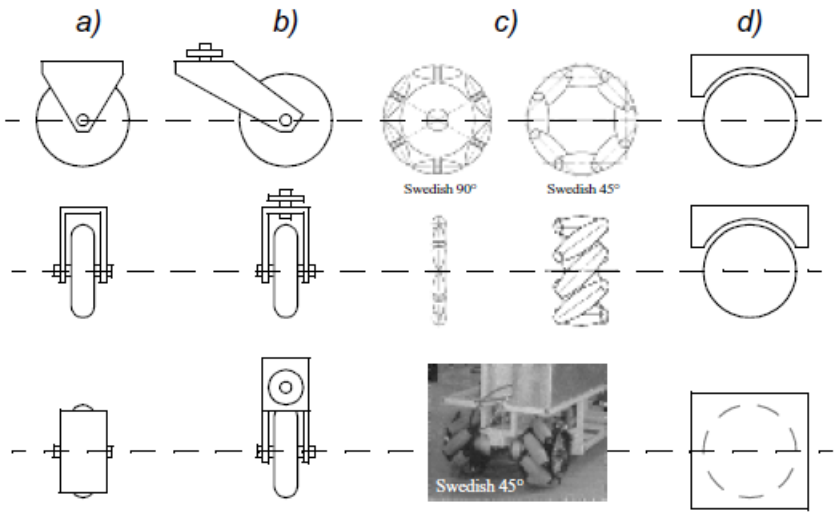


Figura 1-10. Los cuatro tipos básicos de ruedas

a) Rueda Estándar: Tiene dos grados de libertad; rotación alrededor del eje (motorizado) de rotación, y del punto de contacto.

b) Rueda Castor: Tiene dos grados de libertad; rotación alrededor de un punto articulado de que le da dirección con un offset.

c) Rueda Sueca: Tiene tres grados de libertad; rotación alrededor del eje (motorizado) de rotación, alrededor de los barriles de rodamiento, y alrededor del punto de contacto. Existen dos tipos de estas ruedas ambas con barriles de rodamiento a 45° o a 90°.

d) Bola o Rueda Esférica: Es omnidireccional, su realización es técnicamente difícil.

A continuación se muestran algunas de las configuraciones existentes de robots móviles con ruedas.

1.4.1 Configuración triciclo clásico

Tiene una rueda delantera actuada que permite la tracción y además sirve para el direccionamiento, y dos ruedas pasivas en el eje trasero. El ángulo de dirección de la rueda delantera (o su velocidad angular) y su velocidad de giro (velocidad lineal) son las variables de control utilizadas en este caso. El centro del sistema de referencia local está en el centro del eje trasero (Figura 1-11).

Esta configuración tiene una maniobrabilidad muy buena pero tiende a presentar problemas de inestabilidad en terrenos irregulares, el centro de gravedad tiende a desplazarse cuando se desplaza por una pendiente. Gracias a su simplicidad es comúnmente usado en vehículos para exteriores e interiores pavimentados.

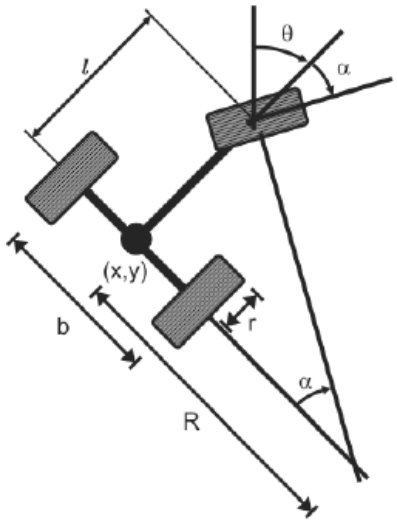


Figura 1-11. Configuración Triciclo Clásico

1.4.2 Configuración diferencial

Cuenta con dos ruedas actuadas independientemente por dos motores (diferencia de velocidades), y una rueda de dirección no actuada al frente o atrás del robot, esta rueda además sirve de apoyo al robot dándole estabilidad, la tracción se consigue con las dos ruedas actuadas. En este caso el centro del sistema de referencia local se ubica en el punto medio del eje de las ruedas traseras (Figura 1-12), las variables de control son las velocidades de las dos ruedas actuadas.

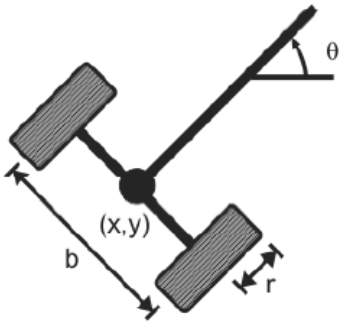


Figura 1-12. Configuración Diferencial

Cabe señalar que el robot usando en el presente trabajo (Robot DaNI versión 2.0) es de tipo diferencial, con las ruedas de tracción en la parte delantera y una rueda tipo sueca 90° en la parte trasera para la dirección del robot (Figura 1-13).

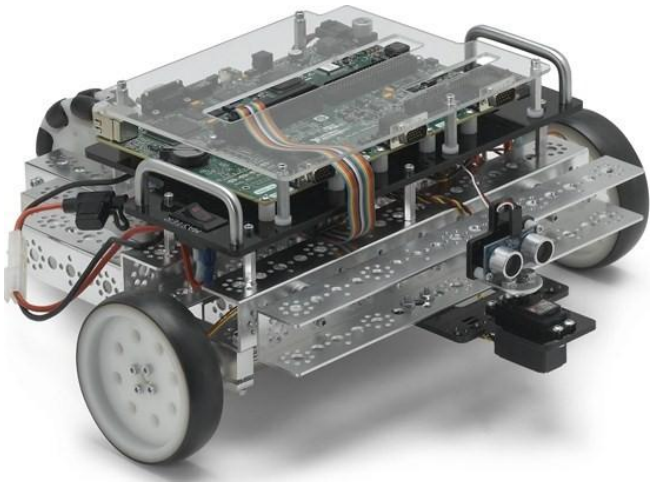


Figura 1-13. Robot DaNI versión 2.0

1.4.3 Configuración ackerman

Esta es la configuración utilizada en los automóviles, tiene dos ruedas actuadas al frente o atrás dependiendo del tipo de tracción (delantera o trasera), y un eje pasivo con dos ruedas (Figura 1-14). El eje delantero sirve para direccionar el vehículo, el direccionamiento de las ruedas debe ser ligeramente diferente entre la ruedas para evitar el deslizamiento de las mismas.

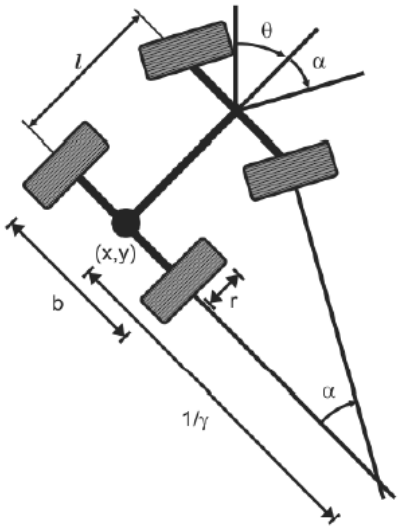


Figura 1-14. Configuración Ackerman

En realidad pocos robots usan esta configuración debido a su pobre maniobrabilidad, a excepción de los robots móviles hechos para ser usados en sistemas de caminos para automóviles, aun así esta configuración tiene la ventaja de tener una muy buena estabilidad lateral en las vueltas a gran velocidad.

1.4.4 Configuración síncrona

En esta configuración se tienen 3 ruedas actuadas y dirigidas por dos motores, uno de traslación que acelera las tres ruedas al mismo tiempo para lograr un desplazamiento lineal v, y otro para la dirección que gira las tres ruedas con una velocidad angular ω alrededor de sus ejes verticales. Se usa una transmisión de coronas de engranajes con correas para transmitir el movimiento de los dos motores a las tres ruedas al mismo tiempo (Figura 1-15). Una ventaja de esta configuración es la omnidireccionalidad que se logra con el mecanismo usado para la dirección y traslación, y una de sus desventajas es que la orientación no es directamente controlable. En esta configuración se tienen dos grados de libertad y una restricción no holonómica (véase tema 2.1).

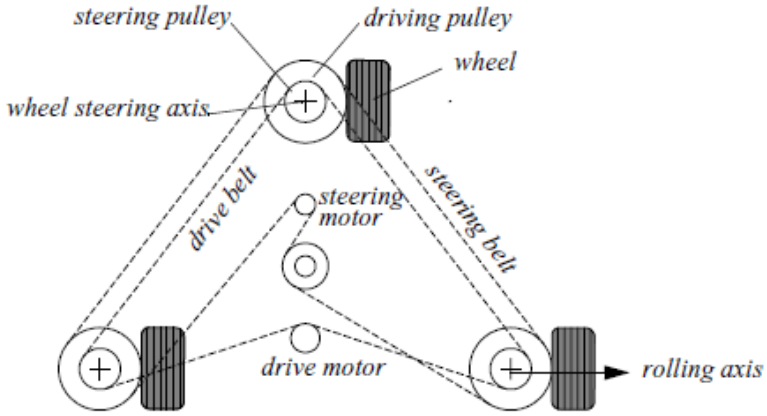


Figura 1-15. Configuración Síncrona

1.4.5 Configuración omnidireccional

Esta configuración permite una maniobrabilidad completa, haciendo que el robot sea capaz de moverse en cualquier dirección (x, y, θ) por lo que se dice que tiene tres grados de libertad, se pueden utilizar ruedas castor, esféricas o suecas para su realización, los robots omnidireccionales (Figura 1-16) son de tipo holonómico.

Figura 1-16. El robot Uranus [18] es un robot omnidireccional con cuatro ruedas suecas 45° actuadas

1.4.6 Tracción por pistas de deslizamiento y patinamiento

Para las configuraciones de ruedas descritas anteriormente se asume que las ruedas no pueden deslizarse o patinar sobre las superficies. Una alternativa para el direccionamiento es la que utiliza pistas de deslizamiento y patinamiento (slip/skid), la cual puede ser usada para reorientar al robot al mover las ruedas en sentidos opuestos y a diferentes velocidades. Un ejemplo de vehículo que usa este principio de direccionamiento es el tanque de guerra. Los robots que hacen uso de este mecanismo (Figura 1-17) tienen una muy buena maniobrabilidad, sus desventajas son la dificultad para predecir la ubicación exacta del centro de rotación, y el cambio exacto en la orientación y posición están sujetos a la variación de la fricción con el suelo, por lo que este tipo de tracción resulta eficiente en terreno suelto pero ineficiente en el caso contrario.

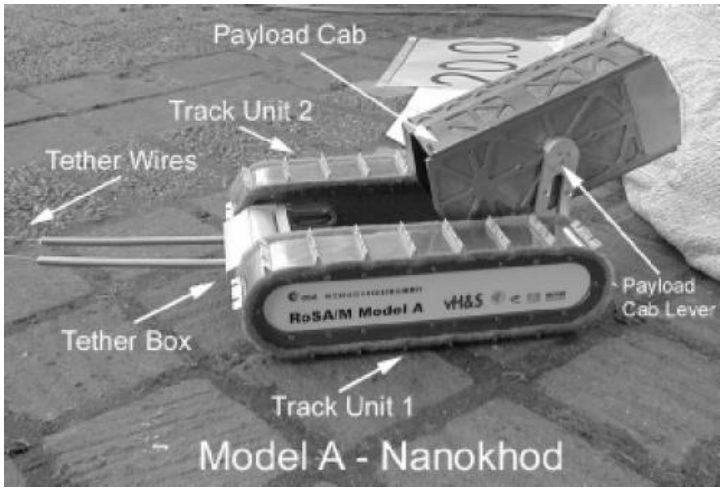


Figura 1-17. Robot con pistas de deslizamiento y patinamiento

Capítulo 2

Marco teórico del movimiento del robot móvil

En este capítulo se obtienen los modelos que describen el movimiento del robot, los cuales son utilizados para la localización y el control del movimiento del mismo.

En primera instancia se adoptan las siguientes hipótesis que simplifican el desarrollo matemático:

- El robot se mueve sobre una superficie plana.
- Los ejes de guiado son perpendiculares al suelo.
- Se supone que las ruedas se mueven con rodadura pura; es decir no existe deslizamiento.
- El robot es visto como un sólido rígido (no tiene partes flexibles).
- Durante un periodo de tiempo lo suficientemente pequeño en el que se mantiene constante la consigna de dirección, el vehículo se moverá de un punto al siguiente a lo largo de un arco de circunferencia.

2.1 Restricciones cinemáticas

Sea un vector $p = [p_1 \quad \cdots \quad p_r]^T$ (donde la transpuesta T indica que es en forma vertical) el cual contiene las variables necesarias para determinar en su totalidad la posición y orientación del robot (sistema físico). En general, es posible formular restricciones de movimiento del tipo:

$$G_k(p, p', t) = 0; \qquad k = 1, \dots, s; \tag{2.1}$$

esto quiere decir que se deben cumplir s restricciones en las variables de p, en sus derivadas (respecto al tiempo) p', y en el tiempo, como las restricciones dependen de p' entonces dependen de la velocidad. Las restricciones que dependen de la velocidad se llaman *restricciones no holónomas*, y tienen la forma (2.1).

Si en las restricciones no intervienen las velocidades se dice que son *holónomas* y tienen la forma:

$$G_k(p, t) = 0; \qquad k = 1, \dots, s; \tag{2.2}$$

Adicionalmente, para que una restricción sea no holónoma es necesario que no sea integrable; esto quiere decir, que no se pueda deducir por derivación total con respecto al tiempo de una restricción holónoma; debido a esto el análisis cinemático, dinámico y de control de vehículos no holónomos es más complejo. En sistemas con restricciones no holónomas, el número de coordenadas que describe su configuración siempre es más grande que el número de grados de libertad que poseen.

Por ejemplo, si se considera el movimiento de la rueda de radio r en una sola dirección (Figura 2-1), la variable del actuador (motor) podría ser el ángulo de giro θ de la rueda, y la variable en el espacio cartesiano el desplazamiento x.

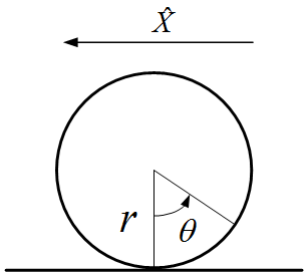


Figura 2-1. Movimiento de la rueda en una sola dimensión

Las dos variables se relacionan con la condición de rodadura:

$$x' = r\theta' \tag{2.3}$$

que depende de las velocidades, pero puede deducirse por derivación de la restricción holónoma:

$$x - r\theta = constante \tag{2.4}$$

Entonces en este caso no existen restricciones no holónomas. Además se puede observar que es necesaria una sola coordenada x ó θ porque existe un solo grado de libertad.

Sin embargo, si la rueda se mueve en un plano (Figura 2-2), aparecen restricciones no holónomas. En este caso, para especificar la posición y orientación de la rueda se puede utilizar cuatro variables: las coordenadas (x, y) del punto de contacto de la rueda y el plano, el ángulo θ entre la vertical y un radio de circunferencia como referencia (este ángulo indica cuánto ha girado la rueda), y por último un ángulo de orientación $\emptyset$.

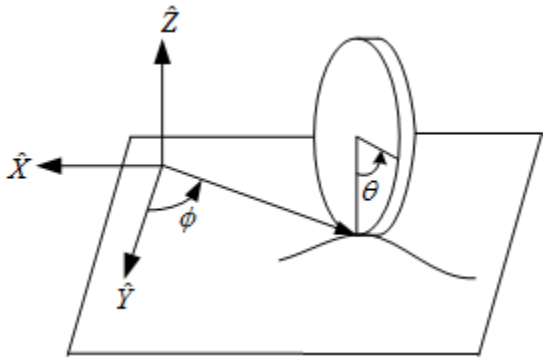


Figura 2-2. Movimiento de la rueda en el plano

Entonces la condición de rodadura pura sin deslizamiento introduce dos restricciones, debido a que la distancia recorrida por el punto de contacto de la rueda con el plano es la misma que la que recorre sobre el borde de la rueda. Entonces si se proyecta sobre los ejes $x - y$, la velocidad del punto de contacto de la rueda con el plano, se tendría:

$$-x' \sin \emptyset + y' \cos \emptyset = \theta' r \tag{2.5}$$

$$x' \cos \emptyset + y' \sin \emptyset = 0 \tag{2.6}$$

Las restricciones (2.5) y (2.6) no son integrables, por lo tanto no se pueden obtener relaciones funcionales entre las variables $(x, y, \theta, \emptyset)$ a partir de las ecuaciones de restricción. Ya que no existen relaciones funcionales entre las variables es posible mover la rueda desde una posición inicial $(x_0,\ y_0,\ \theta_0,\ \emptyset_0)$ hasta una posición final $(x_f,\ y_f,\ \theta_f,\ \emptyset_f)$ rodándola y girándola sobre su eje vertical. Sin embargo, la dirección del movimiento está restringida por las ecuaciones (2.5) y (2.6), por lo tanto el camino entre la posición inicial y final no puede ser cualquiera.

Según las restricciones de movimiento presentes en los robots móviles, estos se pueden clasificar dependiendo de su movilidad en: holonómicos y no holonómicos Los robots holonómicos tienen la capacidad de cambiar de posición en cualquier dirección, los robots no holonómicos son aquellos en los cuales su movimiento se logra solo por dos desplazamientos (ver Figura 2-3), además de que el vehículo posee sólo dos grados de libertad. Como es de esperarse el tipo de ruedas (descritas en el tema 1.4) de un robot móvil, tiene un impacto directo en la cinemática del robot.

En la Figura 2-3 se muestra un robot tipo diferencial (como el usado en este trabajo) el cual es un ejemplo de robot no holonómico, este tipo de robot se puede mover hacia adelante o hacia atrás pero no lateralmente debido al deslizamiento de las ruedas.

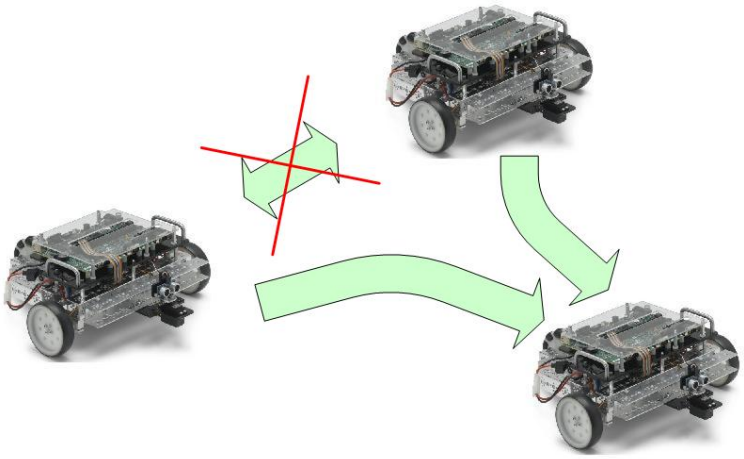


Figura 2-3. Movimiento de un robot no holonómico

2.2 Modelo cinemático del robot móvil

Considérese un sistema de *referencia global* $\{G\}$ y un *sistema local* $\{L\}$ con origen en el punto de guiado del robot, el cual se ubica en el punto medio de la vía del vehículo (distancia que separa las dos ruedas), cuyo eje $\hat{Y}_L$ es coincidente con la dirección del eje longitudinal del vehículo como se muestra en la Figura 2-4.

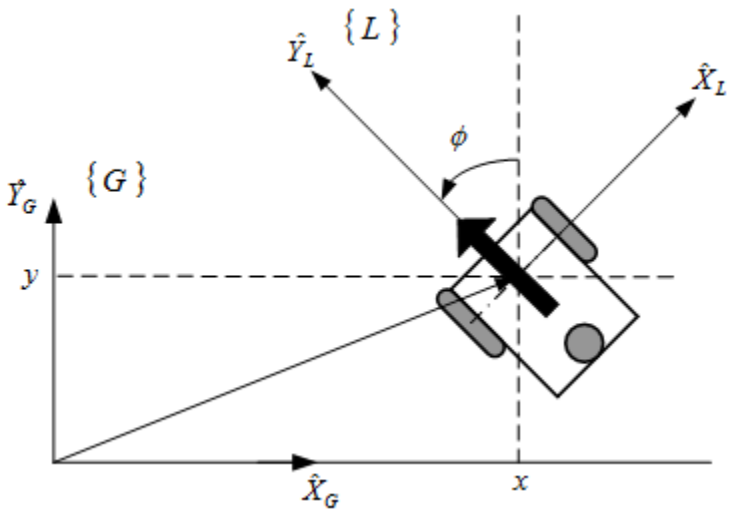


Figura 2-4. Sistemas de referencia presentes en la navegación de un robot móvil

Supóngase que el vehículo se desplaza según una longitud de arco de circunferencia Δs de un punto *A* a un punto *B* como se observa en la Figura 2-5, en un intervalo de tiempo Δt muy pequeño.

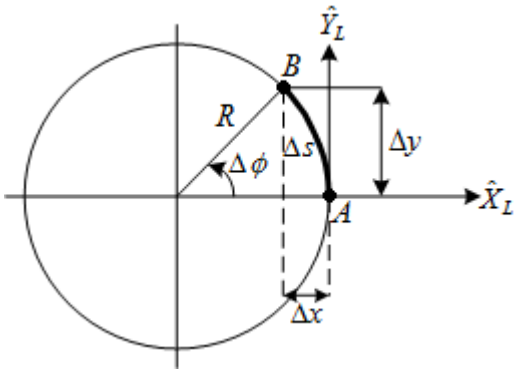


Figura 2-5. Desplazamiento del robot según un arco de circunferencia

La velocidad lineal (avance hacia adelante o hacia atrás) del vehículo está dada por:

$$v = \frac{\Delta s}{\Delta t} \tag{2.7}$$

y la velocidad angular (giro del vehículo) por:

$$\omega = \frac{\Delta \emptyset}{\Delta t} \tag{2.8}$$

donde $\Delta \emptyset$ es el cambio de la orientación del robot durante el intervalo de tiempo Δt.

La longitud Δs del arco recorrido se puede expresar de la siguiente forma:

$$\Delta s = R \Delta \emptyset \tag{2.9}$$

donde R es el radio de giro o radio de la circunferencia que describe el punto de guiado.

La curvatura se puede definir como la inversa del radio de giro:

$$\gamma = \frac{1}{R} = \frac{\Delta \emptyset}{\Delta s} \tag{2.10}$$

Las ecuaciones de movimiento en el sistema $\{L\}$ (ver Figura 2-4) son:

$$^{L}(\Delta x) = -(R - R\cos(\Delta \emptyset)) \tag{2.11}$$

$$^{L}(\Delta y) = R\,\mathrm{sen}(\Delta \emptyset) \tag{2.12}$$

Para expresar estas ecuaciones de movimiento en el sistema global $\{G\}$ es necesario mapearlas utilizando una matriz de rotación ${}^{G}_{L}R$.

$$ {}^{G}_{L}R = [\,{}^{G}\hat{X}_L \quad {}^{G}\hat{Y}_L] = \begin{bmatrix} \hat{X}_L \cdot \hat{X}_G & \hat{Y}_L \cdot \hat{X}_G \\ \hat{X}_L \cdot \hat{Y}_G & \hat{Y}_L \cdot \hat{Y}_G \end{bmatrix} = \begin{bmatrix} cos\emptyset & -sen\emptyset \\ sen\emptyset & cos\emptyset \end{bmatrix} \tag{2.13} $$

Donde $\emptyset$ es el ángulo entre los sistemas $\{G\}$ y $\{L\}$ o dicho de otra forma; el ángulo de orientación del vehículo.

Utilizando la matriz de rotación ${}^{G}_{L}R$ y las ecuaciones de movimiento se tiene:

$$ \begin{bmatrix} {}^{G}(\Delta x) \\ {}^{G}(\Delta y) \end{bmatrix} = {}^{G}_{L}R \begin{bmatrix} {}^{L}(\Delta x) \\ {}^{L}(\Delta x) \end{bmatrix} = \begin{bmatrix} cos\emptyset & -sen\emptyset \\ sen\emptyset & cos\emptyset \end{bmatrix} \begin{bmatrix} {}^{L}(\Delta x) \\ {}^{L}(\Delta y) \end{bmatrix} $$

$$ \begin{aligned} \Delta x = {}^{G}(\Delta x) &= R\cos(\emptyset)\,[\cos(\Delta\emptyset) - 1] - Rsen(\Delta\emptyset)sen(\emptyset) \\ \Delta y = {}^{G}(\Delta y) &= R\,\mathrm{sen}(\emptyset)\,[\cos(\Delta\emptyset) - 1] + Rsen(\Delta\emptyset)cos(\emptyset) \end{aligned} \tag{2.14} $$

Tomando en cuenta que el intervalo de tiempo Δt es muy pequeño, entonces el incremento en la orientación $\Delta\emptyset$ también lo será, debido a esto se pueden hacer las siguientes aproximaciones.

$$ \begin{aligned} \cos(\Delta\emptyset) &\cong 1 \\ \mathrm{sen}(\Delta\emptyset) &\cong \Delta\emptyset \end{aligned} \tag{2.15} $$

Tomando en cuenta estas aproximaciones en las ecuaciones (2.14) se obtiene:

$$ \begin{aligned} \Delta x &= -R\Delta\emptyset\; sen(\emptyset) \\ \Delta y &= R\Delta\emptyset\cos(\emptyset) \end{aligned} \tag{2.16} $$

además si se sustituye (2.9) se tiene que:

$$ \begin{aligned} \Delta x &= -\Delta s\; sen(\emptyset) \\ \Delta y &= \Delta s\cos(\emptyset) \end{aligned} \tag{2.17} $$

Si ambas ecuaciones se dividen por Δt, se toma en cuenta (2.7) y Δt se hace tender a cero se obtiene:

$$ x' = -v\; sen\emptyset \tag{2.18} $$

$$y' = v\,cos\emptyset \tag{2.19}$$

de igual forma si Δt tiende a cero en la ecuación (2.8) se tendría que:

$$\emptyset' = \omega \tag{2.20}$$

Las ecuaciones (2.18), (2.19) y (2.20) constituyen el modelo cinemático del robot móvil.

2.3 Modelo jacobiano

En adelante se considerará que las variables p se expresan en el sistema global de referencia $\{G\}$. Sea p un vector con las n variables suficientes para determinar la posición y orientación del robot y q un vector con m variables de actuación, donde $n > m$. Y sean p' y q' las derivadas de estos vectores, entonces el vector p' se puede expresar como:

$$p' = J(p)q' \tag{2.21}$$

donde $J(p)$ es el jacobiano. Utilizando una forma diferente para expresar al jacobiano esta ecuación se puede expresar de la siguiente manera:

$$p' = f(p) + \sum_{i=1}^{m} g(p)_i q'_i \tag{2.22}$$

donde f y g son funciones vectoriales analíticas.

En este caso se tiene que $p = [x \quad y \quad \emptyset]^T$, este vector contiene las coordenadas del punto guía y la orientación del robot, entonces el modelo cinemático del robot puede expresarse en la forma (2.22) con $f(p) = 0$.

$$p' = \begin{bmatrix} x' \\ y' \\ \emptyset' \end{bmatrix} = \begin{bmatrix} -sen\emptyset \\ cos\emptyset \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{2.23}$$

donde v es la velocidad lineal y ω la angular del vehículo.

Las ecuaciones matriciales (2.23) pueden expresarse también en la forma del modelo (2.21):

$$\begin{bmatrix} x' \\ y' \\ \emptyset' \end{bmatrix} = \begin{bmatrix} -sen\emptyset & 0 \\ cos\emptyset & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2.24}$$

donde $q' = [v \quad \omega]^T$ es el vector con las variables de actuación que son las variables de entrada. La ecuación (2.24) corresponde al *modelo cinematico directo*.

Si se combinan las primeras dos ecuaciones de (2.24) se obtiene una ecuación independiente de v.

$$x' cos\emptyset + y' sen\emptyset = 0 \tag{2.25}$$

esta ecuación es la restricción no holónoma de movimiento (2.6) según la cual el vehículo debe moverse en cada instante según la dirección de su *eje longitudinal de simetría*.

$$\tan \emptyset = -\frac{x'}{y'} \tag{2.26}$$

Tomando en cuenta esto último se observa que la posición (x, y) y la orientación $\emptyset$ del robot no son independientes.

En el caso del modelo inverso es necesaria la inversa del jacobiano. Debido a que el jacobiano no es cuadrado, es necesario emplear la matriz seudoinversa. Entonces si se multiplica ambos miembros de (2.21) por J^T y se despeja q' se obtiene:

$$q' = \{[J(p)]^T J(p)\}^{-1} [J(p)]^T p' \tag{2.27}$$

Así, para el modelo (2.24) se obtiene el modelo cinemático inverso:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} -sen\emptyset & cos\emptyset & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ \emptyset' \end{bmatrix} \tag{2.28}$$

de donde se puede obtener:

$$v = -x' \sin \emptyset + y' \cos \emptyset \tag{2.29}$$

que coincide con la ecuación (2.5).

2.4 Modelo del direccionamiento diferencial

Para especificar la configuración de un robot móvil (diferencial) hay que indicar los valores de las variables $(x, y, \emptyset)$. Se tiene una restricción no holónoma y dos grados de libertad.

En el caso de los robots con direccionamiento diferencial se tienen dos ruedas actuadas de radio r separadas por una distancia b . Esta distancia de separación es la vía del vehículo y a la mitad de esta distancia se ubica el *punto de guiado* p para el cual se hacen los cálculos de la cinemática del vehículo (véase Figura 2-6).

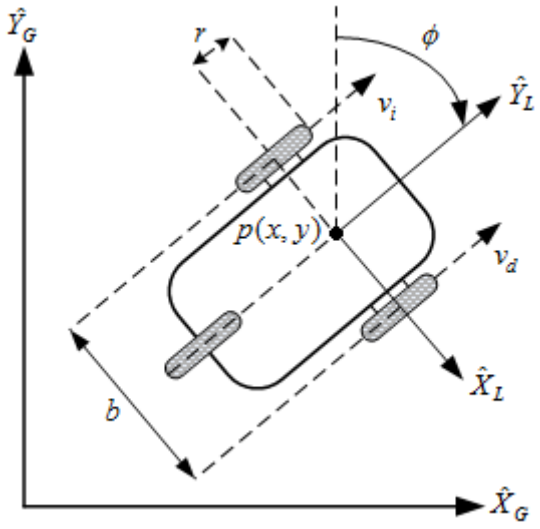


Figura 2-6. Diagrama del direccionamiento diferencial

Como se mencionó en el tema 1.4.2 las variables de control en este tipo de locomoción son las velocidades de las ruedas actuadas. Entonces si ω_i y ω_d son las velocidades angulares de las ruedas izquierda y derecha respectivamente, las velocidades lineales de cada rueda son:

$$v_i = r\omega_i \tag{2.30}$$

$$v_d = r\omega_d \tag{2.31}$$

Supóngase que el robot se mueve a lo largo del eje $+\hat{Y}$ de su sistema local de referencia $\{L\}$[1]. A continuación se considera por separado la contribución de velocidad angular de ambas ruedas para la traslación del punto p en la dirección $+\hat{Y}$.

[1] Esto quiere decir que el sentido positivo del eje Y del sistema local del robot coincide con el avance hacia enfrente del mismo.

Si la rueda derecha gira mientras la izquierda permanece estática (ver Figura 2-7-a), el punto p se moverá con una velocidad lineal de:

$$v_1 = \frac{1}{2} v_d \qquad (2.32)$$

que corresponde a la mitad de la velocidad lineal de la rueda derecha, debido a que el punto p se encuentra a la mitad de ambas ruedas.

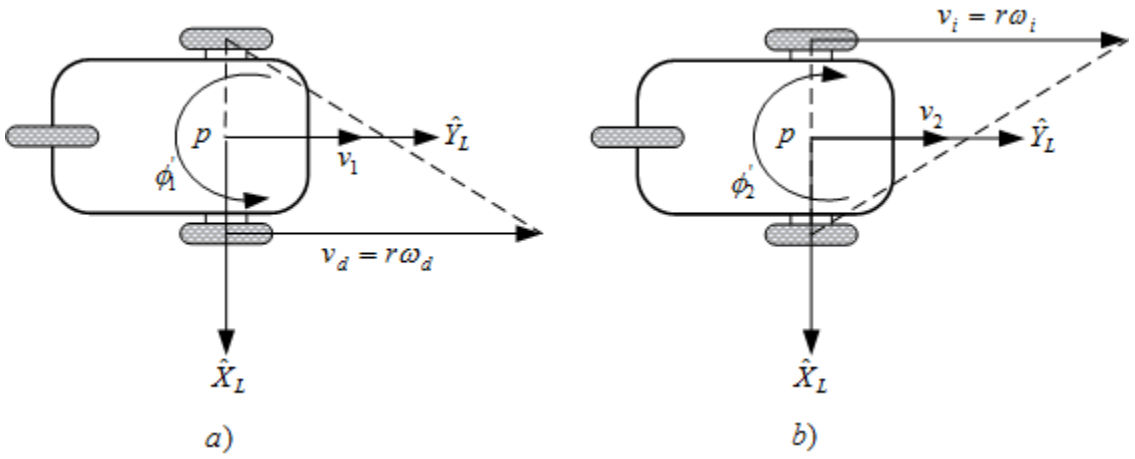


Figura 2-7. Contribuciones por separado de las ruedas a la velocidad lineal y angular del vehículo

Así mismo, si la rueda izquierda gira mientras la derecha permanece estática (ver Figura 2-7-b) la velocidad lineal del punto p será de:

$$v_2 = \frac{1}{2} v_i \qquad (2.33)$$

en el direccionamiento diferencial ambas contribuciones simplemente se suman para obtener la velocidad lineal del robot:

$$v = v_1 + v_2$$

$$v = \frac{v_d + v_i}{2} \qquad (2.34)$$

$$v = \frac{r(\omega_d + \omega_i)}{2} \qquad (2.35)$$

Para calcular la velocidad angular del robot una vez más se consideran independientemente las contribuciones de cada rueda y posteriormente se suman.

Si se considera que solo la rueda derecha gira (ver Figura 2-7-a) y que lo hace hacia delante, se tendrá que el giro del robot será en sentido contrario a las manecillas del reloj. Entonces la rueda derecha se mueve a lo largo de una circunferencia de radio b y la velocidad angular del robot debido solo a la rueda derecha será:

$$\emptyset_1' = \frac{r\omega_d}{b} \tag{2.36}$$

Este desarrollo aplica de igual manera para la rueda izquierda (ver Figura 2-7-b), con la única diferencia de que el giro hacia delante de esta rueda resulta en un giro del robot en sentido de las manecillas del reloj:

$$\emptyset_2' = \frac{-r\omega_i}{b} \tag{2.37}$$

sumando ambas expresiones se tiene que la velocidad angular del robot es:

$$\omega = \emptyset_1 + \emptyset_2$$

$$\omega = \frac{r\omega_d}{b} - \frac{r\omega_i}{b}$$

$$\omega = \frac{r(\omega_d - \omega_i)}{b} \tag{2.38}$$

de esta última expresión además se observa que:

$$\omega = \frac{v_d - v_i}{b} \tag{2.39}$$

En caso de que se especifiquen la velocidad lineal v y la velocidad angular ω del robot, las velocidades angulares que hay que aplicar a las ruedas izquierda y derecha son:

$$\omega_i = \frac{v - (b/2)\omega}{c} \tag{2.40}$$

$$\omega_d = \frac{v + (b/2)\omega}{c} \tag{2.41}$$

Tomando en cuenta (2.35) y (2.38) la ecuación (2.24) puede expresarse en función de las variables de control:

$$\begin{bmatrix} x' \\ y' \\ \emptyset' \end{bmatrix} = \begin{bmatrix} \frac{-r\sin\emptyset}{2} \\ \frac{r\cos\emptyset}{2} \\ -\frac{r}{b} \end{bmatrix} \omega_i + \begin{bmatrix} \frac{-r\sin\emptyset}{2} \\ \frac{r\cos\emptyset}{2} \\ \frac{r}{b} \end{bmatrix} \omega_d$$

$$\begin{bmatrix} x' \\ y' \\ \emptyset' \end{bmatrix} = \begin{bmatrix} \frac{-r\sin\emptyset}{2} & \frac{-r\sin\emptyset}{2} \\ \frac{r\cos\emptyset}{2} & \frac{r\cos\emptyset}{2} \\ -\frac{r}{b} & \frac{r}{b} \end{bmatrix} \begin{bmatrix} \omega_i \\ \omega_d \end{bmatrix} \tag{2.42}$$

2.5 Modelo odométrico del robot diferencial

Al integrar el movimiento de las ruedas se puede *estimar* en todo momento la posición y orientación del robot, esto se logra empleando sensores que permiten medir la distancia recorrida por el vehículo (odómetros), que en este caso son los codificadores ópticos (encoders).

De esta manera, y conociendo la posición y orientación iniciales: $p_0 = [x_0 \quad y_0 \quad \emptyset_0]^T$, al integrar las ecuaciones del modelo (2.24) se tiene:

$$\begin{bmatrix} x \\ y \\ \emptyset \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \emptyset_0 \end{bmatrix} + \begin{bmatrix} \int_0^t -v\sin\emptyset\, d\tau \\ \int_0^t v\cos\emptyset\, d\tau \\ \int_0^t \omega d\tau \end{bmatrix} \tag{2.43}$$

En el caso del direccionamiento diferencial al sustituir (2.34) y (2.39) en (2.43) resulta:

$$\begin{bmatrix} x \\ y \\ \emptyset \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \emptyset_0 \end{bmatrix} + \begin{bmatrix} \int_0^t -\frac{v_d + v_i}{2}\sin\emptyset\, d\tau \\ \int_0^t \frac{v_d + v_i}{2}\cos\emptyset\, d\tau \\ \int_0^t \frac{v_d - v_i}{b} d\tau \end{bmatrix} \tag{2.44}$$

Finalmente, sustituyendo (2.30) y (2.31) la ecuación (2.44) se puede expresar en función de las velocidades angulares de las ruedas información que es dada por los codificadores ópticos:

$$\begin{bmatrix} x \\ y \\ \emptyset \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \emptyset_0 \end{bmatrix} + \begin{bmatrix} \int_0^t \frac{-r \sin \emptyset}{2} (\omega_d + \omega_i) d\tau \\ \int_0^t \frac{r \cos \emptyset}{2} (\omega_d + \omega_i) d\tau \\ \int_0^t \frac{r}{b} (\omega_d - \omega_i) d\tau \end{bmatrix} \tag{2.45}$$

Es conveniente señalar que en odometría existe un error de posición y orientación debido a una variedad de fuentes de error como lo son; las interacciones dinámicas con el entorno (las cual no están contempladas en el modelo cinemático), problemas de resolución (tiempo de muestreo, resolución de los codificadores ópticos, etc.) y deslizamientos inevitables de las ruedas del robot.

Capítulo 3

Entorno de desarrollo LabVIEW

LabVIEW es una plataforma y entorno de desarrollo basado en programación gráfica, que permite realizar aplicaciones y diseñar sistemas de medidas y control científicos y de ingeniería. Brinda una integración con hardware con una amplia compatibilidad, que aunado a la programación gráfica permite realizar aplicaciones de forma rápida y relativamente sencilla.

La palabra LabVIEW es un acrónimo de Laboratory Virtual Instrumentation Engineering Workbench. Esta herramienta debe ese nombre al hecho de que originalmente estaba enfocada al control de instrumentos electrónicos, ya que su apariencia y operación reproduce los instrumentos físicos. Los programas realizados en LabVIEW son llamados Instrumentos Virtuales, se guardan en archivos con la extensión *.vi por las siglas de Virtual Instrument (Instrumento Virtual en español), por lo que es común referirse a ellos como VIs.

3.1 LabVIEW como lenguaje de programación

El lenguaje de programación usado por LabVIEW se denomina lenguaje G, donde la G se refiere a que es gráfico, el lenguaje G tiene los mismos conceptos de programación que se encuentran en la mayoría de los lenguajes tradicionales de programación, tales como tipos de datos, ciclos, programación jerárquica, manejo de eventos, variables, recursividad, programación orientada a objetos, etc.

LabVIEW se diferencia de la mayoría de los otros lenguajes de programación de propósito general en dos formas principalmente. Primero, la programación en LabVIEW se hace conectando entre sí iconos gráficos en un diagrama, posteriormente este código gráfico es compilado directamente a código máquina de manera que el procesador de la computadora pueda ejecutarlo. La segunda diferencia es que el código G desarrollado con LabVIEW se ejecuta de acuerdo a las reglas de flujo de datos en lugar de la manera tradicional que consiste en una serie secuencial de comandos para ser llevados a cabo encontrada en la mayoría de los lenguajes de programación basados en texto como lo son C y C++.

Los lenguajes basados en el flujo de datos como el lenguaje G, utilizan los datos como el principal concepto detrás de cada programa. El orden de ejecución está determinado por el flujo de datos entre los nodos de un programa, no en líneas secuenciales de texto como en los lenguajes tradicionales. Esta pequeña diferencia tiene un gran impacto, ya que traza las rutas de datos entre las diferentes partes del programa, de manera que ese flujo de datos se convierte en el enfoque principal del programador. Los nodos en un programa de LabVIEW (por ejemplo, funciones, estructuras tales como ciclos, subrutinas, etc.) tienen entradas, procesan datos, y producen salidas. Una vez que

todas las entradas de un nodo determinado contienen datos válidos, ese nodo ejecuta su lógica, produce datos de salida, y pasa los datos al siguiente nodo en la ruta del flujo de datos. Un nodo que recibe datos de otro nodo puede ejecutarse solo después de que el otro nodo completó su ejecución como se muestra en la Figura 3-1.

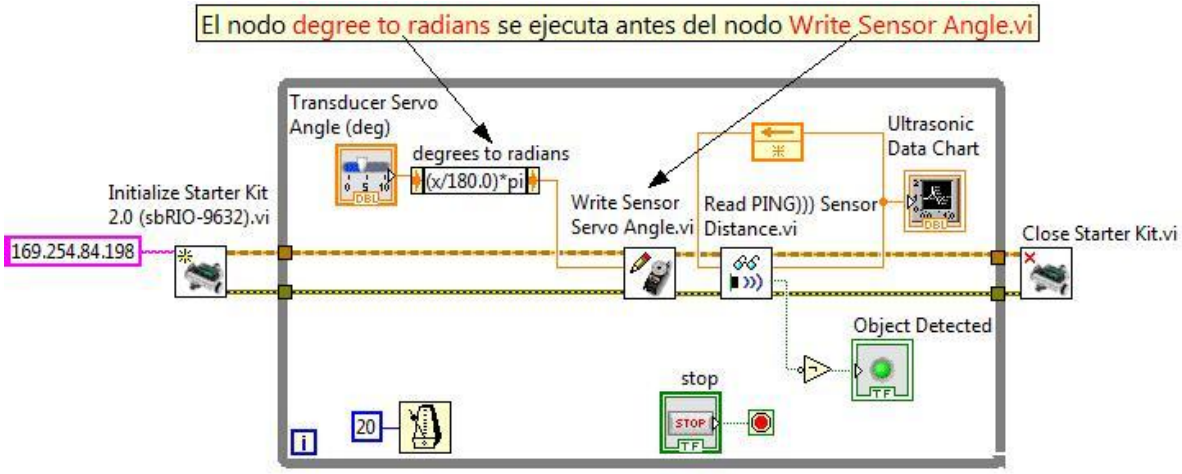


Figura 3-1. Ejemplo del flujo de datos

3.2 Características generales LabVIEW

Como se mencionó anteriormente, los programas hechos en LabVIEW son llamados instrumentos virtuales o VIs debido a que su apariencia simula instrumentos físicos, como por ejemplo un osciloscopio o un multímetro. Labview contiene un amplio conjunto de herramientas para adquirir, analizar, mostrar, y almacenar datos, así como herramientas de ayuda para solucionar problemas con el código escrito.

Un VI se compone de dos ventanas, la ventana del panel frontal y la ventana del diagrama de bloques. El panel frontal (Figura 3-2) es la interfaz gráfica de usuario y se compone de controles e indicadores, los cuales son las terminales de entrada y salida del VI respectivamente, esta interfaz adquiere las entradas procedentes del usuario y muestra la representación de las salidas proporcionadas por el programa, permitiendo al usuario interactuar con el VI, los controles simulan dispositivos de entrada de los instrumentos como perillas, push buttons, selectores, y otros dispositivos de entrada que permiten introducir datos al VI, los indicadores simulan dispositivos de salida de los instrumentos como gráficas, leds, y otros displays empleados para mostrar los resultados producidos por el VI tales como datos adquiridos o resultados de algún proceso u operación.

El diagrama de bloques es el programa propiamente dicho, es donde se crea el código fuente del VI, los objetos del diagrama de bloques llamados nodos son representados por bloques o iconos gráficos, los objetos que se pueden encontrar en el diagrama de bloques incluyen terminales, subVIs, funciones, constantes, y estructuras, los cuales están integrados en diferentes librerías de LabVIEW, todos los objetos colocados en el panel frontal aparecen en el diagrama de bloques como terminales.

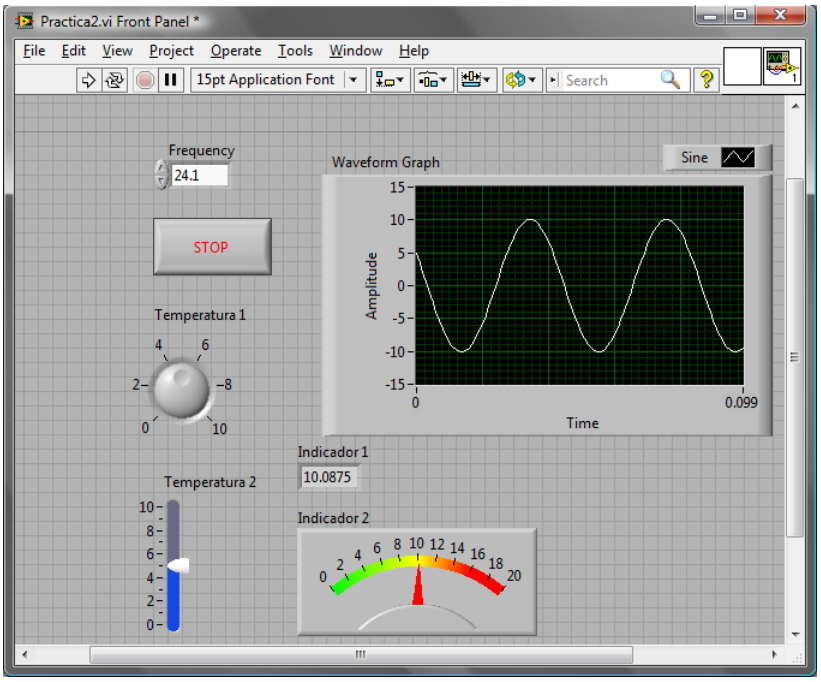


Figura 3-2. Panel Frontal de un VI

El diagrama de bloques (ver Figura 3-3) se construye conectando los distintos objetos (iconos gráficos) entre sí mediante cables, estos cables son las rutas que siguen los datos desde su origen hasta su destino y transfieren los datos entre los diferentes nodos del diagrama de bloques. Cada cable tiene un color diferente dependiendo del tipo de dato (booleano, cadena de caracteres, entero, flotante, etc.) que fluya a través de él. Los controles colocados en el panel frontal proporcionan datos de entrada al diagrama de bloques, y los indicadores del panel frontal despliegan los datos que el diagrama de bloques adquiere o genera.

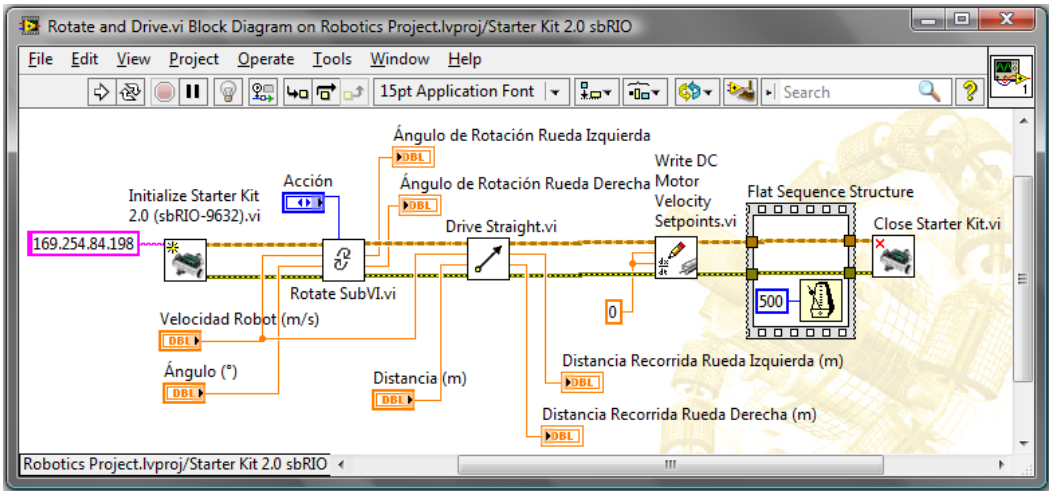


Figura 3-3. Ejemplo de un Diagrama de Bloques con tres subVIs: Rotate SubVI, Drive Straight y Write DC Motor Velocity Setpoints

En el lenguaje G las funciones y las estructuras son nodos elementales, y son análogos a los operadores o librerías de funciones de los lenguajes de programación convencionales. Por ejemplo, una estructura en LabVIEW corresponde a un ciclo en los lenguajes convencionales donde se ejecuta el código que contienen de forma condicional o repetitiva (ciclo for, while, etc.)

Cuando un VI es llamado desde el diagrama de bloques de otro VI (como se observa en la Figura 3-3) recibe el nombre de subVI, estos subVIs permiten encapsular código y hacerlo reutilizable en otros VIs. Además permiten crear VIs más limpios y con una mejor estructura que puede facilitar la programación al hacer más claro el flujo de datos en el diagrama de bloques. Un subVI es similar a una subrutina en los lenguajes de programación basados en texto.

LabVIEW tiene una gran variedad de librerías con muchas utilidades y funciones para adquisición de datos, matemáticas, estadística, acondicionamiento y análisis de señales, diseño de filtros digitales, robótica, algoritmos de control, etc., estas utilidades y funciones de programación se encuentran agrupadas en diferentes categorías a las que se puede acceder mediante la paleta de funciones (Figura 3-4) del diagrama de bloques.

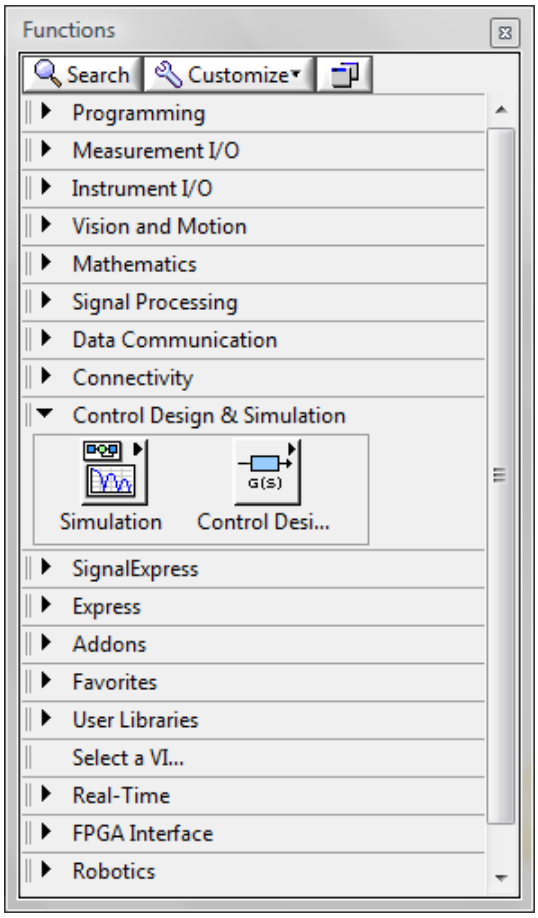


Figura 3-4. Paleta de Funciones

LabVIEW es un entorno de desarrollo gráfico en forma modular, lo que permite al programador añadir juegos de herramientas (toolkits) y módulos adicionales al sistema de desarrollo inicial, haciendo que la funcionalidad de LabVIEW crezca según sea necesario. Este software adicional, que puede ser de National Instruments o de un tercero, contiene funciones diseñadas para aplicaciones específicas en una gran variedad de áreas, algunos ejemplos de toolkits y módulos son: Módulo LabVIEW Real-Time, Módulo LabVIEW FPGA, Módulo Vision Development para LabVIEW, Módulo LabVIEW Robotics, LabVIEW PID and Fuzzy Logic Toolkit, Spectral Measurements Toolkit, y LabVIEW Interface for Arduino Toolkit.

3.2.1 Estructuras de programación

Una estructura es un elemento de control del programa. Las estructuras contienen secciones de código gráfico y controlan cómo y dónde se ejecuta dicho código, controlando así el flujo de datos en un VI. A continuación se describen algunas de las estructuras utilizadas en la programación hecha en este trabajo.

Estructuras de casos

Una estructura de casos tiene dos o más casos donde cada caso tiene una sección de código (subdiagrama), en el diagrama de bloques de un VI solamente es visible un caso, la estructura ejecuta solamente un caso a la vez. Para determinar qué caso se ejecuta se conecta un valor a la terminal de entrada (selector) de la estructura. La estructura de casos es similar a las instrucciones de un interruptor o las instrucciones *si después (if else)* en lenguajes de programación basados en texto. En la parte superior de la estructura de caso se encuentra la etiqueta del selector de caso, la cual contiene el nombre del valor del selector que corresponde al caso en cuestión. Con las flechas de incremento y decremento se desplaza en los casos disponibles (ver Figura 3-5).

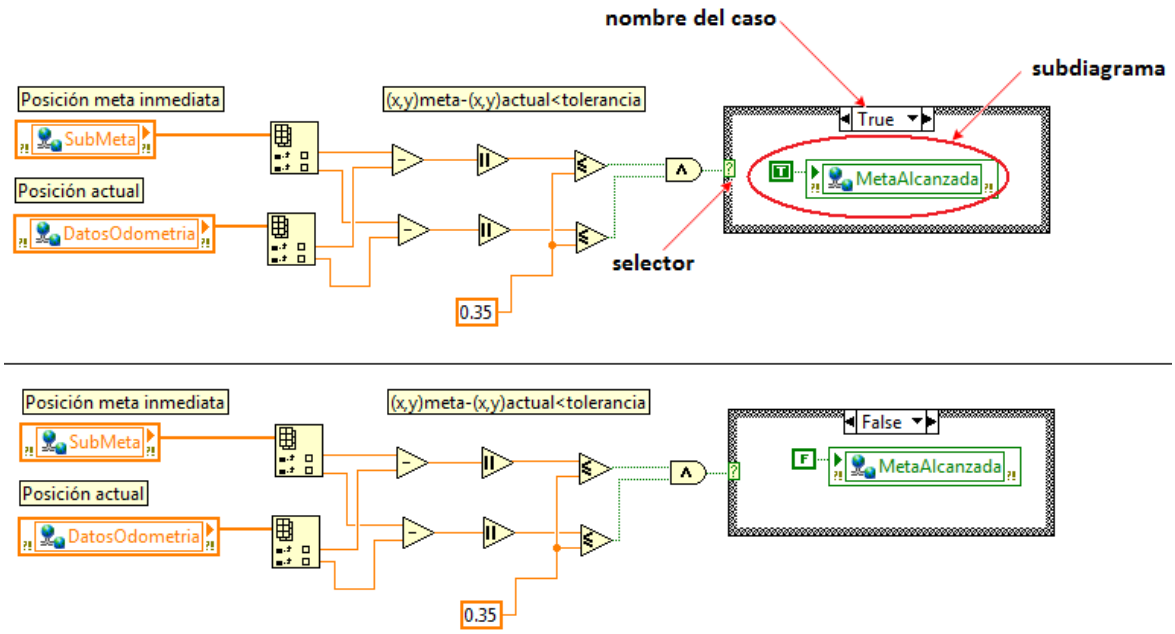


Figura 3-5. Ejemplo de una estructura de caso, con dos casos

En la Figura 3-5 se observa un segmento de código que produce una salida la cual es utilizada para seleccionar el caso que se ejecutará en la estructura de caso, para este ejemplo se tienen solo dos casos, en los cuales se escribe un valor booleano en una variable según sea el caso.

Ciclo temporizado

Esta estructura es en esencia un *ciclo While* (con una o más secuencias) el cual es similar a un *ciclo Do* o a un *Ciclo Repeat-Until* en lenguajes de programación basados en texto. Un ciclo While ejecuta el código dentro de sus bordes hasta que ocurre una condición de paro (ver Figura 3-6). Al ser un ciclo temporizado, permite ejecutar los subdiagramas dentro de esta estructura a una razón de tiempo especificada. Con esta estructura se pueden especificar diferentes tipos de temporización como el periodo, prioridad, límites de tiempo, compensaciones y tiempos de espera.

Algunos de los nodos en la terminal de entrada de este ciclo son:

- Error: El ciclo temporizado no se ejecuta si se recibe una condición de error.
- Periodo: Especifica el periodo de ejecución del ciclo temporizado en las mismas unidades de la fuente de temporización.
- Prioridad: Especifica la prioridad de ejecución del ciclo temporizado. La prioridad de una estructura especifica cuando se ejecuta en el diagrama de bloques en relación con otros objetos en el diagrama.
- Nombre de fuente: Especifica el nombre de la fuente de temporización usada para controlar la estructura. La fuente de temporización puede ser creada en la ventana de configuración del ciclo temporizado la cual aparece al dar doble click en el nodo de entrada del ciclo.

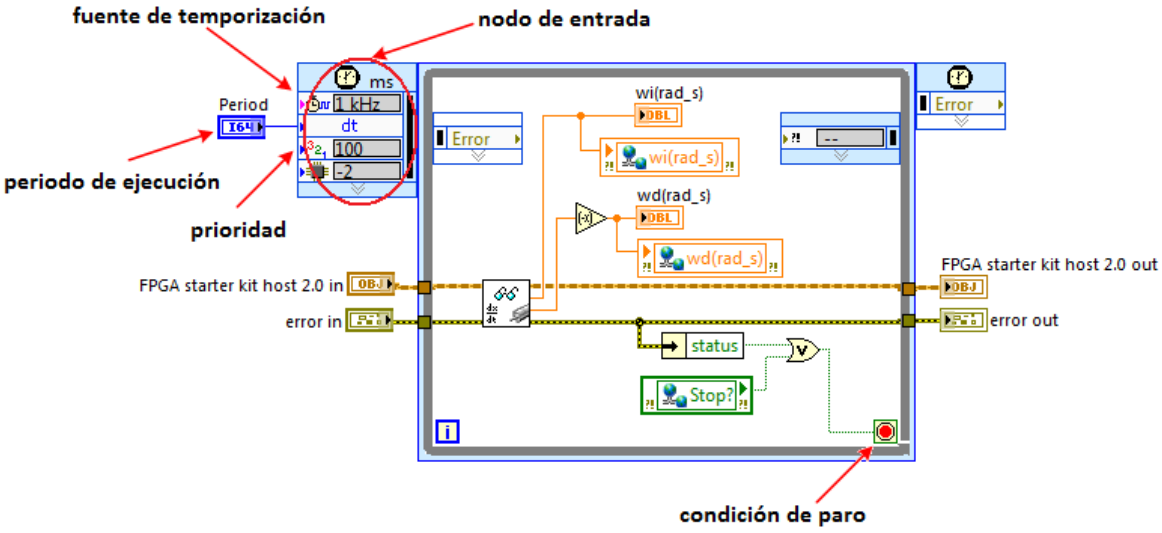


Figura 3-6. Estructura temporizada

En la Figura 3-6 se observa una estructura temporizada, el subdiagrama que se encuentra dentro se ejecuta mientras no se presente la condición de paro, se pude

observar los nodos de entrada que definen la fuente de temporización (reloj de 1kHz), el periodo de ejecución y su prioridad.

Ciclo de control y simulación

Permite analizar el comportamiento de modelos de sistemas simulándolos o implementándolos físicamente. Esta estructura ejecuta el diagrama en su interior hasta que se alcanza el tiempo final de simulación o hasta que una condición de alto detiene la ejecución. Se puede colocar un subdiagrama en el interior de esta estructura o se puede crear un subsistema y colocarlo dentro de la estructura.

Esta estructura tiene un nodo de entrada y uno de salida, el nodo de entrada se utiliza para configurar los parámetros. Los parámetros también pueden ser configurados directamente en el nodo de entrada o en la ventana de configuración de parámetros de simulación la cual aparece al dar doble click en el nodo de entrada. Los parámetros de simulación pueden ser de dos tipos: de simulación y de temporización. En los parámetros de simulación se puede definir: el tiempo inicial y final de simulación, el método de solución, las características del método de solución. En los parámetros de temporización se puede definir: la fuente de temporización, la prioridad y el periodo de ejecución. La terminal de error en el nodo de entrada detiene la ejecución de la estructura de control y simulación en caso de recibir una referencia de error.

El subdiagrama en el interior de la estructura de control y simulación de la Figura 3-7 se ejecuta hasta que se dé la condición de paro, en este subdiagrama se puede observar un par de subsistemas.

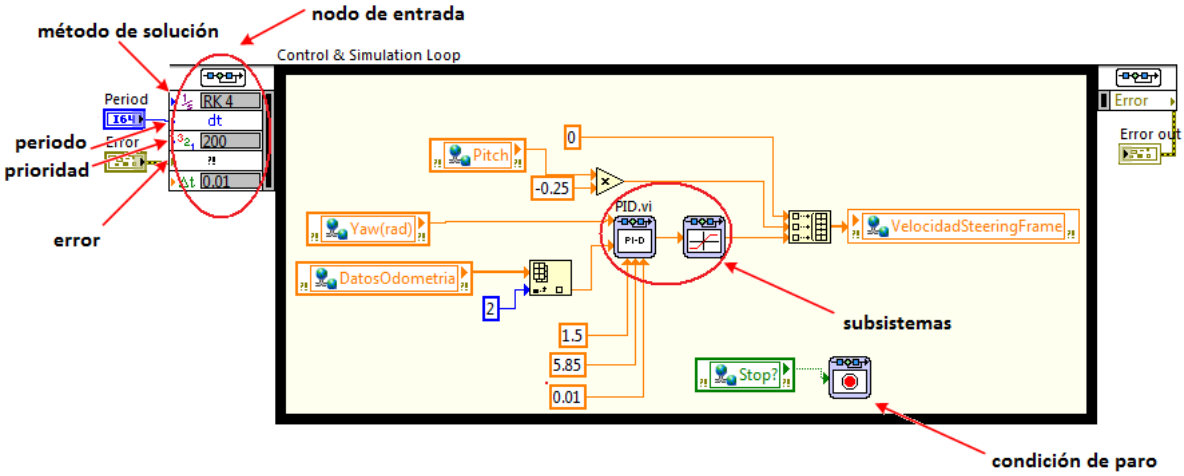


Figura 3-7. Estructura de control y simulación

Variable global

Una variable global es un VI que contiene datos al que se puede tener acceso para leer o escribir datos desde cualquier parte de una aplicación, además este tipo de variables permite enviar o recibir datos desde diferentes VIs activos (ver Figura 3-8).

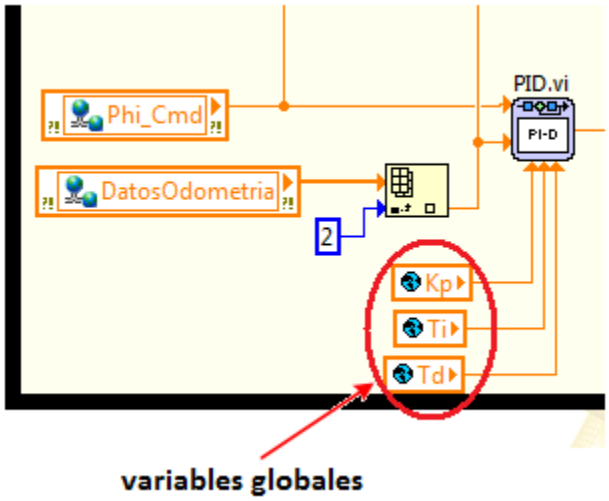


Figura 3-8. Apariencia de variables globales en el diagrama de bloques

Variable compartida

Una variable compartida permite leer y escribir datos entre diferentes VIs en un proyecto a través de una red, a diferencia de las variables globales, las variables compartidas pueden enviar o recibir datos a través de diferentes objetivos (targets) como pueden ser entre una computadora portátil y un controlador en tiempo real o una FPGA (ver Figura 3-9).

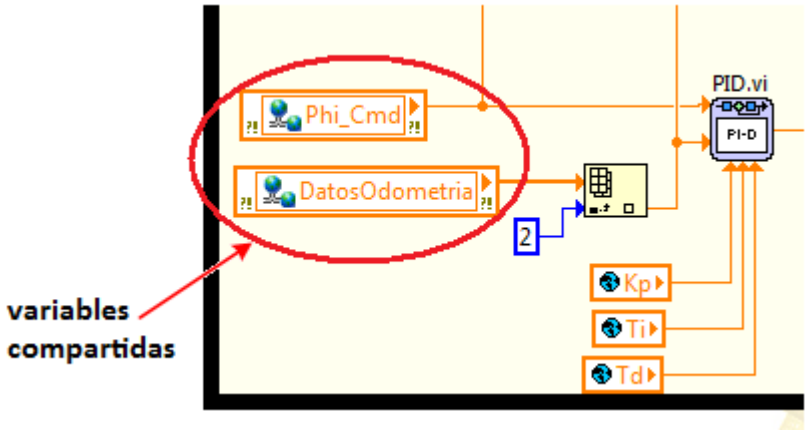


Figura 3-9. Apariencia de variables compartidas en el diagrama de bloques

3.3 LabVIEW Robotics

El módulo LabVIEW Robotics es una herramienta de software que permite desarrollar aplicaciones de robótica usando LabVIEW. Proporciona soluciones estándar para hardware y desarrollo de software para el diseño de control de robots. El módulo de robótica dota a LabVIEW de una vasta librería con utilidades que permiten establecer comunicación con sensores y actuadores de robots, implementación de algoritmos para una operación inteligente y una percepción robusta, funciones de movimiento para hacer

que un robot o vehículo se mueva, e incluso realizar simulaciones de robots en un ambiente diseñado.

El módulo Robotics se usa en conjunto con los módulos Real-Time, FPGA, Vision Development, Control Design and Simulation y SoftMotion principalmente, el uso de otros módulos depende de la aplicación que se desarrolle.

3.3.1 Funciones del módulo LabVIEW Robotics

A continuación se describen las paletas de funciones (Figura 3-10) contenidas en el módulo LabVIEW Robotics.

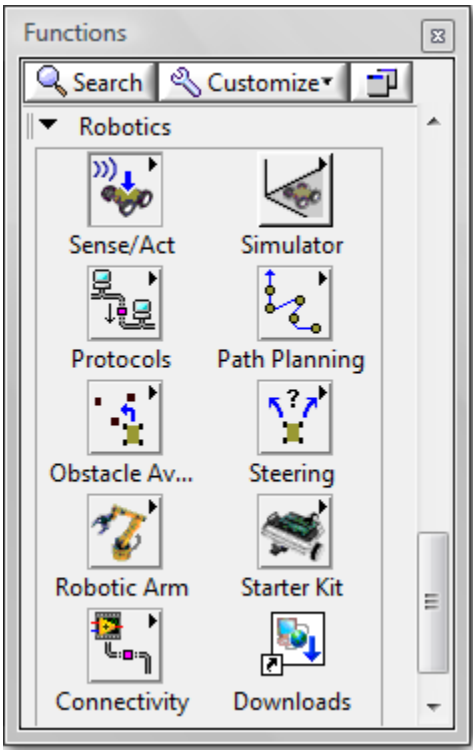


Figura 3-10. Paleta de funciones del módulo Robotics

Subpaleta Sensores y Actuadores (Sensors and Actuators)

Contiene los controladores (*drivers*) necesarios para configurar y controlar algunos de los sensores y actuadores más usados en sistemas de robótica, ya sean reales o simulados (ver Figura 3-11). En la paleta Instrument I/O existen otros drivers de gran utilidad. Para buscar e instalar drivers adicionales se puede usar el *buscador de controladores de instrumentos (NI Instrument Drive Finder).*

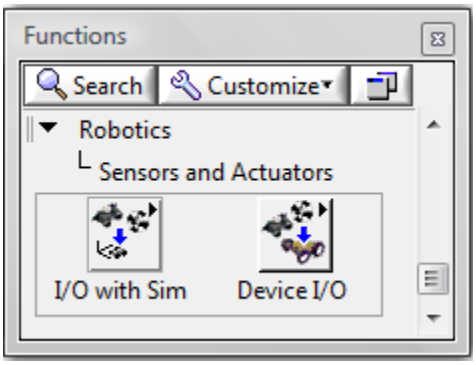


Figura 3-11. Contenido de la subpaleta Sense/Act

Subpaleta Simulator

Esta paleta de herramientas contiene los VIs para controlar el simulador de robótica y para interactuar con los componentes de una simulación de robótica (Figura 3-12). Incluye la subpaleta *Offline Configuration* la cual permite crear y modificar *archivos manifiesto* [2] desde los cuales el simulador de robótica lee para desplegar los componentes simulados. Se recomienda utilizar la *aplicación de creación de ambientes de robótica* (*Robotics Enviroment Simulator Wizard)* si no es necesario incluir componentes personalizados en la escena de simulación.

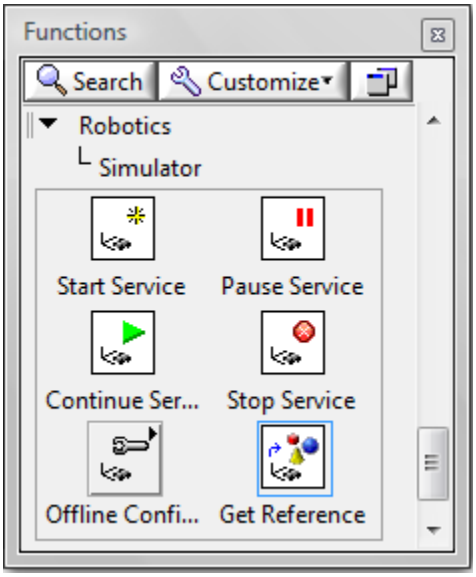


Figura 3-12. Contenido de la subpaleta Simulator

[2] Es un archivo dentro de un proyecto de LabVIEW que contiene información de la simulación tal como: características y constantes físicas del ambiente de trabajo, sensores y dimensiones de la estructura del robot, entre otros

Subpaleta Protocols

Los VIs de esta paleta (Figura 3-13) permiten procesar datos formateados en protocolos de comunicación, tales como datos enviados por sensores, un ejemplo de estos datos serían los provenientes de un GPS.

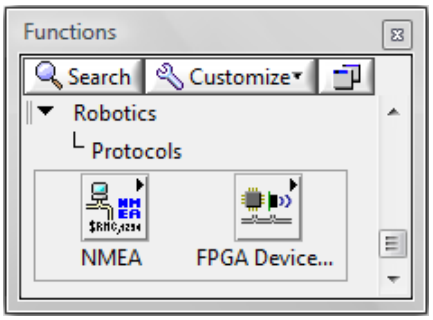


Figura 3-13. Contenido de la subpaleta Protocols

Subpaleta Path Planning

Contiene VIs con algoritmos para calcular trayectorias hacia un punto meta dentro de un mapa que representa el ambiente o espacio de trabajo del robot. Las trayectorias calculadas pueden ser las óptimas o de menor coste. El coste se refiere a un número positivo que representa variables definidas por el usuario en las que los algoritmos de trazado de trayectorias minimizan cuando calculan las trayectorias, estas variables pueden ser la distancia entre dos puntos o la energía necesaria para atravesar un área. Un coste de valor infinito *(Inf)* significa que el área no puede ser atravesada, por ejemplo, porque un obstáculo bloquea el paso a través de un área. Esta subpaleta (Figura 3-14) también contiene las subpaletas *Directed Graph Map* y *Ocuppancy Grid Map*, usadas para crear y controlar trayectorias en mapas que representan el ambiente donde el robot se mueve.

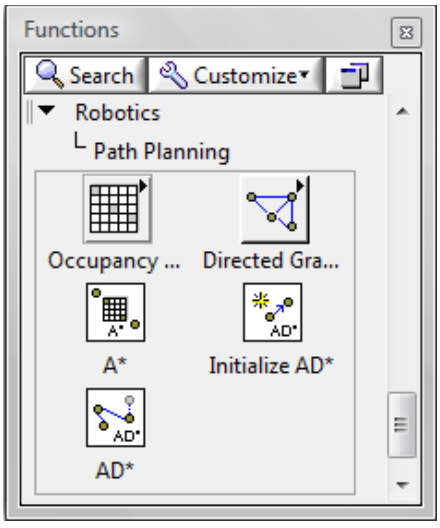


Figura 3-14. Subpaletas y VIs de la subpaleta Path Planning

Subpaleta Obstacle Avoidance

Contiene VIs para implementar la evasión de obstáculos en robots móviles o vehículos (Figura 3-15). Estos VIs identifican obstáculos y huecos, o áreas libres en el ambiente del robot, y regresan la dirección libre de viaje que mejor lleva a la posición de meta, con la información identificada permiten implementar una reacción de movimiento en particular.

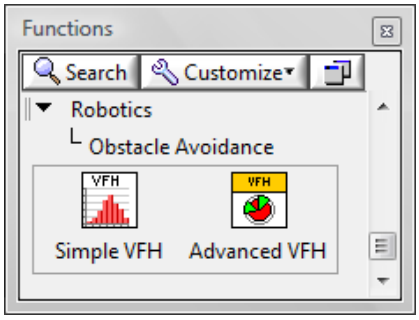


Figura 3-15. VIs de la subpaleta Obstacle Avoidance

Subpaleta Steering

Contiene VIs para crear un sistema en LabVIEW con la información de la estructura del vehículo robótico a desarrollar, por ejemplo; número y tipo de ruedas, dimensiones del robot y tipo de direccionamiento. Estos VIs (ver Figura 3-16) permiten calcular la velocidad del robot y de las ruedas para hacer conversiones entre estas velocidades, además de poder conectarse con los motores que mueven las ruedas se puede establecer el control de estos motores.

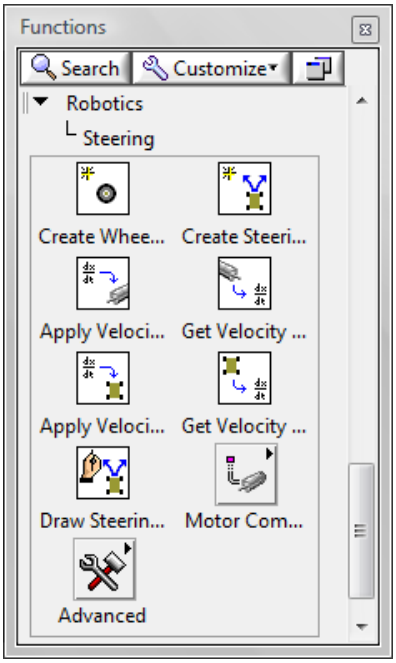


Figura 3-16. Contenido de la paleta Steering

Subpaleta Robotic Arm

Contiene Vis para crear e interactuar con un brazo robótico simulado (Figura 3-17). Se pueden realizar cálculos de la cinemática y dinámica de un brazo, simular un brazo, y posteriormente hacer el prototipo del brazo.

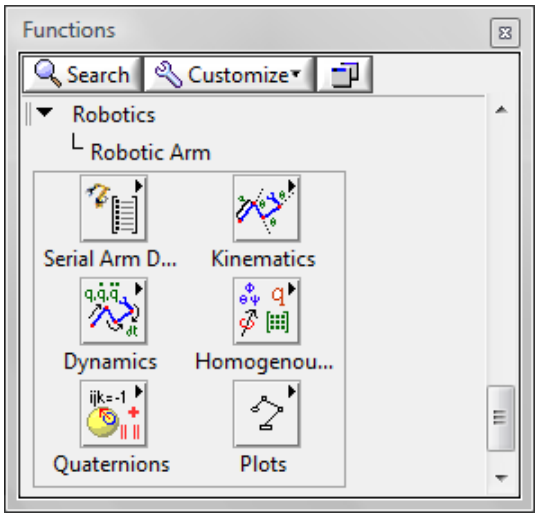


Figura 3-17. Contenido de la subpaleta Robotic Arm

Subpaleta Starter Kit

Contiene dos subpaletas con VIs para controlar al robot DaNI del Kit de Inicio de Robótica de LabVIEW (LabVIEW Robotics Starter Kit), estos VIs permiten controlar al robot sin la necesidad de escribir un VI FPGA. Cada una de las subpaletas que se muestran en la Figura 3-18 corresponden a una de las dos versiones del robot DaNI: 1.0 ó 2.0 (Figura 3-19).

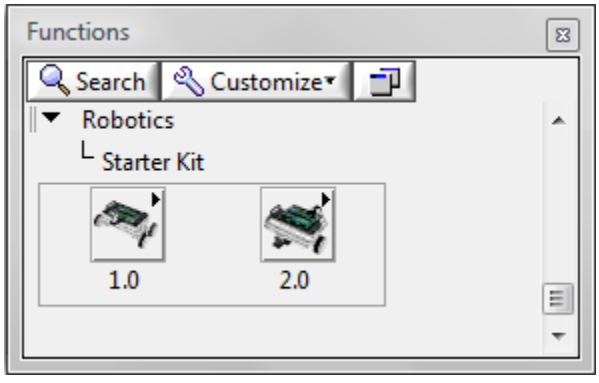


Figura 3-18. Contenido de la subpaleta Starter Kit

Con los VIs contenidos se pueden controlar los motores, adquirir información del sensor ultrasónico y desplegar programas en el controlador NI sbRIO del robot.

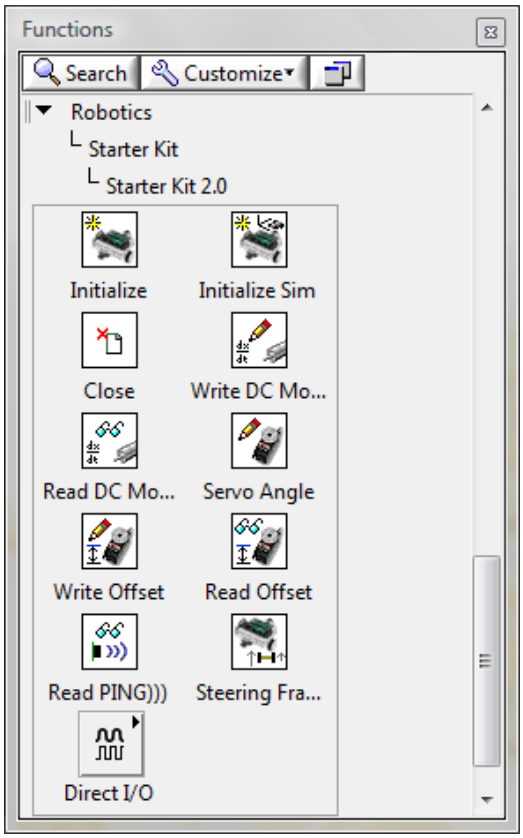


Figura 3-19. VIs de la subpaleta Starter Kit 2.0

Subpaleta Connectivity

Las herramientas de esta paleta (Figura 3-20) permiten buscar en la red VIs que interactúen con otro software de robótica, incluyendo software de otras compañías diferentes a National Instruments.

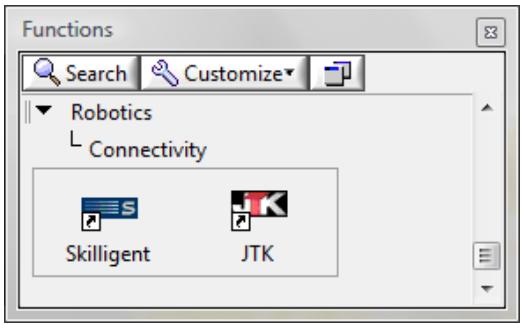


Figura 3-20. Contenido de la subpaleta Connectivity

Subpaleta Downloads

Abre un explorador Web a la sección de descargas relacionadas con robótica (*robotics-related downloads)* en la página de National Instruments.

3.3.2 Componentes de un sistema de LabVIEW Robotics

Un sistema de robótica consiste en componentes de hardware y software. En un sistema diseñado con el módulo LabVIEW Robotics, el software incluye proyectos y VIs creados en LabVIEW. Los componentes de hardware de un sistema de robótica pueden incluir los siguientes:

- Host – Es la computadora *huésped*, se refiere al dispositivo donde se desarrollan y depuran las aplicaciones de robótica como podría ser el caso de una computadora. Se puede establecer comunicación con el robot y registrar datos usando a la computadora huésped, correr aplicaciones desarrolladas en el huésped, o desplegarlas en un sistema basado en Windows o en un controlador de tiempo real.
- Sistema de control del robot – Se refiere a la estructura física con direccionamiento de un robot que incluya los siguientes componentes:
    - Controlador de tiempo real – Ejecuta los programas creados en LabVIEW de forma determinística y controla datos de entrada y salida con un controlador de tiempo real o RT, el cual sirve como el cerebro del sistema robot.
    - Sensores y Actuadores – Permiten adquirir los datos acerca del ambiente del robot con los sensores, y controlar el movimiento del robot con los actuadores.

Un ejemplo de un sistema de robótica de LabVIEW es un proyecto con una computadora huésped (host) y un dispositivo objetivo (target), en el caso del proyecto de la Figura 3-21, el huésped es una computadora portátil y el objetivo es la tarjeta sbRIO-9632 del robot DaNI.

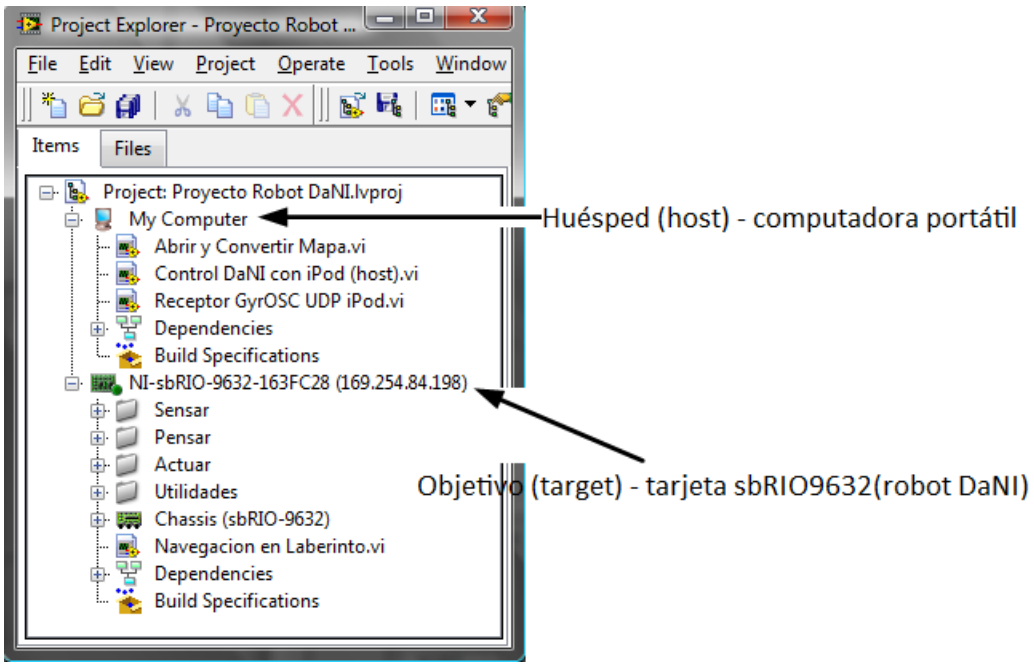


Figura 3-21. Proyecto de robótica de LabVIEW

Capítulo 4

Descripción LabVIEW Robotics Starter Kit 2.0

El *LabVIEW Robotics Starter Kit 2.0* o también conocido como *robot DaNI* (Figura 4-1) es una plataforma de robótica comercial de grado industrial diseñada para la enseñanza, investigación y desarrollo de prototipos usando el software LabVIEW. El robot DaNI se utiliza en conjunto con los módulos de software de LabVIEW necesarios para su programación.

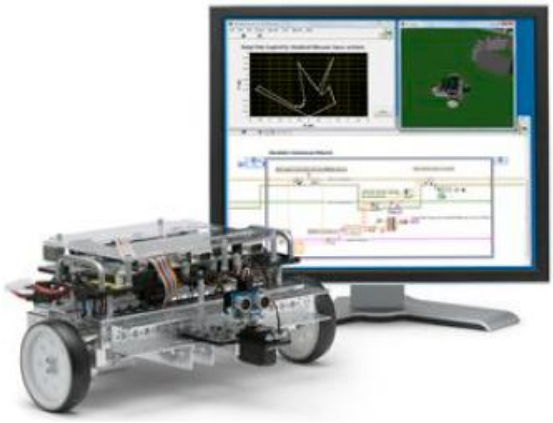


Figura 4-1. Robot DaNI

El robot DaNI es un robot móvil que consiste en una estructura física de robot marca Pitsco TETRIX ya ensamblada con direccionamiento de tipo diferencial con ruedas, un sensor ultrasónico, un par de codificadores ópticos de cuadratura (encoders) acoplados a los dos motores de DC, un controlador (driver) para el control de los motores, una batería para la alimentación del robot, y una tarjeta embebida de control y adquisición de datos NI sbRIO-9632, la ubicación de los componentes básicos en el robot DaNI puede verse en la Figura 4-2.

La programación se realiza en una computadora huésped (*host)* y la comunicación entre ésta y el robot se establece mediante una conexión de red, en este caso una conexión Ethernet.

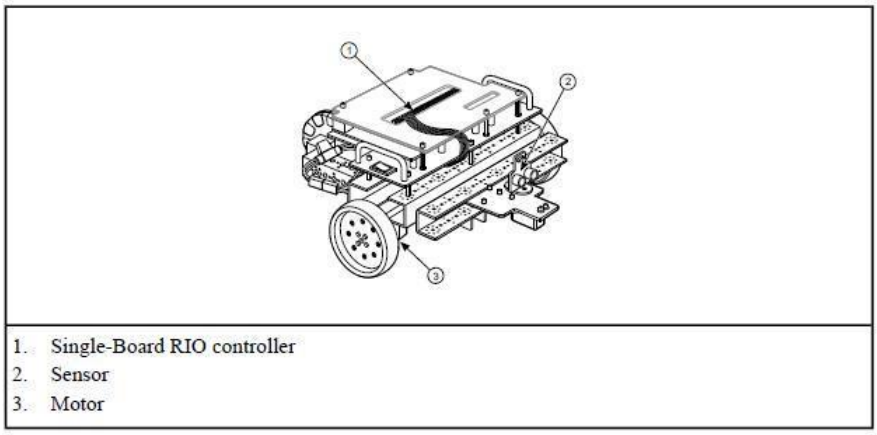


Figura 4-2. Componentes principales del robot DaNI

4.1 Tarjeta NI sbRIO-9632

La tarjeta NI sbRIO-9632 (ver Figura 4-3) es un dispositivo embebido para control y adquisición de datos que integra un controlador en tiempo real, una FPGA reconfigurable, y puertos de entrada y salida en una sola tarjeta.

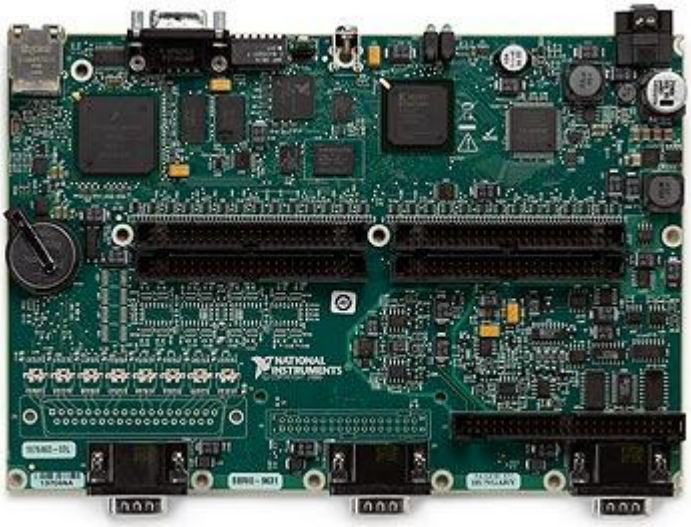


Figura 4-3. Tarjeta NI sbRIO-9632

El procesador de la tarjeta sbRIO-9632 es un procesador de grado industrial en tiempo real Freescale MPC5200 de 400 MHz que permite aplicaciones determinísticas en tiempo real. El procesador esta internamente combinado vía PCI con una FPGA Xilinx Spartan-3 de 2M de compuertas. La FPGA está conectada directamente a todas las entradas y salidas (E/S) digitales. De igual manera cada módulo integrado de E/S analógico y digital tiene una conexión dedicada a la FPGA.

La tarjeta sbRIO-9632 tiene líneas digitales de E/S, canales de E/S analógicos, también tiene tres conectores para expansión de los puertos de E/S. Su rango de

temperatura de operación es de -20 a 55 °C. Se puede alimentar con una fuente de potencia de 19 a 30 VDC, además tiene 128 MB en DRAM para operación embebida y 256MB de memoria no volátil para programas de almacenamiento y registro de datos. Tiene un puerto Ethernet de 10/100 Mbits/s que permite realizar la programación de la tarjeta desde una computadora o *Host* a través de una red. Además cuenta con un puerto serial RS232 para controlar dispositivos externos o periféricos.

El dispositivo sbRIO-9632 es programado a través LabVIEW, usando los módulos LabVIEW Real-Time (VxWorks) y LabVIEW FPGA. El procesador en tiempo real corre el módulo LabVIEW Real-Time en el sistema operativo Wind River VxWorks (RTOS).

Como se muestra en la Figura 4-4, los diferentes dispositivos con que cuenta el robot DaNI se encuentran conectados a la tarjeta sbRIO-9632 a través de los puertos digitales de entrada y salida (DIO).

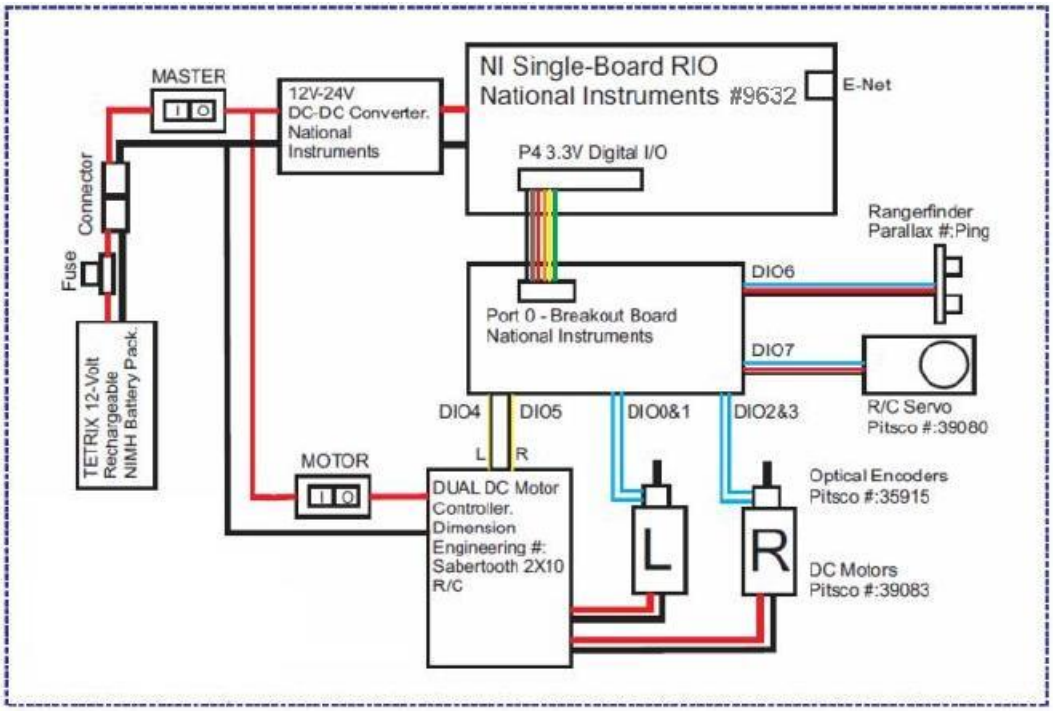


Figura 4-4. Diagrama de bloques con la interconexión de los dispositivos que integran el robot DaNI

En la Figura 4-4 se puede observar que la batería de 12V DC alimenta a todos los componentes del robot. Un driver controla los motores izquierdo (L) y derecho (R), este driver está conectado a los puertos DIO4 y 5 de la tarjeta sbRIO. El sensor ultrasónico está conectado al puerto DIO6 y el servo en el que está montado está conectado al puerto DIO7. La tarjeta sbRIO está conectada a la batería vía un convertidor de DC-DC.

Las especificaciones técnicas de la tarjeta sbRIO-9632 se pueden consultar a detalle en [15] y [19].

4.2 Motores y ruedas

Para lograr su locomoción, el robot DaNI posee dos motores de DC de 12 V con caja reductora de engranes de 1:52 marca Pitsco Education, cada uno de los dos motores tiene acoplada a su eje una rueda estándar marca Pitsco TETRIX de 4 in de radio (ver Figura 4-5).

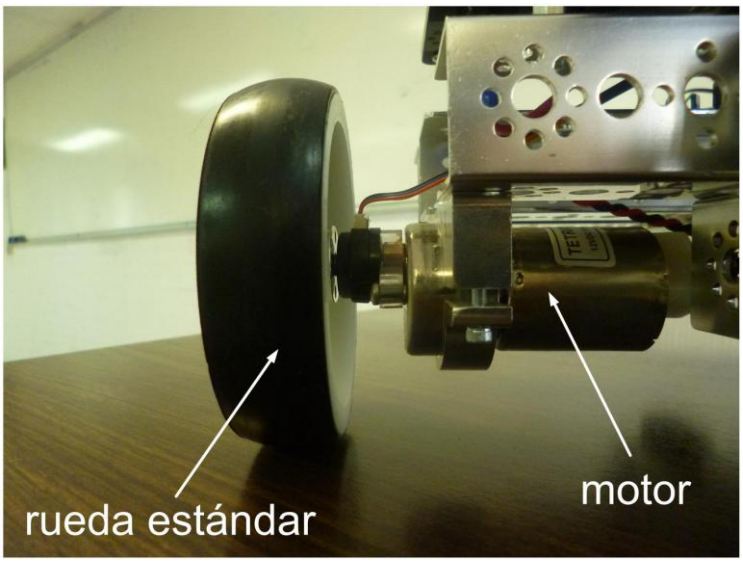


Figura 4-5. Vista de uno de los motores del robot DaNI y su rueda

La estructura del robot DaNI tiene una tercera rueda en la parte posterior (Figura 4-6), esta rueda es de tipo pasivo ya que no está acoplada a ningún motor y su función es dar estabilidad a la estructura. Esta rueda es de tipo sueco 90° con barriles que giran libremente y que permiten movimientos perpendiculares al sentido de giro de la rueda.

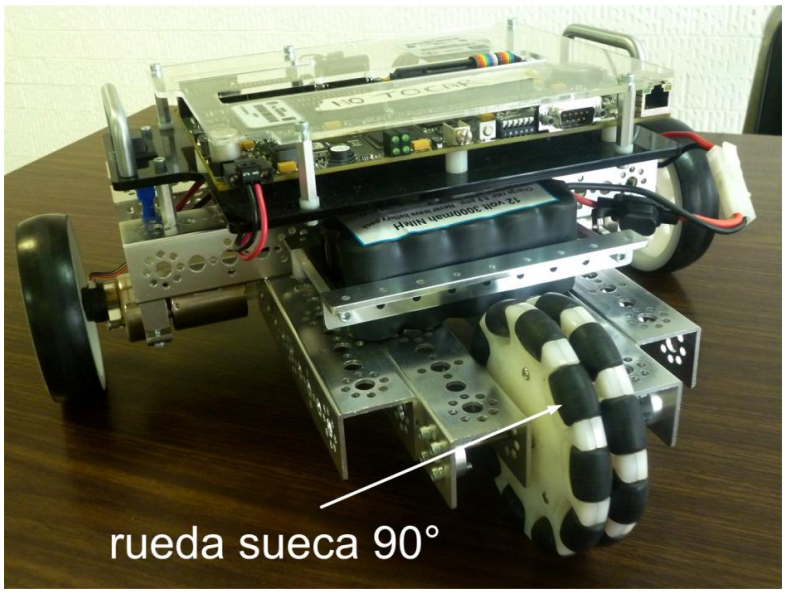


Figura 4-6. Vista posterior del robot DaNI donde se aprecia la rueda omnidireccional

Esta configuración de ruedas y motores le da a la estructura del robot DaNI un direccionamiento de tipo *diferencial*, donde el avance y giro del robot se logra con la *"diferencia"* de velocidades angulares de las dos ruedas actuadas.

El control de la velocidad de los motores se logra variando el voltaje que reciben éstos usando modulación por ancho de pulso ó PWM (Pulse Width Modulation), el driver para motores marca Dimension Engineering mostrado en la Figura 4-7 es el encargado de esto. Este driver está conectado a la batería del robot de dónde obtiene la energía con que alimenta a los dos motores, a su vez se encuentra conectado a la tarjeta sbRIO-9632 (véase Figura 4-4) de donde recibe las instrucciones para variar el voltaje que entrega a los motores.

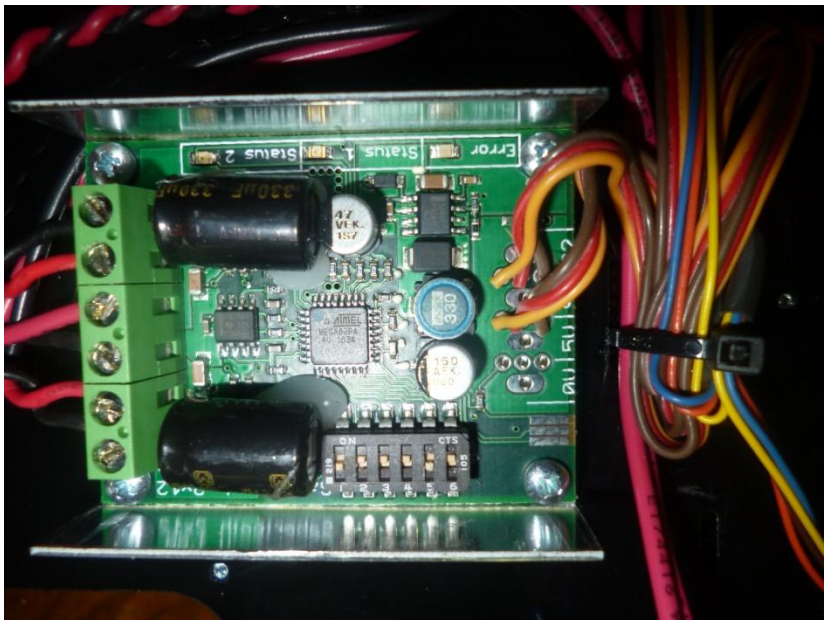
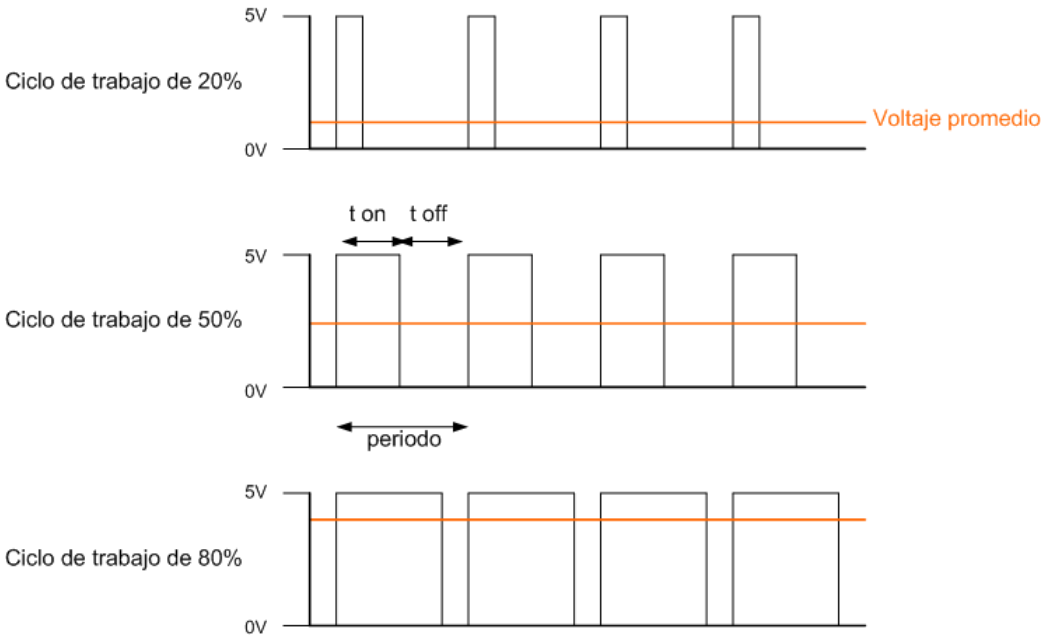


Figura 4-7. Driver de los motores

La modulación por ancho de pulsos es una técnica que permite controlar la potencia suministrada a un dispositivo eléctrico, especialmente a cargas como motores. Consiste en controlar el valor del voltaje (y corriente) promedio que alimenta la carga, al encender y apagar muy rápidamente el interruptor ubicado entre la carga y la fuente de energía, este encendido y apagado corresponde a un tren de pulsos aplicado al interruptor, donde un cero lógico representa al interruptor apagado y un uno lógico representa al interruptor encendido, así la energía suministrada a la carga será directamente proporcional al tiempo que el interruptor estuvo encendido.

La relación del tiempo que el interruptor esta encendido (t_{on}) entre el periodo del tren de pulsos se define como ciclo de trabajo y se expresa en porcentajes (ver Gráfica 4-1).



Gráfica 4-1**.** PWM con diferentes ciclos de trabajo, se puede apreciar la relación entre ciclo de trabajo y voltaje promedio

Entre más tiempo esté encendido el interruptor comparado con el tiempo que permanece apagado (t_{off}), la potencia suministrada a la carga es mayor, en el caso de los motores entre más grande sea el ciclo de trabajo se tendrá mayor potencia y por tanto una mayor velocidad de giro. En el robot DaNI la señal digital PWM proveniente de la tarjeta sbRIO-9632 proporciona la misma señal PWM de potencia a los dos motores para activar los motores a diferentes velocidades y sentidos.

El módulo LabVIEW Robotics contiene el bloque *Write DC Motor Velocity Setpoints* (Figura 4-8), este bloque es un VI que permite aplicar valores de velocidad en sentido contrario a las manecillas del reloj en unidades de radianes por segundo [rad/s] a los motores izquierdo y derecho del robot, en los cuales se identifican su posiciones al observar al robot DaNI por detrás. La velocidad angular máxima a la que pueden girar los motores es de 15.7 [rad/s] lo que corresponde a una velocidad de avance hacia enfrente del robot DaNI de aproximadamente 0.79 [m/s].

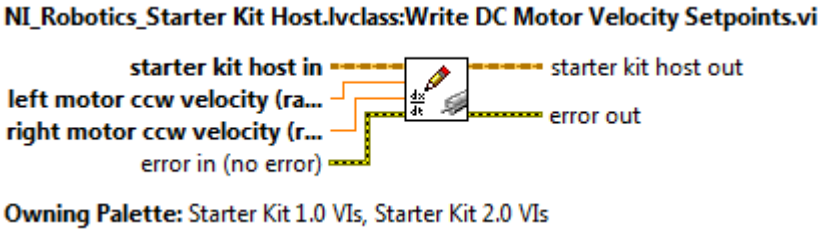


Figura 4-8. Bloque *Write DC Motor Velocity Setpoints* y sus conexiones de entrada y salida

Las especificaciones técnicas de los motores y del controlador de los motores del robot DaNI se pueden consultar a detalle en [16], [17].

4.3 Codificadores ópticos de cuadratura (encoders)

Para medir la velocidad a la que giran los dos motores, cada uno de ellos tiene incorporado un codificador óptico de cuadratura (ver Figura 4-9) de 400 pulsos por revolución marca Pitsco TETRIX.

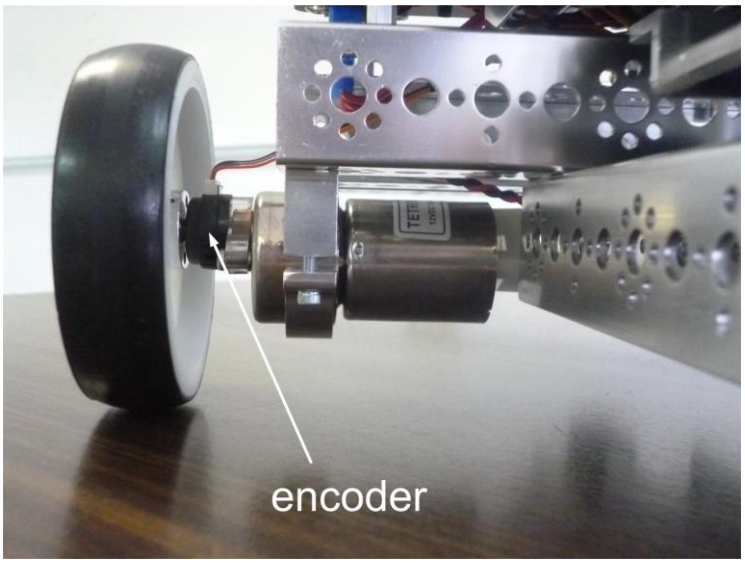


Figura 4-9. Ubicación del encoder acoplado a un motor

Un codificador óptico de cuadratura es un dispositivo que permite medir la posición y velocidad angular del eje de un motor al cual esta acoplado. El dispositivo consiste en un disco con un fino patrón ranurado que gira junto con el eje del motor, el disco permite o no el paso de luz proveniente de un led emisor de luz hacia los detectores ópticos colocados al otro lado del disco (véase Figura 4-10), esto provoca una variación de la luz recibida por los detectores ópticos los cuales producen una señal discreta cuadrada donde se tienen dos estados: luz u obscuro, esta señal discreta posteriormente se digitaliza.

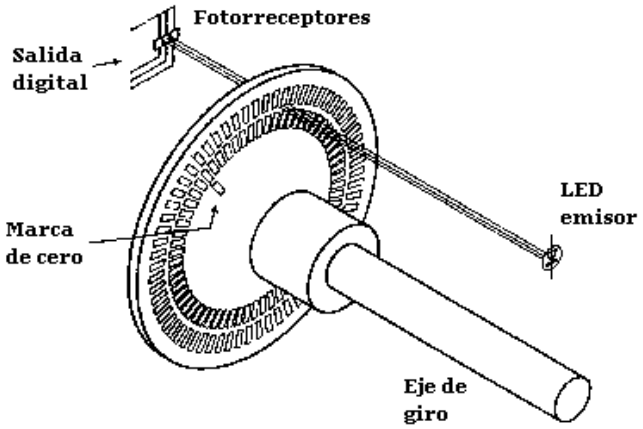
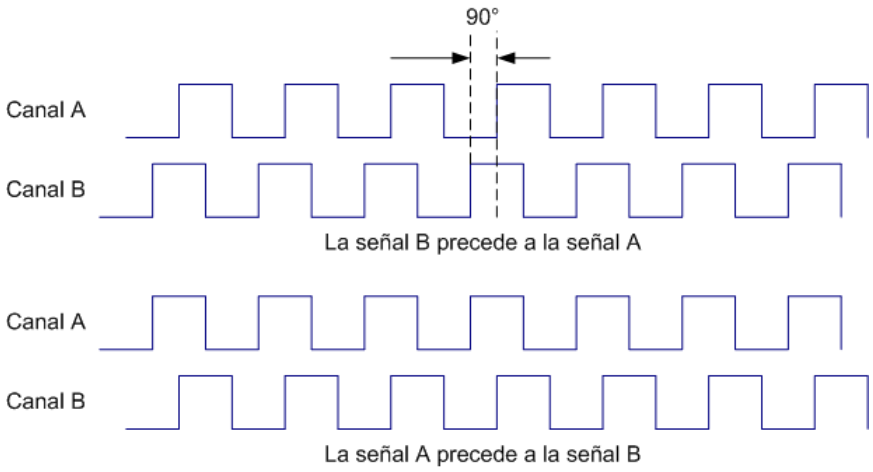


Figura 4-10. Esquema de un encoder óptico de cuadratura

En este tipo de encoders, el patrón en el disco ranurado y los dos detectores ópticos hacen que se obtengan dos señales defasadas 90° como se observa en la Gráfica 4-2, el orden en el cual se produce primero un flanco de subida por alguna de la señales identifica la dirección de rotación del motor; cuando la señal B antecede a la señal A se tiene un sentido de giro en contra de las manecillas del reloj, la situación inversa corresponde a un sentido de giro en el sentido de las manecillas del reloj.



Gráfica 4-2. Tipos de señales que genera el encoder óptico de cuadratura

En el robot DaNI cada encoder acoplado a los motores entrega dos señales a la tarjeta sbRIO-9632 a través de los puertos digitales DIO0 y DIO1 para el motor izquierdo, y DIO2 y DIO3 para el motor derecho (véase Figura 4-4).

El módulo LabVIEW Robotics contiene un bloque llamado *Read DC Motor Velocities* (Figura 4-11) el cual es un subVI que lee y entrega el valor en [rad/s] de la velocidad angular a la que giran los motores izquierdo y derecho en sentido contrario a las manecillas del reloj.

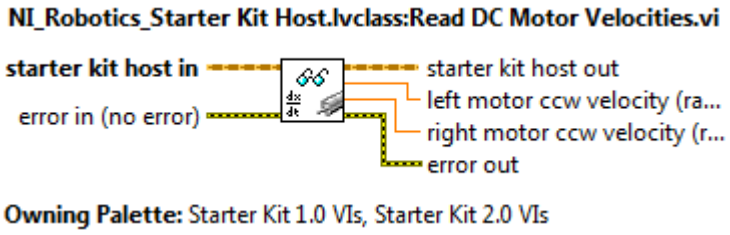


Figura 4-11. Bloque *Read DC Motor Velocities con sus conexiones de entrada y salida*

4.4 Sensor ultrasónico

El robot DaNI posee un sensor ultrasónico modelo PING))) de la marca Parallax (Figura 4-12), el cual permite hacer mediciones de distancia entre el sensor y un objeto sin tener contacto físico dentro del rango de 2cm a 3m.

Figura 4-12. Sensor ultrasónico PING)). El sensor posee un sonar que emite ondas ultrasónicas y un micrófono para detectarlas, tiene tres terminales: GND, 5V y SIG para las señales de control.

Para medir la distancia hasta un objeto, el sensor PING))) usa su sonar incorporado para transmitir una ráfaga de ondas ultrasónicas (por encima del umbral del oído humano) de 40kHz durante 200ms, inmediatamente que deja de transmitir el sensor PING))) habilita el estado de receptor, las ondas ultrasónicas viajan por el aire hasta golpear un objeto cuando sucede esto parte de las ondas se reflejan y regresan pudiendo así ser detectadas por el micrófono del sensor PING))) como se muestra en la Figura 4-13.

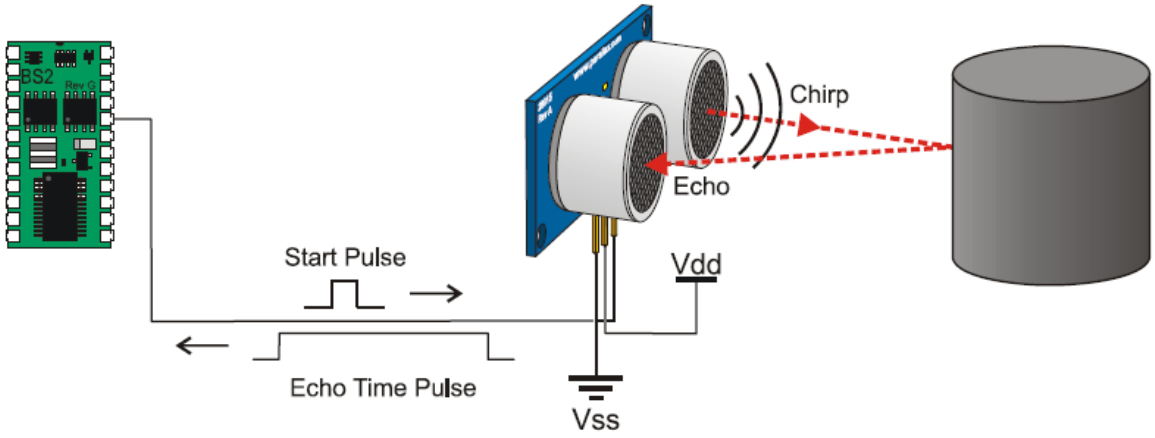


Figura 4-13. Detección de un objeto con el sensor PING)))

De esta manera la distancia d entre el sensor y un objeto se determina de la siguiente manera:

$$d = \frac{c_{aire} * t}{2}$$

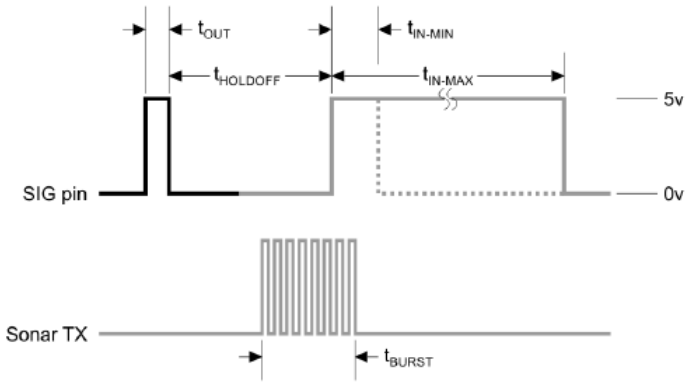
donde c_{aire} es la velocidad de las ondas ultrasónicas en el aire y t es el tiempo en segundos que le toma a las ondas ultrasónicas ir y regresar después de reflejarse al golpear con un objeto. La distancia debe dividirse entre dos debido a que las ondas ultrasónicas recorren dos veces la distancia entre el sensor y el objeto detectado, al viajar de ida, golpear el objeto y regresar.

La velocidad c_{aire} se calcula de la siguiente manera:

$$c_{aire} = 331.5 * (0.6 * T_C)\ m/s$$

donde T_C es la temperatura ambiente, para fines prácticos el valor de la temperatura ambiente se considera constante de valor 20°C debido a que el robot DaNI está pensado para ser utilizado en interiores.

Cuando el sensor PING))) termina de emitir la ráfaga de ondas ultrasónicas, éste envía una señal digital en alto por el puerto digital DIO6 a la FPGA de la tarjeta sbRIO-9632 (ver Figura 4-14), esta señal se mantiene en alto hasta que la onda ultrasónica reflejada es detectada, la FPGA cuenta el tiempo que se mantuvo la señal digital en alto usando su reloj interno de 40MHz, de esta manera se calcula el tiempo t que se mantuvo la onda ultrasónica viajando por el aire.



▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

Figura 4-14. Protocolo de comunicaciones del sensor PING))). La terminal SIG del sensor recibe los pulsos que activan al sensor, posteriormente a través de ella se manda un pulso a la tarjeta sbRIO-9632 con la duración del viaje de la onda emitida al ser detectada de regreso.

El módulo LabVIEW Robotics contiene el bloque *Read PING))) Sensor Distance* (Figura 4-15), este es un SubVI entrega el valor de la distancia que existe entre el sensor ultrasónico PING))) y un objeto dentro del rango de detección del sensor, en caso de que no haya un objeto entrega un valor por default, que en este caso lo recomendable es que sea 3m debido a que es el máximo valor de distancia que puede ser detectado según sus especificaciones técnicas.

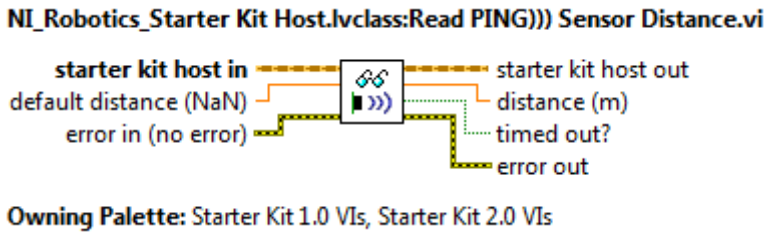


Figura 4-15. Bloque *Read PIING))) sensor Distance* con sus conexiones de entrada y salida

Existen algunas condiciones (ver Figura 4-16) para las cuales el sensor PING))) no está diseñado y no puede medir con exactitud la distancia a un objeto:

a) El objeto se encuentra a más de tres metros del sensor.

b) El objeto tiene una superficie reflectora en un ángulo muy pequeño de tal manera que las ondas ultrasónicas no se reflejan de regreso al sensor.

c) El objeto es muy pequeño para reflejar suficientes ondas ultrasónicas.

d) El objeto tiene una superficie suave e irregular de algún material como tela, alfombra o peluche de manera que no puede reflejar suficientes ondas ultrasónicas para ser detectadas

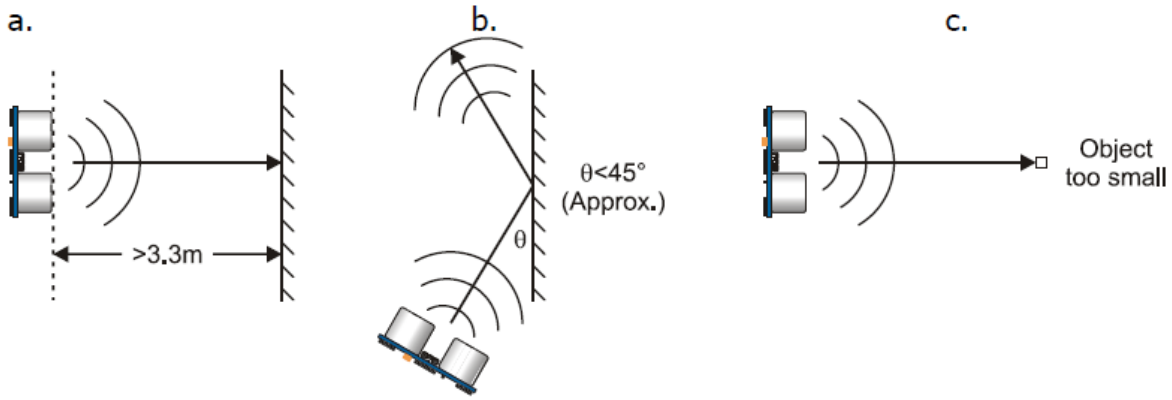


Figura 4-16. Algunas situaciones en las cuales el sensor PING))) no puede detectar con exactitud un objeto

El sensor PING))) está ubicado al frente de la estructura del robot DaNI para poder medir a qué distancia se encuentran los objetos que son potenciales obstáculos que impiden el libre tránsito del robot. Este sensor puede medir distancias a un objeto dentro de un rango o campo de visión limitado, esto quiere decir que pueden existir obstáculos no detectados que estén fuera de este campo de visión y que pudieran golpear la estructura del robot DaNI mientras éste avanza y se acerca a los obstáculos. Para evitar esto el sensor PING))) se encuentra montado sobre un servomotor como se observa en la Figura 4-17, este servo al girar hace que el sensor PING))) haga un barrido permitiéndole "ver" o detectar un espacio mayor del entorno que se encuentra frente al robot DaNI.

El servomotor se hace girar un total de 130°, los cuales se dividen en dos sectores: uno positivo y uno negativo cada uno de 65°, los ángulos positivos se encuentran del lado izquierdo del valor de 0° y los negativos del lado derecho.

El valor de 0° corresponde a la posición del servo donde la placa del sensor PING))) se encuentra paralela a la estructura del robot como se observa en la Figura 4-17. Debido a que el servomotor puede girar más de 130° es necesario definir su posición de 0°, esto se puede hacer cuando se configura el robot DaNI (ver Figura 4-26). Las especificaciones técnicas del sensor PING))) se pueden consultar a detalle en [21].

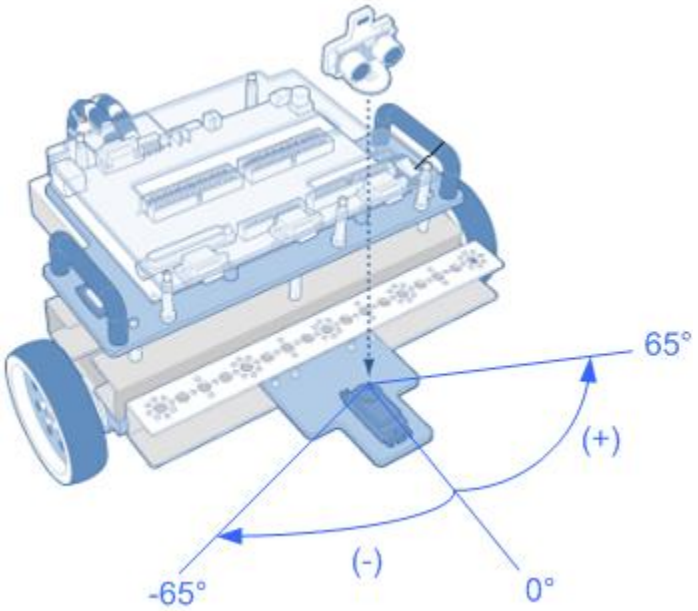


Figura 4-17. Sensor PING))) sobre el servomotor

4.5 Comunicación del robot DaNI con una computadora huésped (host)

La programación del robot DaNI se hace desde una computadora externa o *host*. En esta computadora se debe tener instalada la plataforma de LabVIEW y todos los módulos necesarios. Una vez que se realiza el programa se descarga y ejecuta en la tarjeta sbRIO-9632 del robot.

La conexión entre el host y el robot DaNI se hace mediante una conexión de red (ver Figura 4-18). Donde es posible conectar directamente el robot DaNI al Host mediante una conexión Ethernet usando un cable CAT 5, o establecer una conexión inalámbrica con el robot a través de una red Wi-Fi siempre y cuando se incorpore al robot un router inalámbrico.

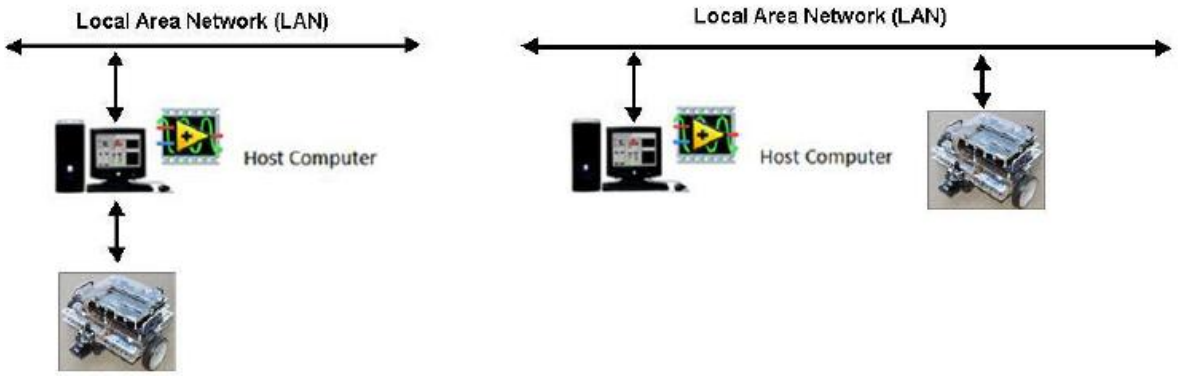


Figura 4-18. Conexión en red del robot DaNI y el *host*

La forma más sencilla de establecer comunicación entre el robot DaNI y el host es usando la *aplicación de configuración de hardware (Hardware Setup Wizard)*. Esta aplicación va guiando paso a paso y configura la comunicación automáticamente.

Para empezar, es necesario se escoger el ambiente *LabVIEW Robotics* en la ventana de selección de ambiente (ver Figura 4-19).

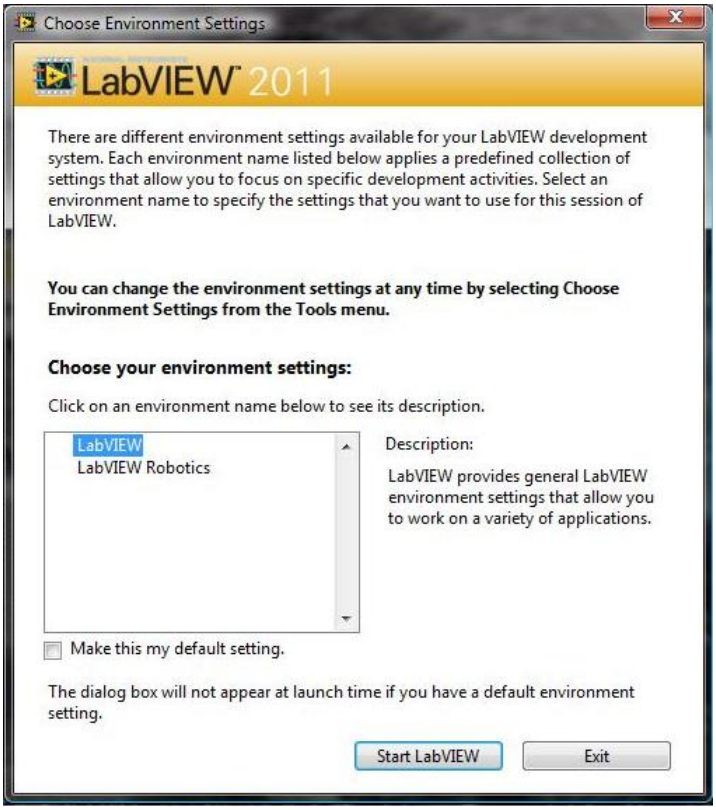


Figura 4-19. Ventana de selección de ambiente

Después se selecciona el *Hardware Setup Wizard* en la ventana *Getting Started* (Figura 4-20).

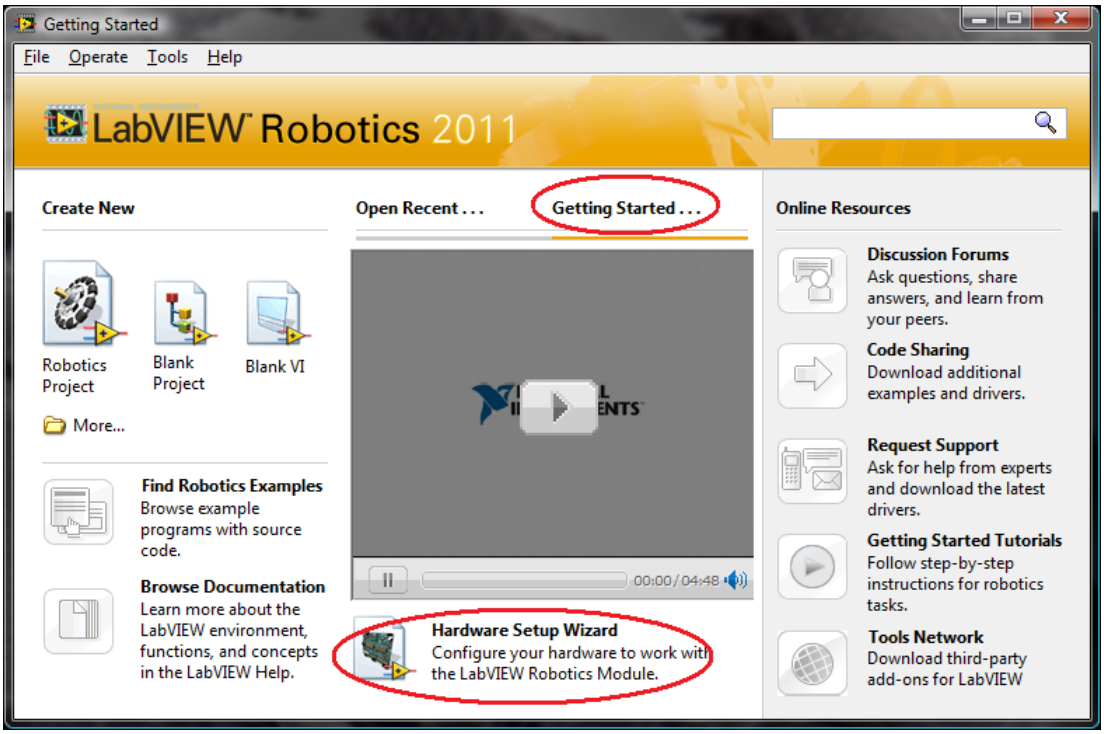


Figura 4-20. Ventana Getting Started

El *Hardware Setup Wizard* guía a través de una serie de pasos para configurar al robot DaNI en la computadora *host.* Como se observa en la Figura 4-21. Primero se selecciona el tipo de hardware que se desea configurar, en este caso corresponde al *Starter Kit 2.0*.

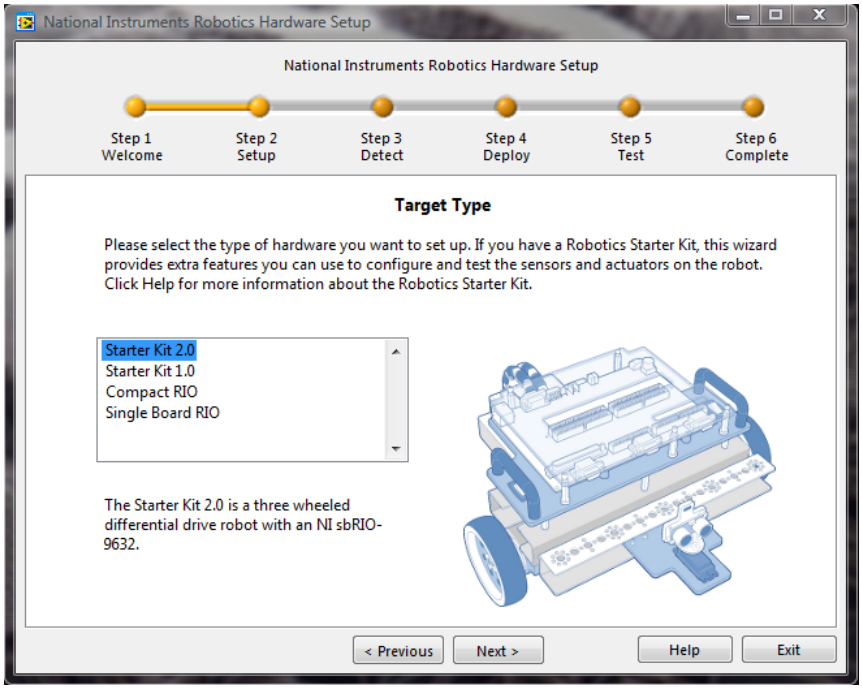


Figura 4-21. Selección de hardware a configurar

Una vez seleccionado el hardware se conecta el cable Ethernet y el cable de alimentación y se verifica que el led indicador de alimentación esta encendido (Figura 4-22).

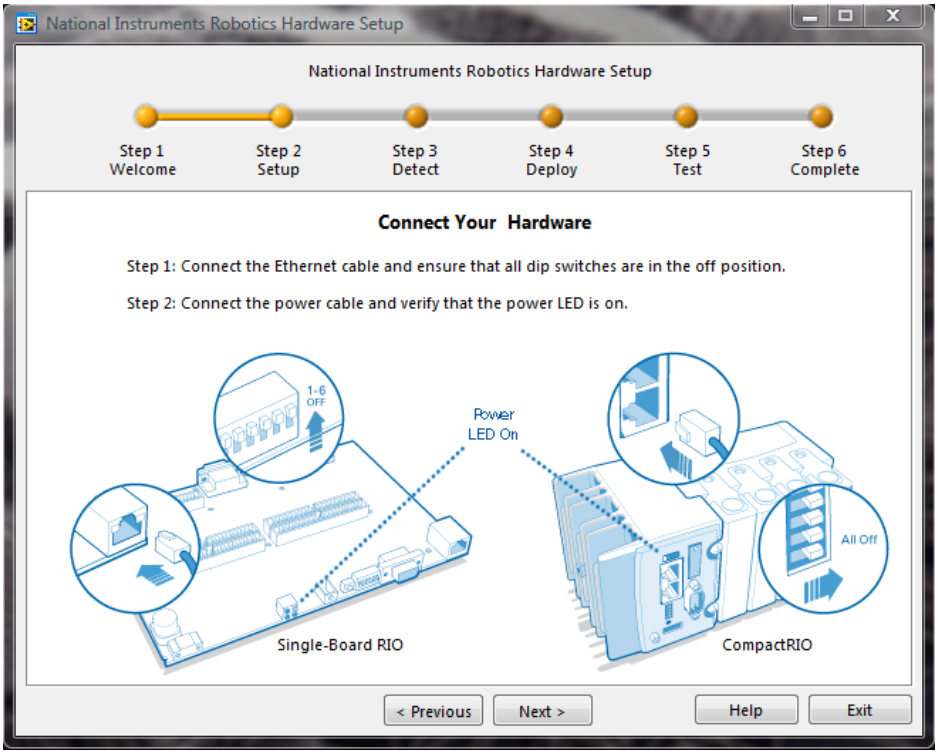


Figura 4-22. Pasos de la conexión del hardware

Posteriormente es necesario esperar unos segundos para que el robot DaNI sea detectado por la computadora. Una vez que el hardware es detectado automáticamente se le asigna una *dirección IP* (Figura 4-23). Hay que verificar que se ha seleccionado el dispositivo con el número de serie correcto. El número de serie está impreso en una pequeña calcomanía verde en la esquina inferior derecha de la tarjeta sbRIO-9632.

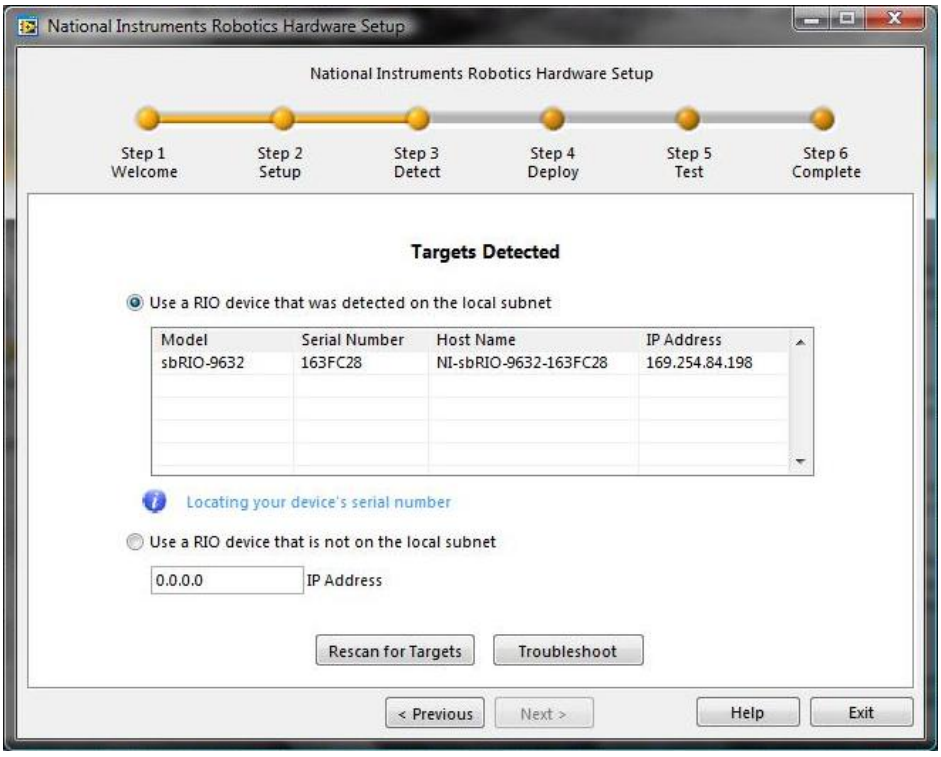


Figura 4-23. Dispositivo hardware detectado

Después el *Wizard* instalará el software y copiará archivos sobre el dispositivo sbRIO-9632 como se ve en la Figura 4-24. Este proceso toma algunos minutos.

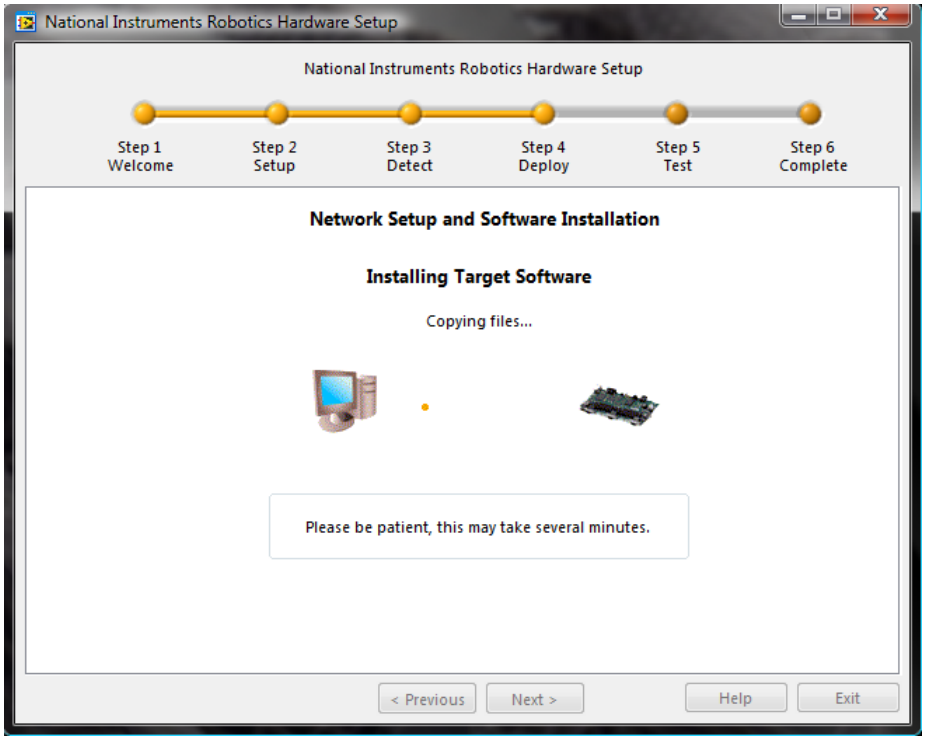


Figura 4-24. Instalación del software en el dispositivo a configurar

Una vez que el software es instalado, aparece la ventana (Figura 4-25) y la manera de comprobar que el robot DaNI está conectado correctamente.

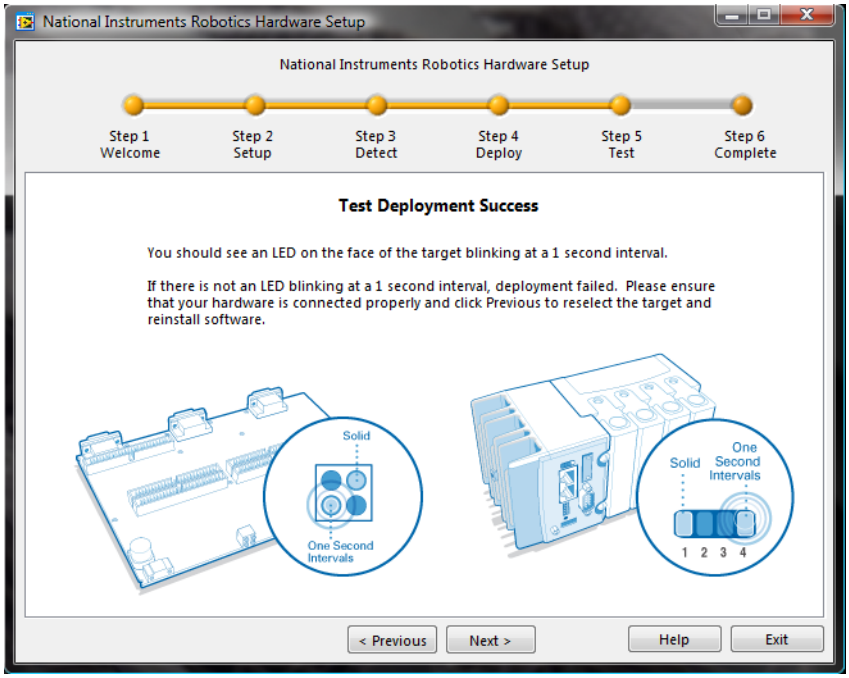


Figura 4-25. Cuando se instala exitosamente el software en el dispositivo debe parpadear el led indicador

Posteriormente, el *Wizard* mostrará la ventana (Figura 4-26) donde se calibra la posición de 0° del servo en el cual está montado el sensor ultrasónico PING))).

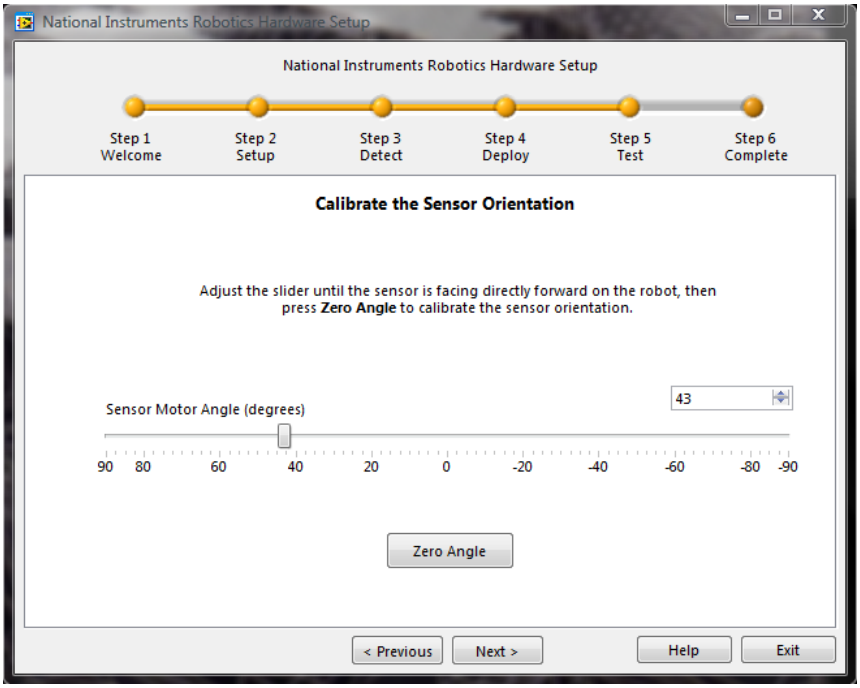


Figura 4-26. Calibración del offset del servomotor del sensor ultrasónico

Una vez calibrado el servo, se prueba que el sensor PING)) está funcionando y detecta cuando se coloca un objeto frente al él (Figura 4-27).

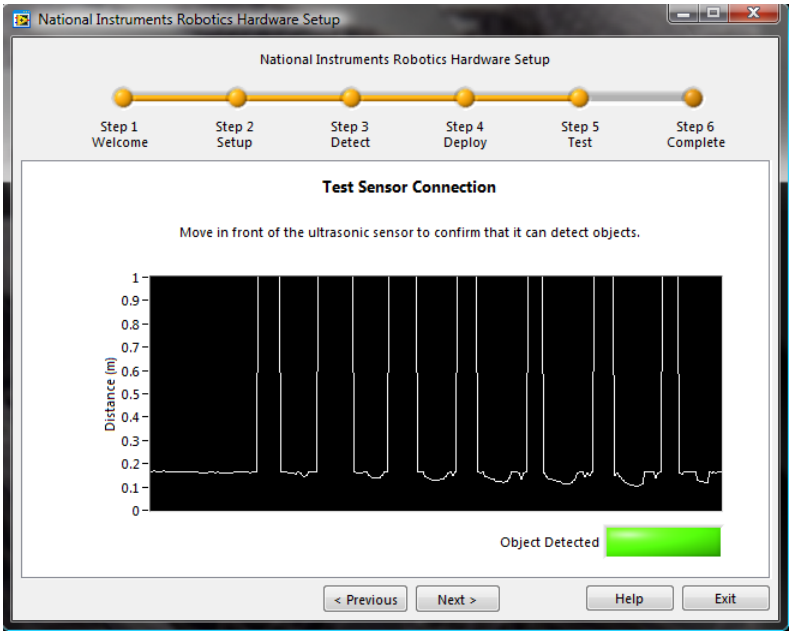


Figura 4-27. Prueba del sensor ultrasónico

A continuación aparecerá la ventana (Figura 4-28) para verificar que los dos motores actúan correctamente, en donde se escribe una velocidad en RPM y un sentido de giro (positivo o negativo según las manecillas del reloj) para activar los motores, y se comprueba que la velocidad leída por los encoders *tiende* a la velocidad escrita en los motores (set point), esto se aprecia en la gráfica incluida en la ventana.

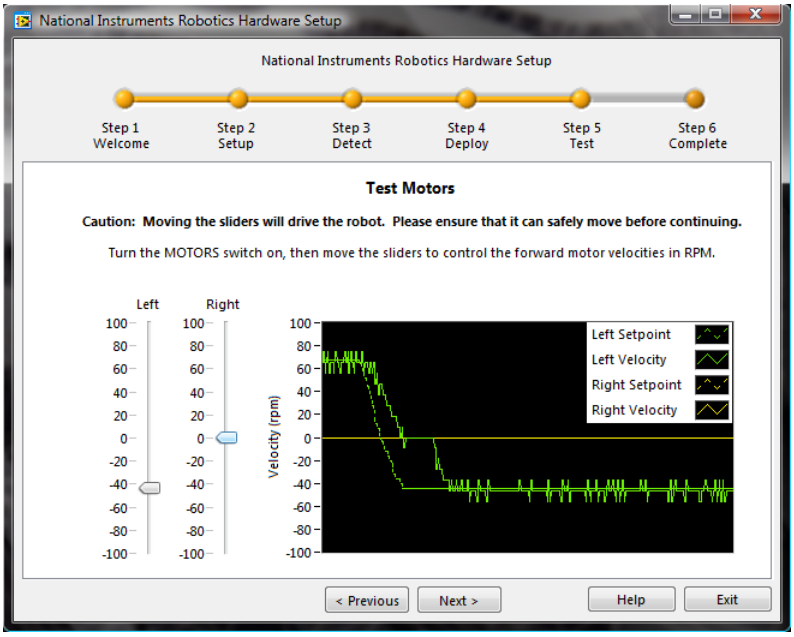


Figura 4-28. Prueba de los motores

Al final del proceso aparecerá la ventana (Figura 4-29) con el nombre del dispositivo configurado que en este caso es la tarjeta sbRIO-9632 del robot DaNI, además de la *dirección IP*. Es importante anotar y guardar la dirección IP ya que cada vez que se cree un proyecto de LabVIEW con el robot DaNI en la computadora *host* donde acaba de ser configurado, será necesaria esta dirección., además de que ya no será necesario realizar todos los pasos de configuración descritos.

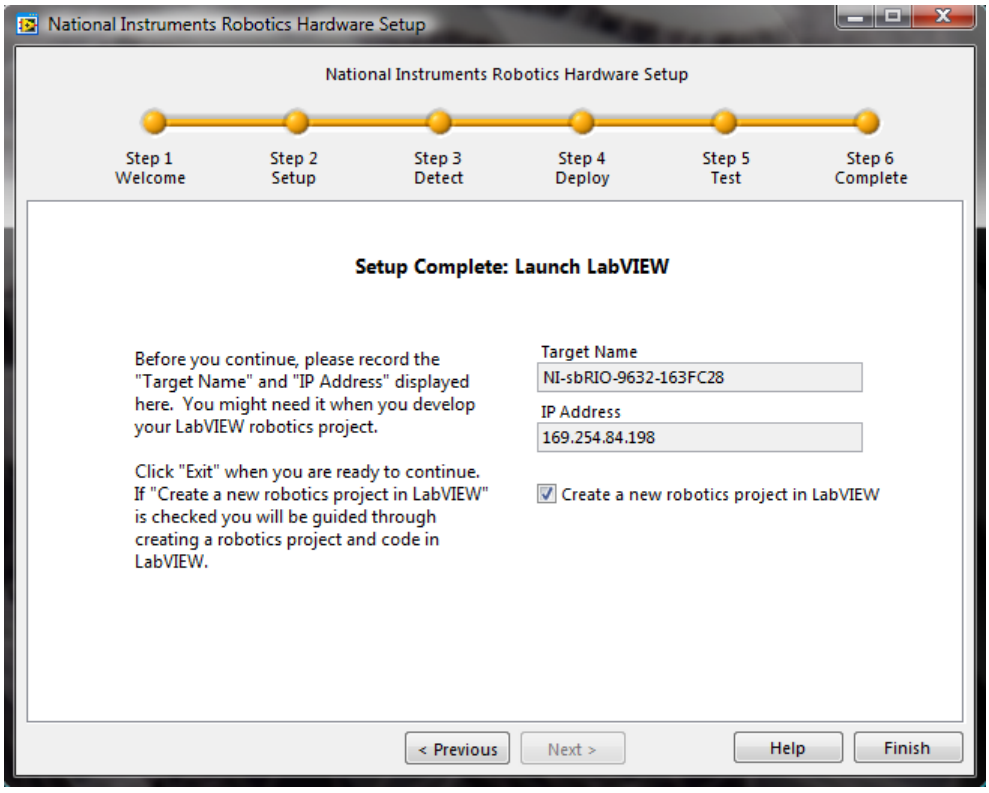


Figura 4-29. Configuración del robot DaNI en el *host* completada

Capítulo 5

Desarrollo de la programación para la navegación autónoma

5.1 Descripción general

El desarrollo de la programación para la navegación autónoma del robot se organiza en las tres etapas indicadas en la Figura 5-1. Cada etapa se divide en tareas, las cuales al cumplirse en conjunto permiten la navegación autónoma del robot DaNI. Las tareas que se llevan a cabo son: *percepción, localización, planeación y reacción.*

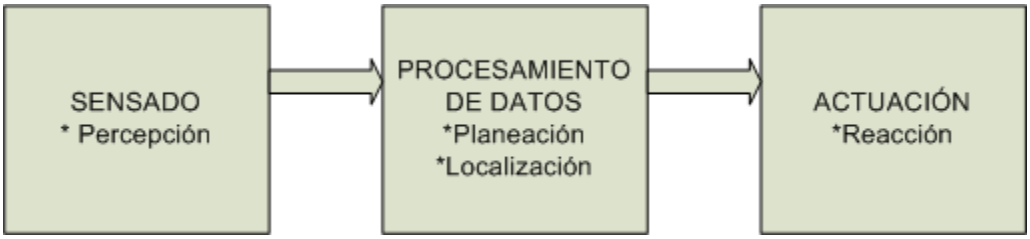


Figura 5-1. Etapas en las que se divide la actividad de navegación autónoma del robot

Este enfoque por tareas facilita la comprensión de todos los pasos necesarios para lograr la navegación autónoma, y cómo interaccionan los procesos involucrados. A su vez, para el cumplimiento de dichas tareas cada etapa está compuesta por varios *ciclos o procesos*.

Dado que en el lenguaje de programación de LabVIEW, el orden de ejecución está determinado por el flujo de datos entre los nodos de un programa, la aplicación desarrollada para la navegación autónoma del robot se puede ver como un flujo de datos entre los diversos procesos que componen el programa general, como se aprecia en la Figura 5-2, en donde los bloques representan los procesos y las flechas contienen el flujo de datos de las variables que comparten.

La velocidad de respuesta del robot depende del tiempo entre cada iteración de los diferentes procesos involucrados en las tres etapas descritas, debido a esto el programa realizado tiene una ejecución en forma *paralela y asíncrona*, esto quiere decir que los diferentes procesos se ejecutan simultáneamente y a diferentes velocidades.

Los procesos se componen básicamente por ciclos que se encuentran como subrutinas dentro de subVIs que son llamados desde el programa principal como se muestra en la Figura 5-3. Esta característica de subrutinas otorga al programa resultante una mayor limpieza de código que permite localizar fallas más fácilmente.

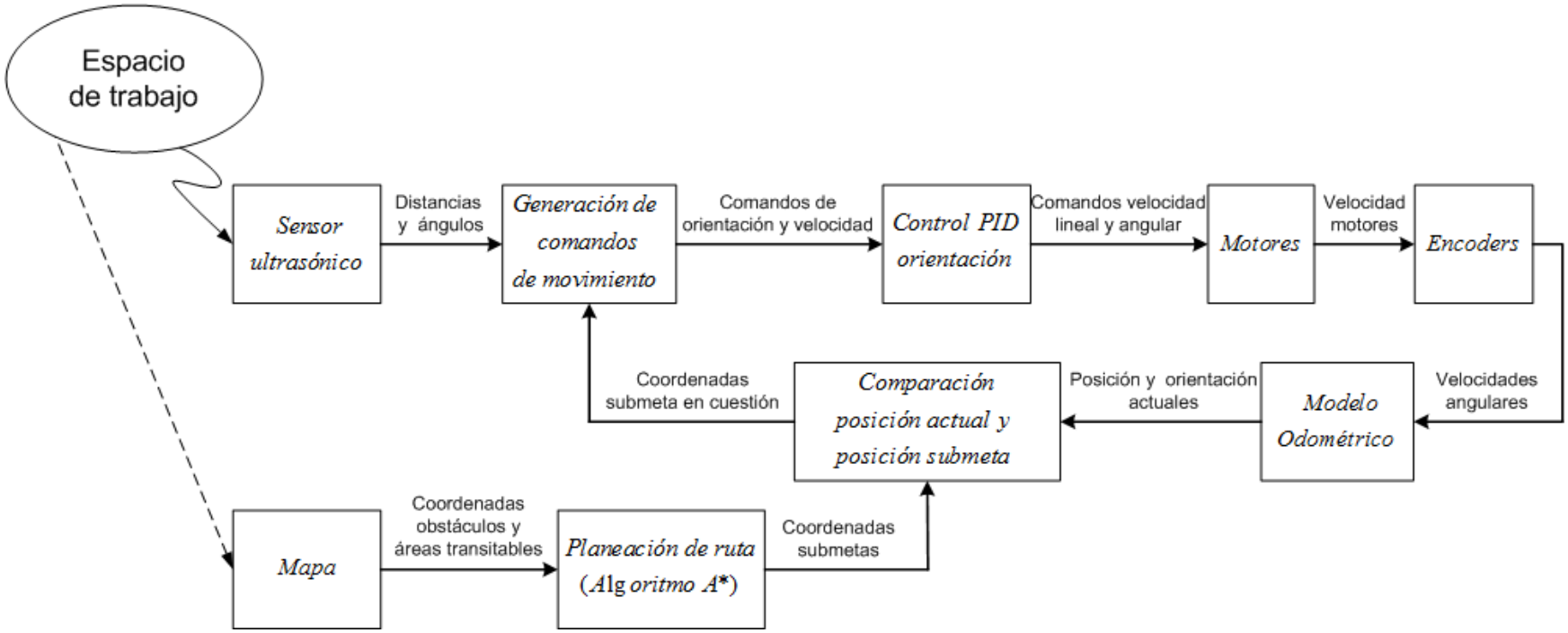


Figura 5-2. Diagrama de bloques para la navegación autónoma del robot y flujo de variables.

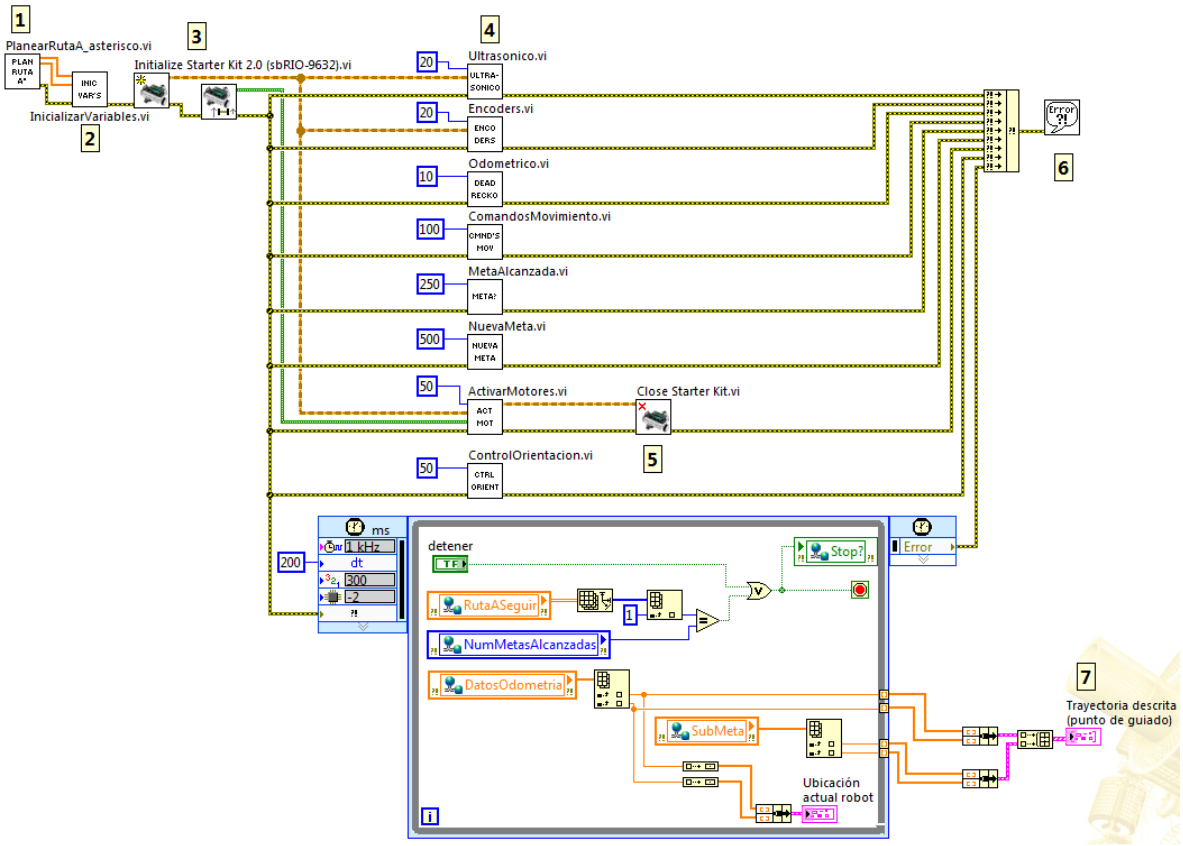


Figura 5-3. Diagrama de bloques VI principal

El funcionamiento general del programa principal el cual se muestra en la Figura 5-3, es el siguiente:

En (1) se planea la ruta a seguir, en (2) se inicializan algunas variables que se utilizarán entre los procesos, en (3) se crean las referencias del robot lo cual permite establecer comunicación entre la tarjeta sbRIO-9632 del robot con los sensores y los actuadores, además de establecer el tipo de direccionamiento del robot DaNI. Posteriormente en (4) se ejecutan los subVIs correspondientes al cumplimiento de las tareas necesarias para la navegación autónoma. Una vez finalizada la navegación del robot, en (5) se cierra la comunicación de la tarjeta con los sensores y actuadores, en (6) se revisa en caso de haber algún error, además en (7) se muestra la trayectoria descrita por el robot al ir avanzando, esto último se observa únicamente si el robot permanece conectado a la computadora huésped (*host*). Este VI principal se ha nombrado *Navegación Autónoma.*

Cada subVI tiene una condición de paro que detiene su ejecución como se muestra en la Figura 5-4, esta condición se presenta cuando existe un error en la ejecución del subVI o cuando se acciona el paro general.

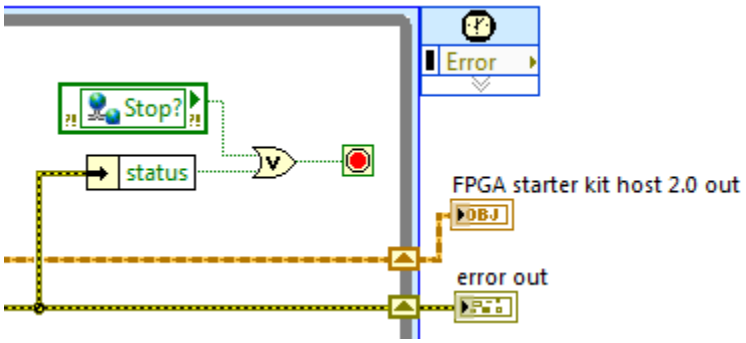


Figura 5-4. Condición de paro de los SubVIs

A su vez el paro general se acciona cuando se presiona el botón de stop en el panel frontal del VI principal (siempre y cuando el robot esté conectado al huésped) o cuando se alcanza la posición de meta. Para que el paro global detenga todos los procesos se creó una *variable compartida* de nombre *Stop?* como se puede ver en la Figura 5-5.

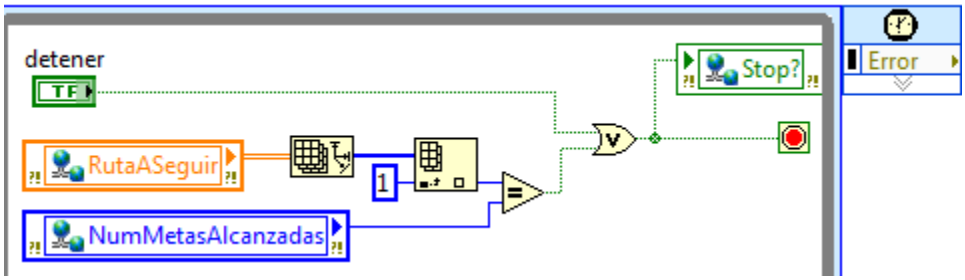


Figura 5-5. Condiciones que accionan el paro global

Todos los ciclos o procesos se comunican entre sí a través del concepto de *variables compartidas (Shared Variables)* utilizado LabVIEW. Este tipo de variables pueden ser compartidas entre los diferentes subVIs dentro de un proyecto, e incluso entre diferentes objetivos (*targets*) como puede ser una computadora y una FPGA o un controlador en tiempo real, como lo es en esta aplicación.

Para la llevar a cabo la programación en su conjunto se creó el proyecto de LabVIEW mostrado en la Figura 5-6, donde se encuentran ordenados los VIs creados y las herramientas de los módulos utilizados.

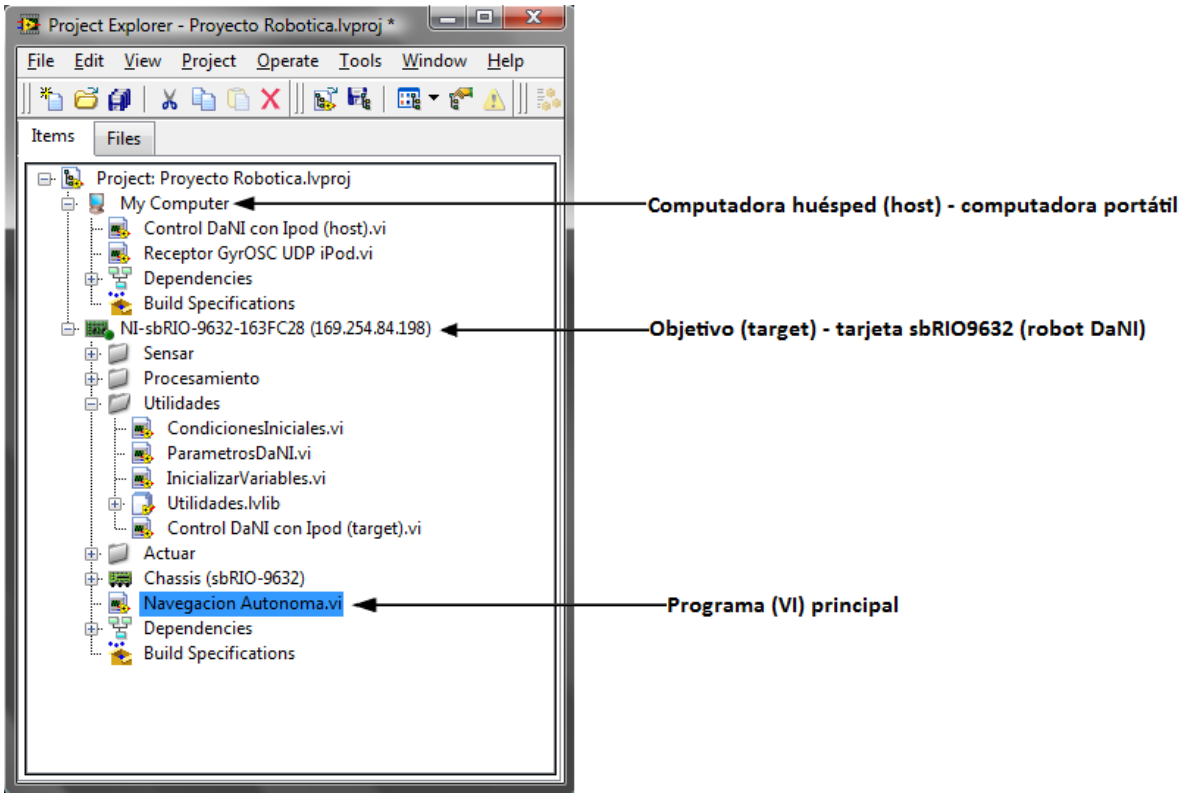


Figura 5-6. Proyecto creado para la navegación autónoma del robot DaNI

En la Figura 5-6 se pueden ver algunos de los VIs creados ordenados por carpetas, todos los VIs debajo del target (robot DaNI) se ejecutan en la tarjeta sbRIO-9632.

5.2 Percepción

Una de las tareas más importantes en un sistema de robótica es obtener información acerca del entorno en el que se desplaza el robot, esto se logra adquiriendo las señales provenientes de los sensores con que se cuenta.

5.2.1 Sensado del entorno frente al robot

En este caso para percibir el espacio inmediato frente al robot y detectar si existen obstáculos para evadirlos se utilizó el sensor ultrasónico incorporado. Como se describió en el tema 4.4 este sensor está montado sobre un servomotor (ver Figura 5-7) el cual le permite hacer un barrido del espacio que se encuentra frente al robot.

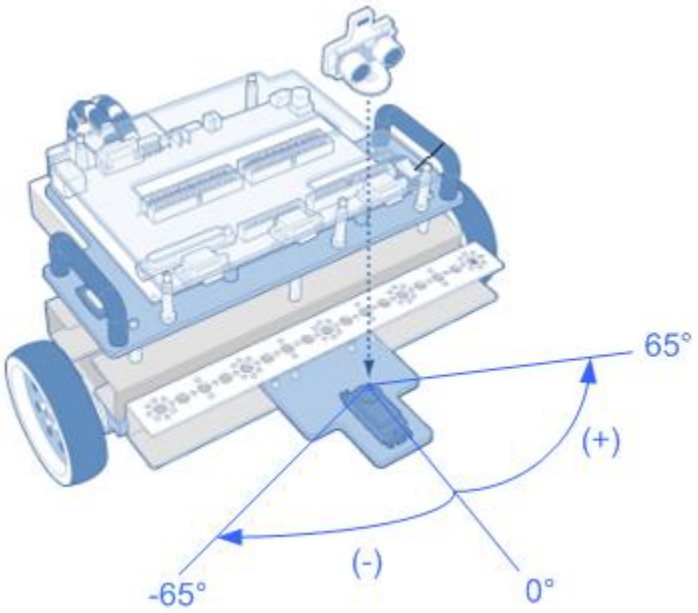


Figura 5-7. Sentido de giro del sensor ultrasónico montado sobre el servo

Las mediciones del sensor PING))) se hacen a intervalos de 4°, de esta forma a cada una de las orientaciones le corresponde un valor de distancia hasta un objeto, en caso de no detectar un objeto en esa dirección se toma un valor de 3m que es el valor máximo de distancia que puede ser medido por el sensor.

Para los incrementos de los ángulos de orientación del sensor PING))) se utiliza un *Shift Register* dentro de un ciclo temporizado *(Timed Loop)* como se puede observar en la Figura 5-8, de esta manera a partir de la orientación inicial del sensor (0°) se van haciendo los incrementos a intervalos de 4°, este valor se define antes de empezar el ciclo, así como el valor máximo y mínimo del ángulo del servo. Las mediciones se logran escribiendo en el servo el valor del ángulo y leyendo el valor entregado por el sensor PING))), estos dos valores de distancia y su respectivo ángulo de orientación se guardan en pares en un arreglo.

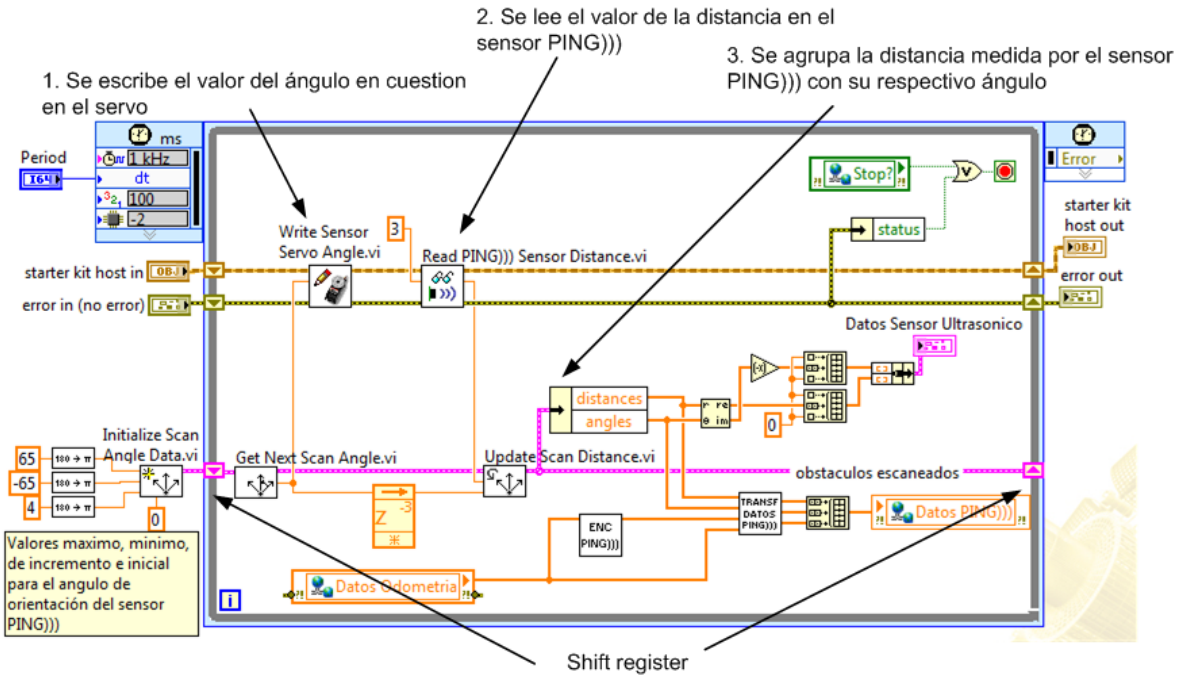


Figura 5-8. Programa para la percepción del robot DaNI

Antes de enviar los datos recabados a los demás procesos se ubica la posición del sensor PING))) en la estructura del robot tomando como referencia el punto de guiado del robot el cual corresponde al origen del sistema de referencia local del robot[3]. Además se considera la orientación del robot, de esta manera se toma en cuenta en los demás procesos que los datos recabados por el sensor PING))) corresponden a una orientación particular del robot. Una interpretación de los datos recabados por el sensor se muestra en la Figura 5-9, la cual es una representación en forma polar, para esto se utiliza la distancia medida por el sensor PING))) como radio vector y el ángulo al cual fue medido como el ángulo de este radio vector.

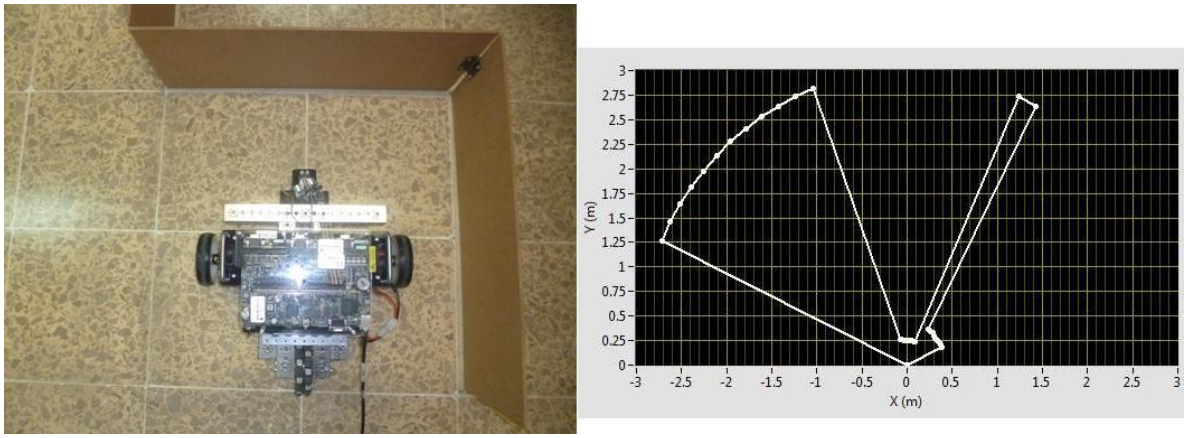


Figura 5-9. Representación de los obstáculos

[3] Los sistemas de referencia se presentan en el tema 2.2

En la Figura 5-9 se observa la representación de obstáculos en diferentes direcciones. La distancia de separación del sensor ultrasónico y la pared frente a él es de aproximadamente 25cm, este valor se puede apreciar en la gráfica del lado derecho.

El ciclo temporizado (*Timed Loop)* tiene un periodo de sensado del espacio frente al robot de 20ms, este es el valor mínimo ya que valores menores hacen que el movimiento del servo se reinicie antes de terminar de hacer todo el barrido. Los datos recabados se escriben en la variable *Datos PING)))* y se envían al proceso de *generación de comandos de movimiento,* esta variable contiene la magnitud de la distancia a la que se encuentran los obstáculos con su respectiva dirección, además de la posición y orientación del robot para las cuales se hizo la medición. Los datos de la posición y orientación del robot provienen de la estimación odométrica, la cual se explica en el tema 5.3.

El programa elaborado se guardó con el nombre de *Ultrasonico* como se muestra en la Figura 5-10*,* este VI se llama como subrutina en el programa principal (ver Figura 5-3). Las entradas de este VI son la referencia del robot DaNI (*starter kit host in*), el periodo de ejecución del VI, y la referencia de error *(error in)*, las salidas son la referencia de error y la del robot.

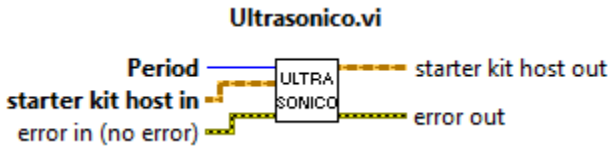


Figura 5-10. Apariencia del programa encapsulado en el bloque Ultrasonico.vi

5.2.2 Sensado de la velocidad de las ruedas

Para medir la velocidad de las ruedas se utilizan la información proveniente de los codificadores ópticos de cuadratura. El ciclo encargado de la lectura de las velocidades (ver Figura 5-11) se ejecuta cada 20ms.

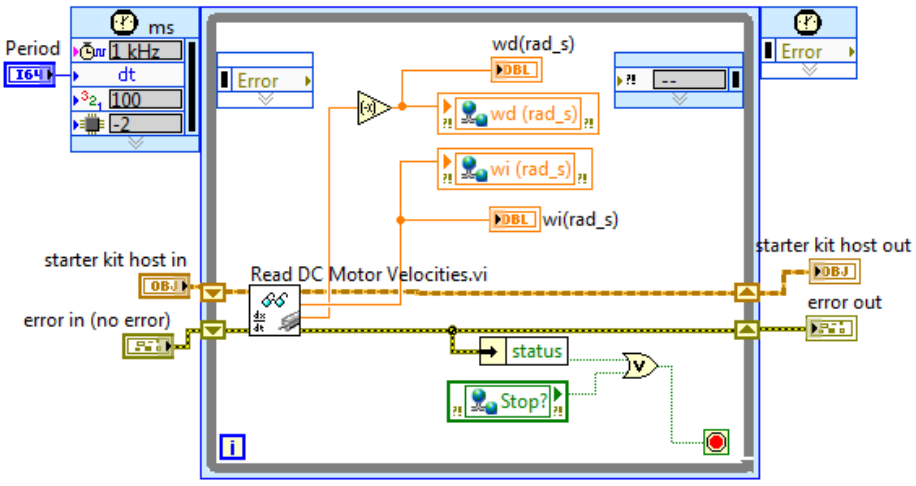


Figura 5-11. Ciclo Timed Loop para la lectura de los encoders

Las velocidades de las ruedas izquierda y derecha en unidades de (rad/s) se escriben en las variables *wi(rad_s) y wd(rad_s)* respectivamente. El proceso de lectura de los codificadores ópticos se guardó como subrutina en el subVI *Encoders* (Figura 5-12).

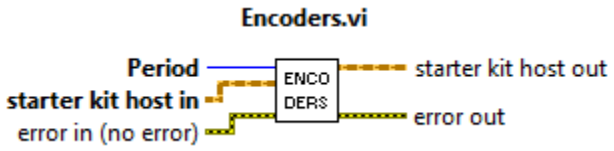


Figura 5-12. Apariencia como bloque del programa de sensado de los codificadores ópticos

5.3 Localización

Es indispensable para cualquier robot autónomo conocer su ubicación en su entorno en todo momento, es por eso que localizar su posición dentro de un sistema global de referencia es una tarea necesaria para su navegación. Para lograrlo se utilizó el *modelo odométrico* o *estimación de la posición y orientación* (ver tema 2.5), la técnica basada en este modelo utiliza la cinemática del robot, donde a partir de una posición y orientación iniciales se puede estimar la posición y orientación instantáneas del robot usando la información proveniente de los codificadores ópticos acoplados a las ruedas actuadas.

Es conveniente recordar las ecuaciones (2.45) obtenidas en el desarrollo hecho en el tema 2.5:

$$\begin{bmatrix} x \\ y \\ \emptyset \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ \emptyset_0 \end{bmatrix} + \begin{bmatrix} \int_0^t \frac{-r \sin \emptyset}{2} (\omega_d + \omega_i) d\tau \\ \int_0^t \frac{r \cos \emptyset}{2} (\omega_d + \omega_i) d\tau \\ \int_0^t \frac{r}{b} (\omega_d - \omega_i) d\tau \end{bmatrix} \tag{5.1}$$

En el caso del robot DaNI: $r = 0.0508[m]$ y $b = 0.3302[m]$, estos valores se guardaron como *variables globales* (ver tema 3.2.1) en el proyecto. Las velocidades angulares provenientes del proceso de sensado de la velocidad de las ruedas izquierda y derecha: ω_i y ω_d , se obtienen leyendo las variables *wi(rad_s)* y *wd(rad_s)*. Como se mencionó en el tema 5.2.2 estas variables tienen la información de los encoders en unidades de rad/s.

El ángulo $\emptyset$ corresponde a la orientación del robot, y las coordenadas (x, y) corresponden a la posición del punto de guiado p como se observa en la Figura 5-13.

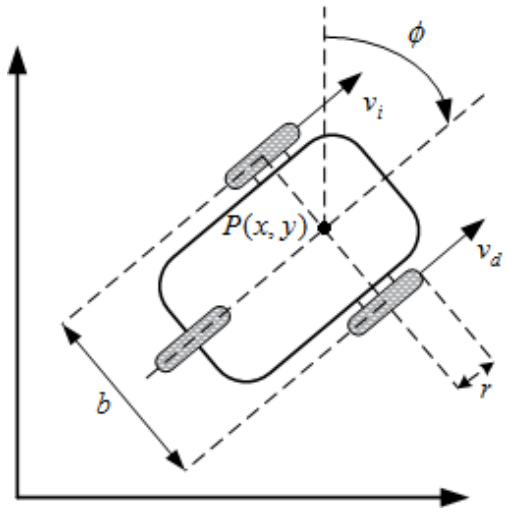


Figura 5-13. Parámetros del direccionamiento diferencial

En la ecuación (5.1) el vector $p_0 = [x_0 \quad y_0 \quad \emptyset_0]^T$ tiene las condiciones iniciales que le indican al robot la posición de donde parte y la orientación con que lo hace. Para esto se utilizan tres variables: *x(0), y(0),* y *phi(0)*°, en las cuales se guardan los valores de la posición y orientación iniciales, estos valores se escriben en el proceso de inicialización de variables antes de que el robot inicie su movimiento.

Para implementar en el robot la ecuación (5.1) se utilizó la estructura *Control & Simuation Loop* mostrada en la Figura 5-14, esta estructura es un ciclo que ejecuta el diagrama en su interior hasta que se alcanza el tiempo final de la simulación o hasta que la función *Halt Simulation* detiene la ejecución.

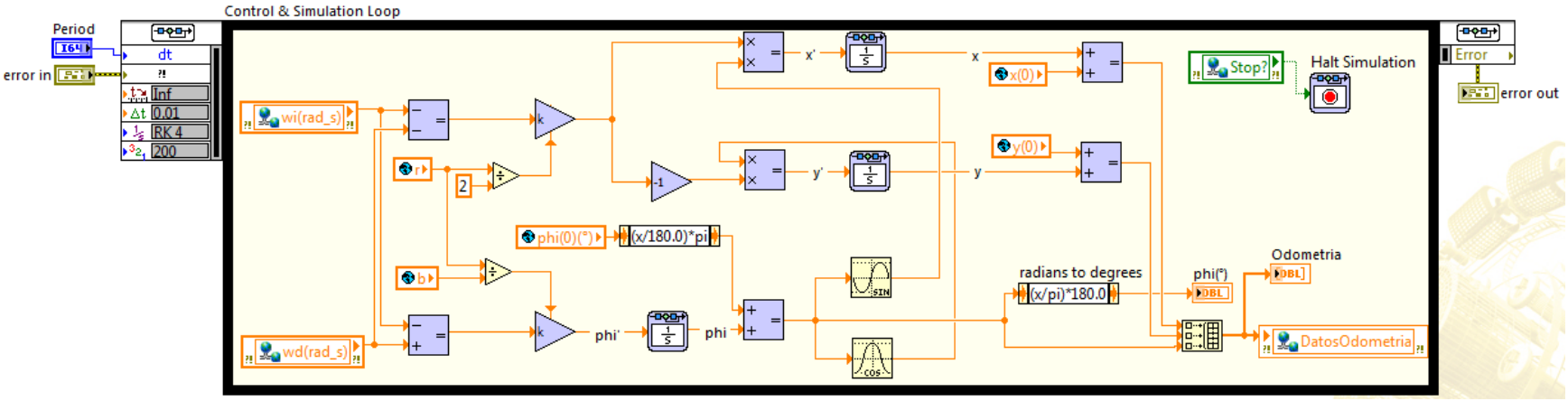


Figura 5-14. Implementación de las ecuaciones (5.1) en el robot

Los valores de posición y orientación resultantes de este proceso se guardan como un arreglo de tres elementos en la variable compartida *Datos Odometria.*

Para el tiempo final de simulación se puso un valor infinito, de esta manera el ciclo se ejecuta en todo momento y solo se detendrá cuando exista un error en su interior o cuando se accione la función *Halt Simulation* vía el paro general. Este valor para el tiempo de simulación junto con el Método de solución (*ODE Solver),* y el tamaño de escalón (*Step Size)* se configura en la pestaña *Simulation Parameters* (Figura 5-15) en la ventana *Configure Simulation Parameters* la cual aparece al dar doble click sobre el nodo izquierdo del ciclo.

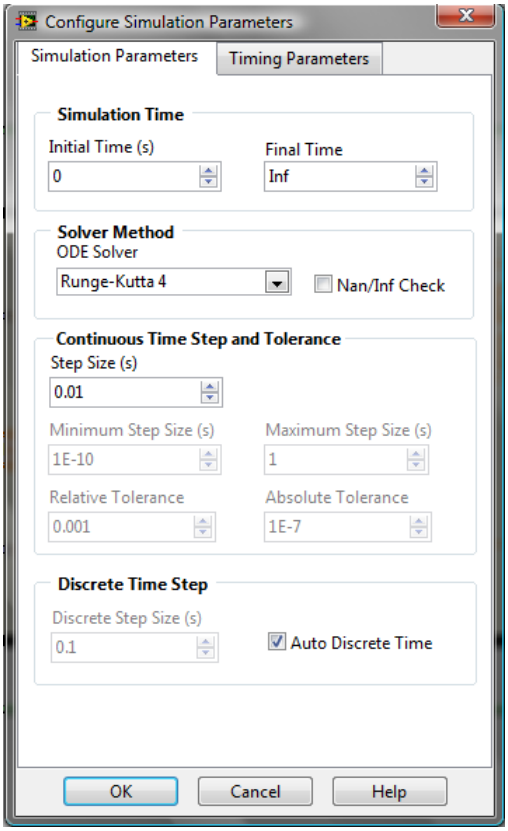


Figura 5-15. Parámetros de simulación del *Control & Simulation Loop*

Los parámetros de tiempo para las iteraciones y el periodo entre iteraciones del ciclo se configuran en la pestaña *Timing Parameters* (Figura 5-16).

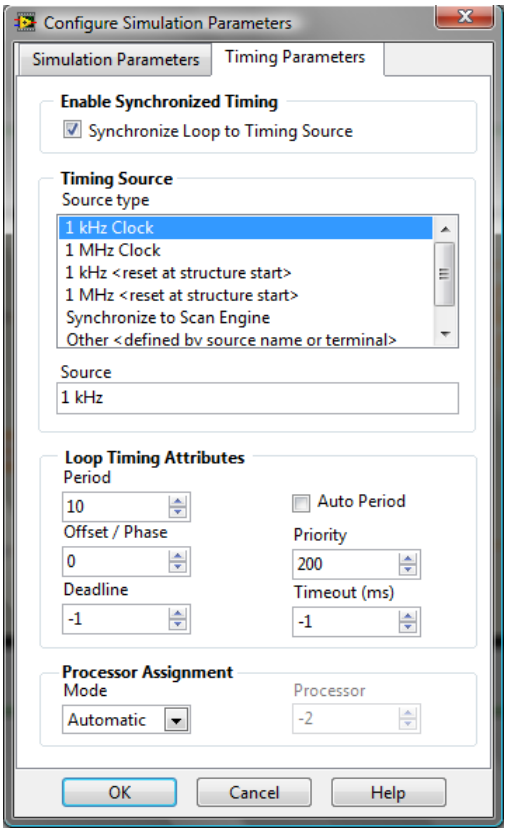
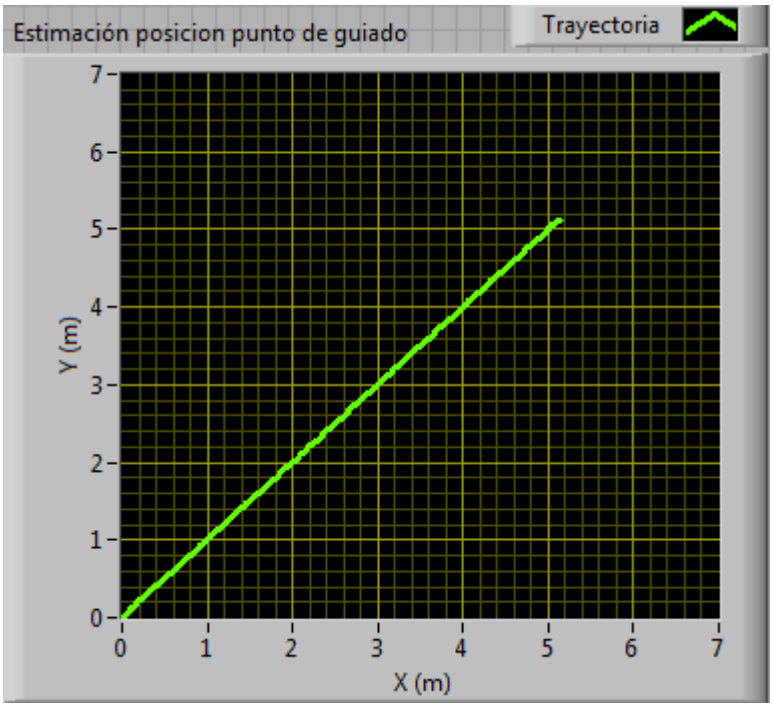


Figura 5-16. Parámetros de tiempo del *Control & Simulation Loop*

El reloj de la tarjeta sbRIO-9632 se utiliza como la referencia de tiempo con una frecuencia de $1kHz$, el periodo entre iteraciones es de 9ms**.** Este valor para el periodo se estableció con base en las pruebas experimentales explicadas más adelante en el tema 6.1.

De esta manera, con el programa mostrado en la Figura 5-14 el robot puede estimar en todo momento en donde se encuentra dentro del sistema global de referencia.

Por ejemplo, si el robot parte de la posición $(0,0)$ con una orientación de $-45°$ y ambas ruedas giran con la misma magnitud de velocidad angular. Graficando las posiciones que recorre el robot se obtiene la trayectoria descrita por el punto de guiado como se muestra la Gráfica 5-1.



Gráfica 5-1. Trayectoria del punto de guiado

El proceso de localización del robot DaNI mediante la estimación odométrica se guardó en el VI *Odometrico* (Figura 5-17).

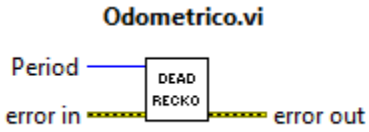


Figura 5-17. Apariencia como bloque del VI *Modelo Odometrico*

Es importante señalar que para poder utilizar el *Control & Simulation Loop* y las funciones que van dentro de este ciclo, es necesario instalar en el robot DaNI el *toolkit Control Design & Simulation,* ya que este toolkit no se instala por defecto con el procedimiento de configuración del robot DaNI descrito en el tema 4.5. En el apéndice A se explica cómo instalar manualmente este toolkit en el robot DaNI.

5.4 Planeación

Como se mencionó, el objetivo de este trabajo es que el robot alcance una posición de meta a partir de una posición y orientación iniciales conocidas, entonces la planeación se refiere a establecer en el robot DaNI el objetivo y la forma en cómo se debe alcanzarlo.

En la Figura 5-18 se puede observar que en primera instancia el robot debe de planear una ruta dentro del entono, de tal suerte que al seguir esta ruta se llegue a la posición de meta a partir de su posición inicial. Para trazar esta ruta, previamente se le proporciona al robot DaNI información sobre el entorno en el que se va a mover.

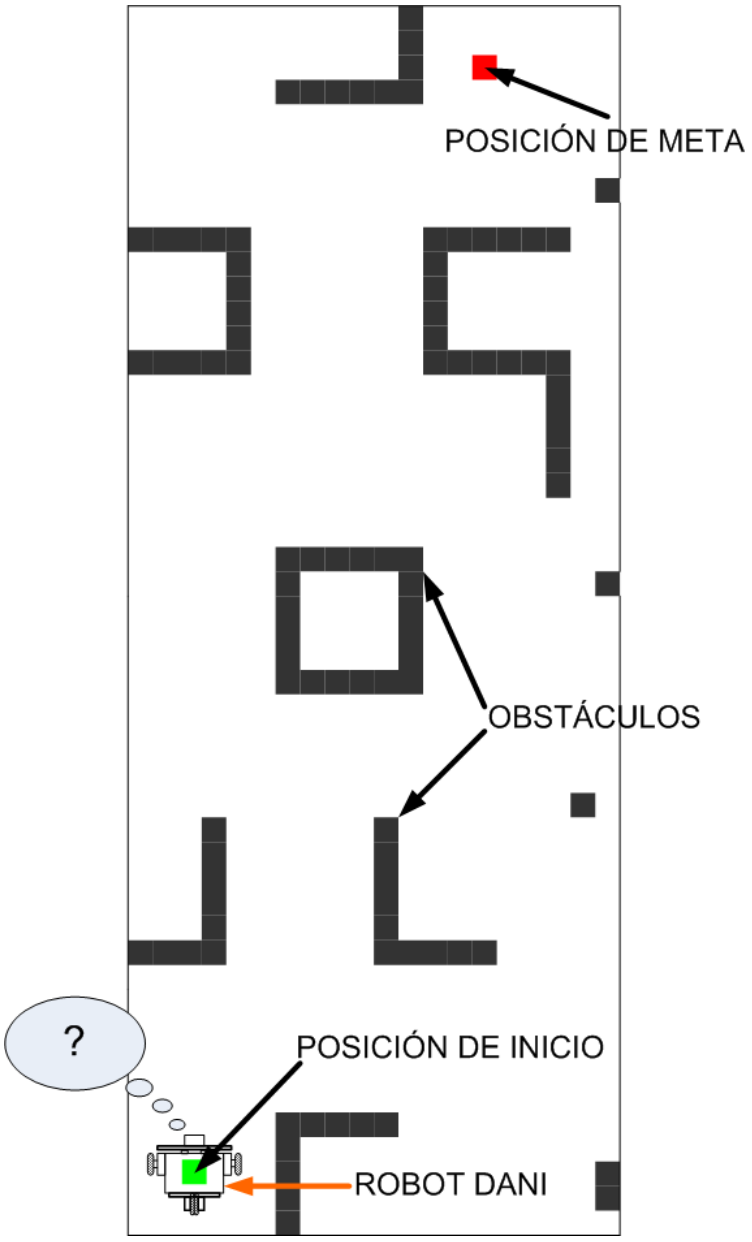


Figura 5-18. Misión del robot DaNI. El robot debe de planear como llegar hasta la meta

Esta información previa es: su posición y orientación inicial, la posición de meta hasta donde debe llegar, y el mapa del ambiente en donde se moverá. La manera en que se desarrolló esta parte de la planeación consiste en hacer que el robot siga la ruta que previamente calculó con la información del mapa. Para esto se ha empleado el algoritmo A*, el cual está incluido como una función dentro del módulo Robotics de LabVIEW (ver tema 3.3.1).

5.4.1 Algoritmo A*

El algoritmo de búsqueda A* (A estrella o A asterisco) es un algoritmo de búsqueda en grafos que encuentra la ruta de menor coste entre dos puntos siempre y cuando se cumplan una serie de condiciones. Las aplicaciones van desde videojuegos hasta software de dirección de manejo y control autónomo de vehículos. Al aplicar el algoritmo A* en el área de la robótica permite dar a los vehículos robóticos un sistema de navegación autónomo. En videojuegos se puede usar para determinar el recorrido que un objeto debe seguir; por ejemplo en Pacman se aplicaba al comportamiento de los fantasmitas, también se puede usar para resolver problemas de Sudoku de 8 variables, y para resolver el cubo Rubik mostrando cómo hacerlo en la menor cantidad de movimientos.

Para entender cómo funciona el algoritmo A* a continuación se verá un ejemplo sencillo:

Supongamos que el robot se encuentra en el punto A y debe moverse hasta el punto B por el camino o ruta más eficiente posible, además ambos puntos se encuentran separados por una pared como se ve en la Figura 5-19.

El primer paso para hallar la ruta a seguir es simplificar el área de búsqueda, en este caso se divide el área de búsqueda en una cuadrícula. Esta simplificación permite reducir el área de búsqueda a un arreglo de dos dimensiones (un *array* en LabVIEW), donde cada elemento del arreglo representa uno de los cuadros de la cuadricula los cuales pueden tener dos estados: *transitable o no transitable*. La ruta se encuentra al identificar que cuadros se deben tomar para ir del punto A al B, y posteriormente avanzando desde los puntos medios o *nodos* de cada cuadro a lo largo de la ruta. Cabe mencionar que es posible dividir el área de búsqueda en otras formas diferentes e incluso irregulares, pero en el caso de este ejemplo y para la navegación autónoma del robot DaNI se usan cuadros debido a que es más sencillo de implementar.

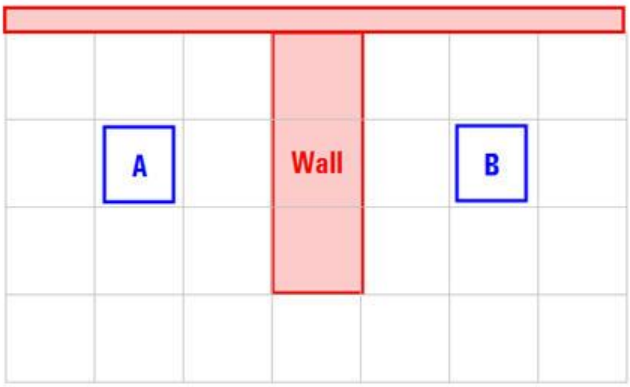


Figura 5-19. Punto inicial A y punto de meta B separados por una pared

Una vez simplificada el área de búsqueda en un número manejable de cuadros como se observa en la Figura 5-20, el siguiente paso es realizar una búsqueda para

encontrar la ruta más corta. Esto se hace empezando en el punto A y evaluando los nodos adyacentes para evaluar el mejor paso siguiente a tomar, esto se repite una y otra vez (iterando) hasta llegar al punto B.

La implementación del A* lleva a cabo esta búsqueda formando dos listas. La *lista abierta* es una lista con los nodos que se consideran en cada instante como potenciales candidatos para ser el siguiente paso a tomar. La *lista cerrada* es una lista con los nodos que ya han sido evaluados y que ya no es necesario considerarlos de nuevo.

Para empezar, en la primera iteración se agrega a la lista cerrada el nodo de inicio, se observan todos los nodos adyacentes al punto inicial, y se agregan todos los nodos transitables a la lista abierta mientras se ignoran los nodos que no son transitables. En el ejemplo todos los nodos adyacentes son transitables, entonces se agregan 8 nuevos nodos a la lista abierta, a estos nodos se les asigna un *nodo padre*, en este caso el punto A. En la Figura 5-20, se puede observar los nodos de la lista abierta delineados en color verde, y su padre se señala por flechas grises.

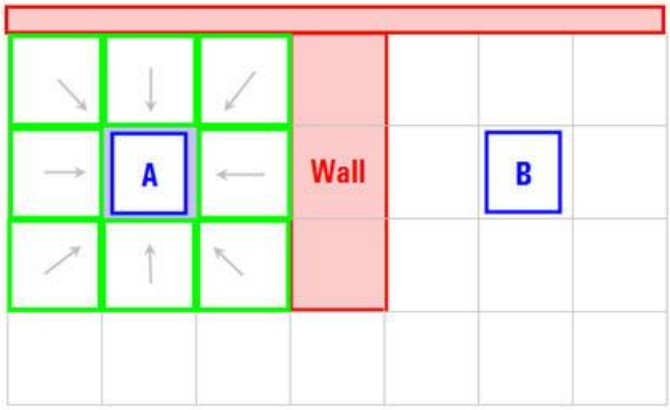


Figura 5-20. Primera evaluación de los nodos

Para saber que nodo debe seleccionarse en cada una de las iteraciones se utiliza la siguiente ecuación:

$$F = G + H \tag{5.2}$$

Donde G es el coste de movimiento de ir desde el punto inicial A hasta un cierto nodo actual de la rejilla.

H es el coste estimado de moverse desde un determinado nodo actual hasta el punto de destino B, comúnmente conocido como la heurística.

La ruta se genera al ir escogiendo una y otra vez el nodo de la lista abierta con la menor puntuación de F.

Para el caso de G en este ejemplo, se asigna un coste de 10 a los movimientos horizontales o verticales, y un costo de 14 a los movimientos diagonales.

En el caso de H este puede estimarse de diferentes maneras, LabVIEW utiliza el método Manhattan, donde se calcula el número total de nodos atravesados horizontal y verticalmente hasta alcanzar el nodo destino B desde el cuadro actual sin hacer movimientos en diagonal, e ignorando cualquier obstáculo en el camino, posteriormente se multiplica el total por 10.

El resultado de la primera iteración puede verse en la Figura 5-21, donde las puntaciones para F, G y H estan escritas para cada cuadro. En el nodo que esta resltado $G = 10$ porque esta a un solo nodo del nodo inicial A en dirección horizontal. Los nodos que están arriba, abajo, a la izquierda y a la derecha del nodo inicial tienen una puntuación de 10, los nodos en diagonal tienen una puntuación de 14.

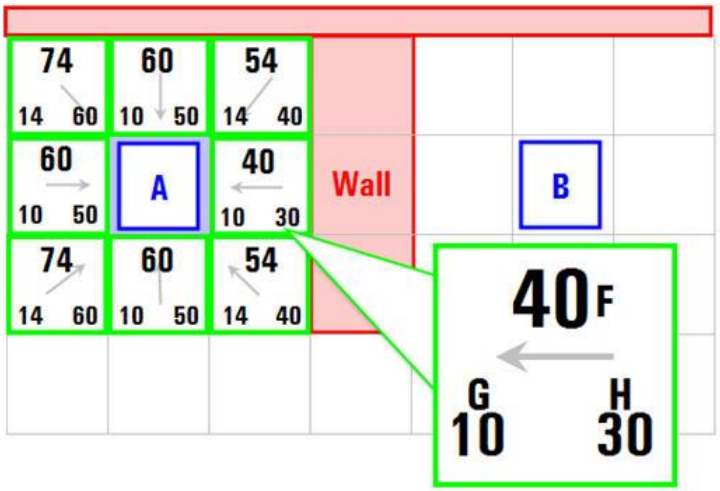


Figura 5-21. Costes resultantes de la primera evaluación.

Para las puntuaciones de H se utiliza el método Manhattan estimando la distancia hasta el nodo de meta B solo con movimientos verticales u horizontales e ignorando la pared en el camino. De esta manera, el cuadro a la derecha del inicial esta a 3 cuadros del cuadro B para una puntuación $H = 30$. El nodo arriba a la derecha del cuadro inicial A esta a 4 cuadros de distancia del cuadro B (recordando que son solo movimiento horizontales y verticales) para una puntuación de $H = 40$.

A continuación se escoge de la lista abierta el nodo con la puntuación más baja de F y lo movemos a nuestra lista cerrada, todos los nodos adyacentes a este que son transitables deben ser agregados a la lista abierta. Los nodos que están en la lista cerrada deben ser ignorados, los nodos que están en la lista abierta deben ser nuevamente evaluados ya que puede darse el caso que en futuras iteraciones se escoja un mejor camino.

En la Figura 5-22 se puede observar que de los 9 nodos iniciales, quedan 8 en la lista abierta (contorno verde), de estos el que tiene el menor costo de F es el que esta inmediatamente a la derecha del cuadro inicial A que tiene una puntuación F=40. Se selecciona este cuadro y se mueve de la lista abierta a la lista cerrada (es por eso que

ahora está coloreado azul), a continuación se evalúan todos los cuadros adyacentes a este, los que se encuentran a la derecha corresponden a la pared y deben ser ignorados, el que está a la izquierda es el cuadro inicial que se encuentra en la lista cerrada y también se debe ignorar.

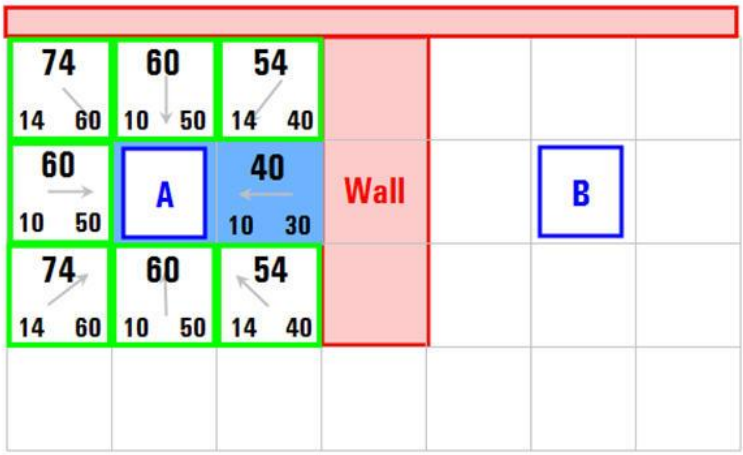


Figura 5-22. Elección de nodo según su puntuación de F

Los otros 4 cuadros ya están en la lista abierta, entonces será necesario evaluar si el camino a esos nodos es mejor que el camino que pasa por el nodo actual (azul) y que va a esos nodos. Para esto se usa el valor de G como referencia.

Por ejemplo, tomando el nodo situado por debajo del actual (azul) vemos que G=14, pero si nos movemos a través del nodo actual hasta ese nodo, G sería igual a 20, 10 del nodo actual más 10 de moverse verticalmente hacia abajo. Debido a que 20>14 no es buen camino. Dicho de otra manera es mejor moverse directamente a ese nodo que atravesar por el nodo actual.

Repitiendo este proceso para los demás nodos adyacentes de la lista abierta, se descubre que los demás caminos que van a esos nodos y atraviesan el nodo actual no disminuyen su coste (no son buenos caminos), entonces se ignora y no se modifica nada. Ahora que ya se han hecho todas las evaluaciones y se tiene un nuevo nodo y se procede a buscar en la lista abierta el mejor siguiente nodo.

Al repetir el proceso de búsqueda en la lista abierta (coste de F más bajo), se tendría un resultado como el de la Figura 5-23.

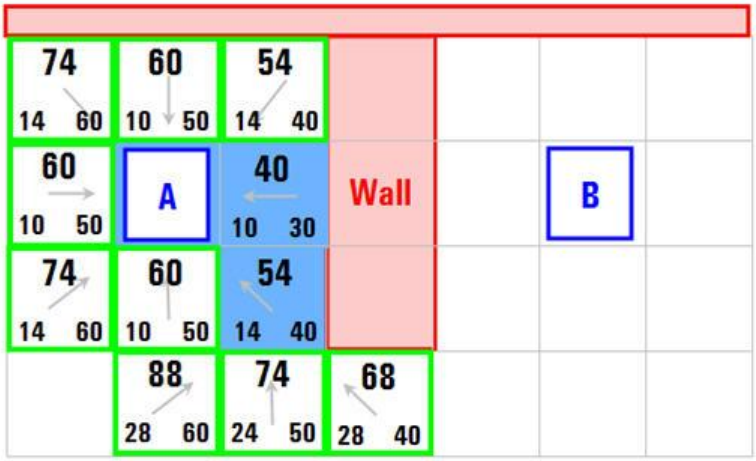


Figura 5-23. Siguiente nodo seleccionado con la puntuación más baja de F

Una vez más se repite la evaluación de los nodos adyacentes al recién adquirido nodo actual (coloreado azul con F=54). Los que están a la derecha corresponden al muro y se ignoran, al igual que el nodo inicial y el que está arriba por que se encuentran ya en la lista cerrada. Los tres nodos debajo del nodo actual no están en la lista abierta entonces se agregan a esa lista, y el nodo actual se convierte en su nodo padre. Para el nodo que está a la izquierda del actual se comprueba si el coste de G del camino que llega hasta ese nodo y atravesando por el nodo actual es menor al coste de G llegando directamente desde el nodo inicial hasta el. Al hacerlo no hay suerte entonces no se hace cambio alguno y una vez más pasamos a la lista abierta a seleccionar el mejor nuevo nodo.

El proceso se va repitiendo hasta que se añade el nodo de meta a la lista cerrada y finaliza la búsqueda como en la Figura 5-24.

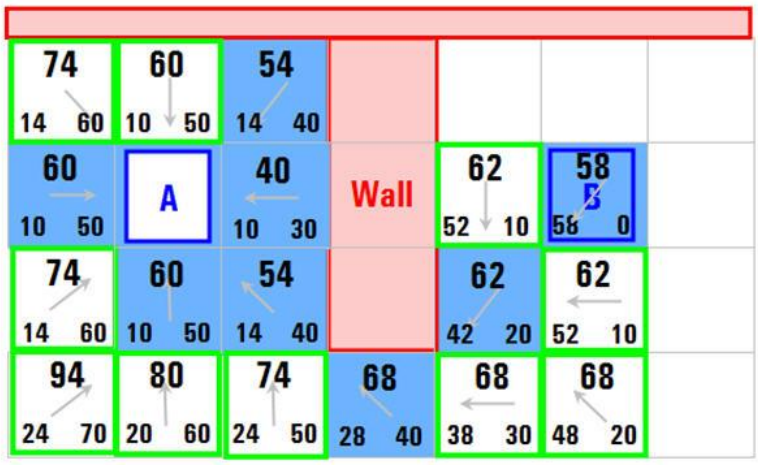


Figura 5-24. El nodo de meta ahora está en la lista cerrada

Finalmente para determinar la ruta, se empieza en el nodo de meta y nos vamos moviendo hacia atrás de un nodo hacia su nodo padre siguiendo las flechas como indica la Figura 5-25. El camino resultante corresponde a la ruta de más bajo coste que va del nodo de inicio A al nodo de meta B.

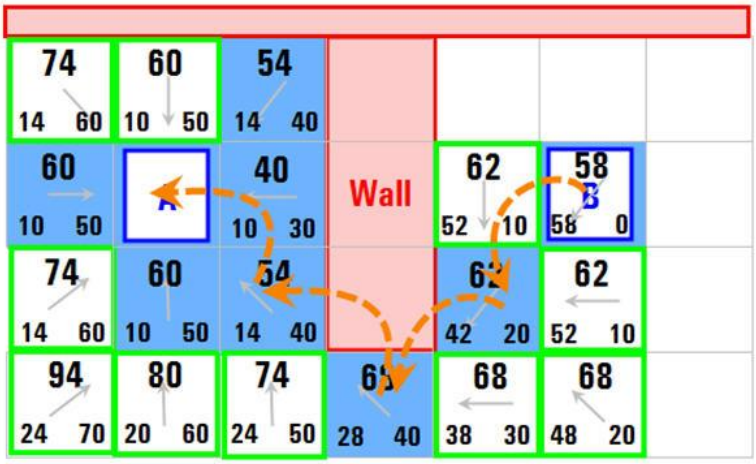


Figura 5-25. Definición de la ruta de resultante

5.4.1.1 Implementación de la ruta más corta

El módulo Robotics de LabVIEW incluye algoritmos de planeación de rutas en los cuales está incluido el algoritmo A* (ver Figura 5-26).

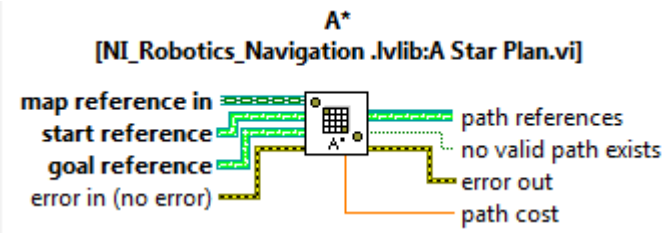


Figura 5-26. Utilidad A* incluida en LabVIEW

Como se mencionó en la explicación del funcionamiento del algoritmo A*, es necesario simplificar el área de búsqueda dividiéndola en una cuadricula. Para esto el espacio de trabajo se dividió en cuadros de 12cm por lado. La idea es que los nodos (puntos intermedios de los cuadros) que componen la ruta calculada por el algoritmo A* sean las diferentes posiciones que el punto de guiado del robot DaNI debe ir alcanzando conforme avance hacia la meta como se ilustra en la Figura 5-27.

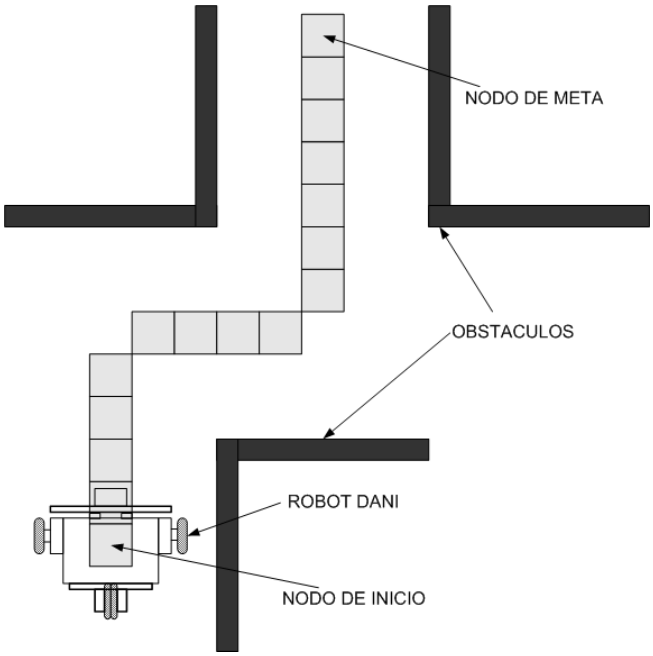


Figura 5-27. Los nodos (cuadros grises) forman la ruta a seguir por el robot

Se escogió este tamaño de cuadro debido a que permite representar las paredes que son los obstáculos sin perder significativamente el espacio libre cerca de estas paredes, además que el total de cuadros resulta en un número manejable al momento de hacer el mapa de todo el espacio de trabajo. Un ejemplo de esta representación se observa en la Figura 5-28.

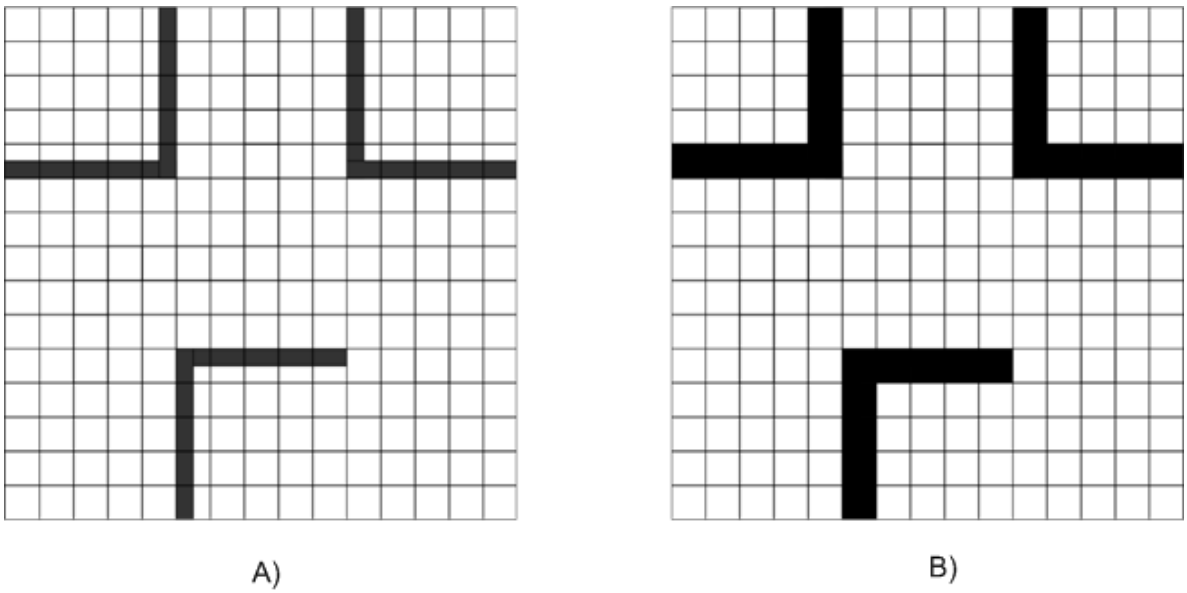


Figura 5-28. Representación de las paredes (obstáculos). En A) se observa el área de búsqueda dividida en cuadros, y en B) el área se representa por cuadros

La cuadrícula del área de búsqueda o espacio de trabajo se representa como un *mapa de rejilla de ocupación*, donde a cada uno de los cuadros que componen el mapa

del espacio de trabajo se le asigna un valor que indica la probabilidad de que el área que representa un cuadro este ocupada. De esta manera un cuadro con un valor de 100 indica que se encuentra ocupado por algún obstáculo y representa un área que no es transitable, y un valor de 0 indica un área que es transitable. La lógica fundamental de esto es que el robot solo puede a travesar un cuadro si este es transitable. De esta manera, en el ejemplo de la Figura 5-28-B) su rejilla de ocupación quedaría como se muestra en la Tabla 5-1.

Tabla 5-1. Mapa de rejilla de ocupación

0	0	0	0	100	0	0	0	0	0	100	0	0	0	0
0	0	0	0	100	0	0	0	0	0	100	0	0	0	0
0	0	0	0	100	0	0	0	0	0	100	0	0	0	0
0	0	0	0	100	0	0	0	0	0	100	0	0	0	0
100	100	100	100	100	0	0	0	0	0	100	100	100	100	100
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	100	100	100	100	100	0	0	0	0	0
0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
0	0	0	0	0	100	0	0	0	0	0	0	0	0	0
0	0	0	0	0	100	0	0	0	0	0	0	0	0	0

El espacio de trabajo en el que se usó el robot DaNI se muestra en la Figura 5-29.

Figura 5-29. Espacio de trabajo

La representación de este espacio de trabajo dividido en cuadros con la ubicación del sistema de referencia global se muestra en la Figura 5-30.

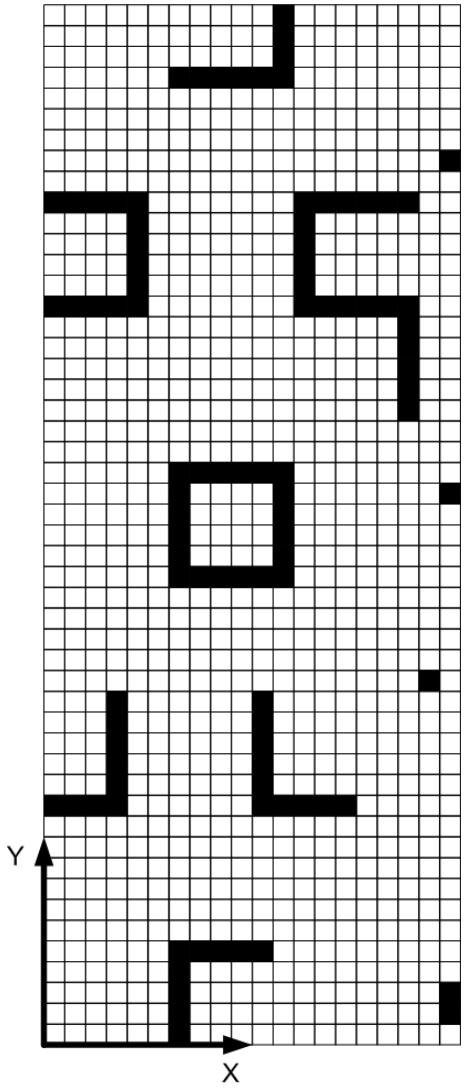


Figura 5-30. Representación del espacio de trabajo. El origen del sistema de referencia global se ubicó en la esquina inferior izquierda.

Para crear el mapa de rejillas de ocupación se elaboró en Excel la cuadricula del espacio de trabajo mostrada en la Tabla 5-2, posteriormente este archivo se guardó como un archivo de texto.

Tabla 5-2. Mapa de rejilla de ocupación hecho en Excel

0	0	0	0	0	0	0	0	0	2	2	100	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	100	1	1	1	1	100	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	2	100	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	100	1	1	1	1	100	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	2	100	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	100	1	1	1	1	100	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	2	2	2	100	100	100	100	100	100	2	2	0	2	2	2	2	2	2	2	2	2	2	0	0	2	2	2	100	1	1	1	1	100	2	0	0	0	0	0	0	0	0	0	0	0
2	2	2	2	2	2	2	0	2	2	2	2	2	2	2	2	2	2	2	0	2	2	2	2	2	2	2	2	2	2	0	0	2	2	2	100	100	100	100	100	100	2	0	2	2	0	2	2	2	2	2	2	0
2	2	2	2	2	2	2	0	2	2	2	2	2	2	2	2	2	2	2	0	2	2	2	2	2	2	2	2	2	2	0	0	2	2	2	2	2	2	2	2	2	2	0	2	2	0	2	2	2	2	2	2	0
100	100	100	100	100	2	2	0	2	0	0	0	0	0	0	0	0	0	0	0	2	2	100	100	100	100	100	100	2	2	0	0	2	2	2	2	2	2	2	2	2	2	0	2	2	0	2	2	2	100	2	2	0
2	2	2	2	100	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	100	1	1	1	1	100	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	2	2	2	100	2	2	0
2	2	2	2	100	2	2	0	0	2	2	2	2	2	2	2	2	2	2	0	2	2	100	1	1	1	1	100	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	2	2	2	100	2	2	0
0	0	2	2	100	2	2	0	0	2	2	2	2	2	2	2	2	2	2	0	2	2	100	1	1	1	1	100	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	2	2	2	100	2	2	2
0	0	2	2	100	2	2	0	0	2	2	100	100	100	100	100	100	2	2	0	2	2	100	1	1	1	1	100	2	2	0	0	2	2	2	2	2	2	2	2	2	2	0	2	2	0	2	2	2	100	2	2	2
0	0	2	2	2	2	2	0	0	2	2	100	2	2	2	2	2	2	2	0	2	2	100	100	100	100	100	100	2	2	0	0	2	2	2	2	2	2	2	2	2	2	0	2	2	0	2	2	2	100	100	100	100
0	0	2	2	2	2	2	0	0	2	2	100	2	2	2	2	2	2	2	0	2	2	2	2	2	2	2	2	2	2	0	0	2	2	2	100	100	100	100	100	100	2	0	2	2	0	2	2	2	2	2	2	2
0	0	0	0	0	0	0	0	0	2	2	100	2	2	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	0	0	2	2	2	100	2	2	2	2	100	2	0	2	2	0	2	2	2	2	2	2	2
0	0	0	0	0	0	0	0	0	2	2	100	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	100	2	2	2	2	100	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	100	2	2	2	2	100	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	2	2	2	2	2	0	2	2	2	2	2	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	100	2	2	2	2	100	2	0	0	0	0	0	0	0	0	0	0	0
2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	0	0	0	0	2	2	2	2	2	2	100	100	100	100	100	100	2	2	2	2	100	2	2	2	2	2	2	2	0	0	0	0	0
2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	2	2	100	2	2	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0
2	100	100	2	2	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	0	0	0	0	2	2	100	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	100	2	2	100	2	2	0	0	0	0	0

En la Tabla 5-2 se observa que los valores de los cuadros adyacentes de las paredes es de 2, esto es debido a que al calcular la ruta de menor coste LabVIEW traza la ruta con los cuadros adyacentes a las paredes, y debido a que es el punto de guiado del robot el que seguirá la ruta debe existir un espacio entre los cuadros que componen esta ruta y las paredes, este espacio corresponde a las dimensiones del robot. Esto se ilustra en la Figura 5-31.

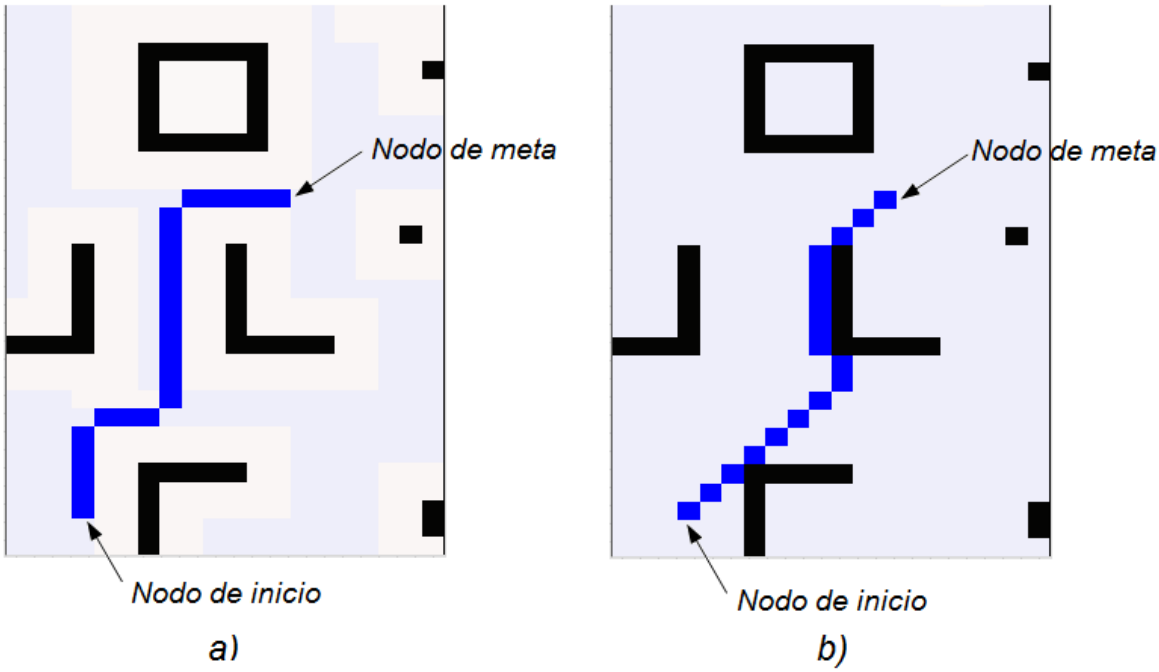


Figura 5-31. En a) los espacios blancos alrededor de las paredes corresponden a las dimensiones de la estructura del robot, en b) no están presentes.

Un valor de 2 en los cuadros adyacentes a las paredes es suficiente para que LabVIEW trace la ruta suficientemente despegada de las paredes y el robot pueda seguir la ruta sin chocar con las paredes.

Además, el mapa de rejillas de ocupación se debe crear de forma horizontal (rotado 90°) como se ve en la Tabla 5-2, esto se debe a la forma en que LabVIEW lee los caracteres del archivo de texto para crear el array de dos dimensiones.

El segmento de código mostrado en la Figura 5-32 convierte los valores del documento de texto en un array de dos dimensiones de manera que pueda ser leído por el VI *Create Occupancy Grid Map* para generar el mapa.

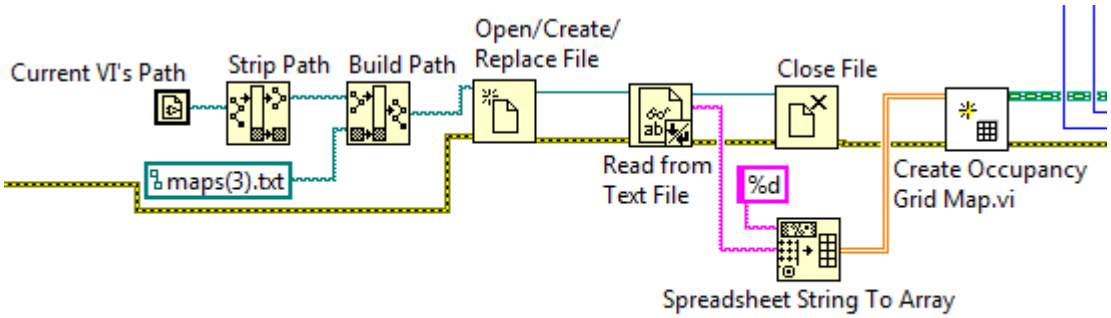


Figura 5-32. Segmento de código para leer el mapa guardado en el archivo de texto *maps(3)* en este caso

Para que el robot DaNI pueda leer el archivo de texto con la información del mapa, debe guardarse dicho archivo en la tarjeta sbRIO-9632. Para esto se debe conectar la computadora huésped al robot encendido, y en el explorador de Windows se debe poner la dirección ftp://<dirección IP>/ni-rt/startup/ en esa dirección se debe colocar el archivo de texto como se muestra en la Figura 5-33.

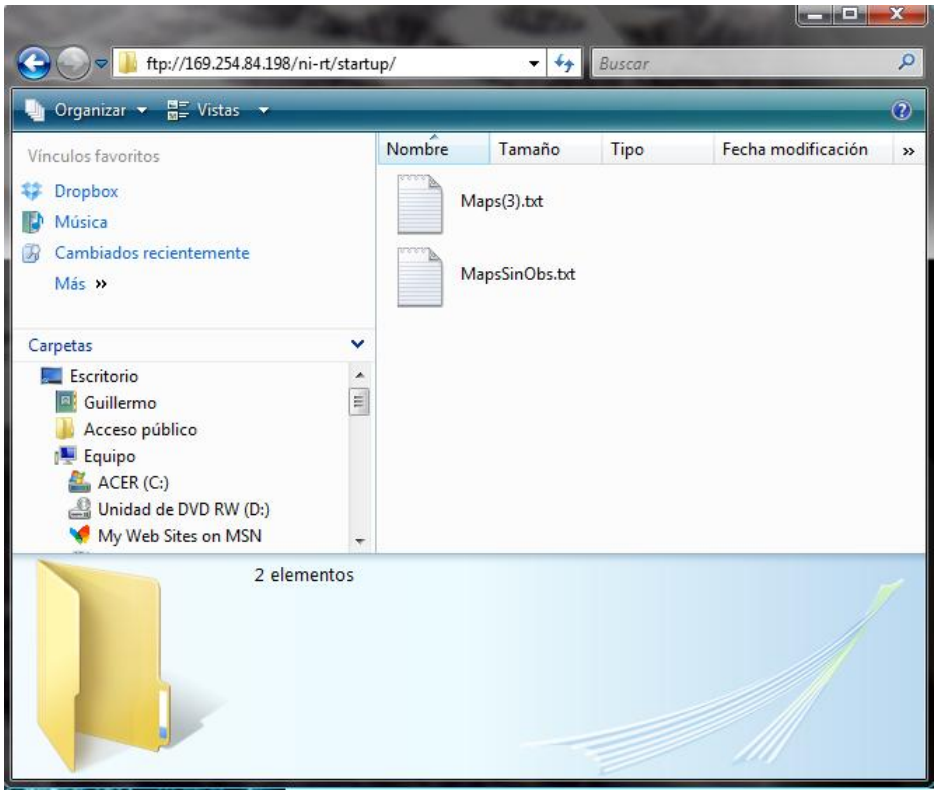


Figura 5-33. Ubicación en el robot DaNI de los mapas del espacio de trabajo

Posteriormente se indica las coordenadas de los cuadros en el mapa que corresponden a la posición de inicio y a la posición de meta con el VI *Get Cell Reference* como se ve en la Figura 5-34. De esta manera se procede a calcular la ruta de menor coste utilizando el VI *A Star Plan*.

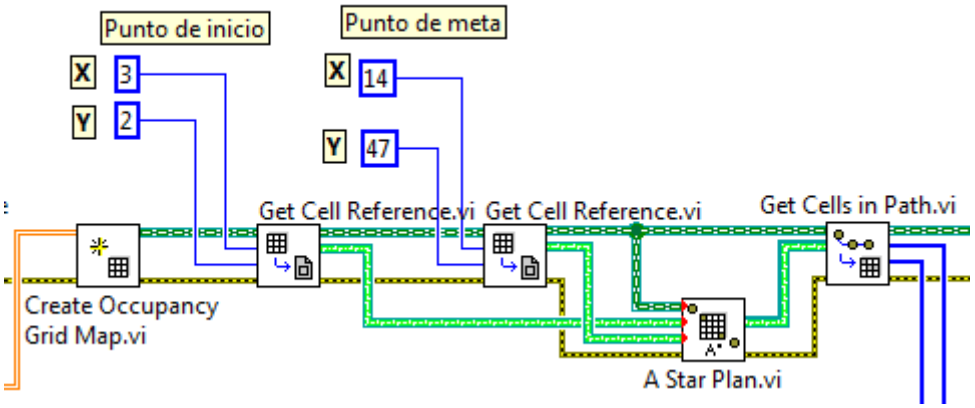


Figura 5-34. Segmento de código donde se indican las coordenadas de inicio y de meta

Utilizando el VI *Get Cell in Path* se obtienen las coordenadas de todos los cuadros que componen la ruta calculada por el algoritmo A*, finalmente lo que se tiene que hacer es transformar las coordenadas de los cuadros de la ruta a las posiciones en metros que deben ser alcanzadas por el robot en su camino a la meta, esta transformación para los 25 cuadros cercanos al origen del sistema global de referencia se muestra en la Figura 5-35.

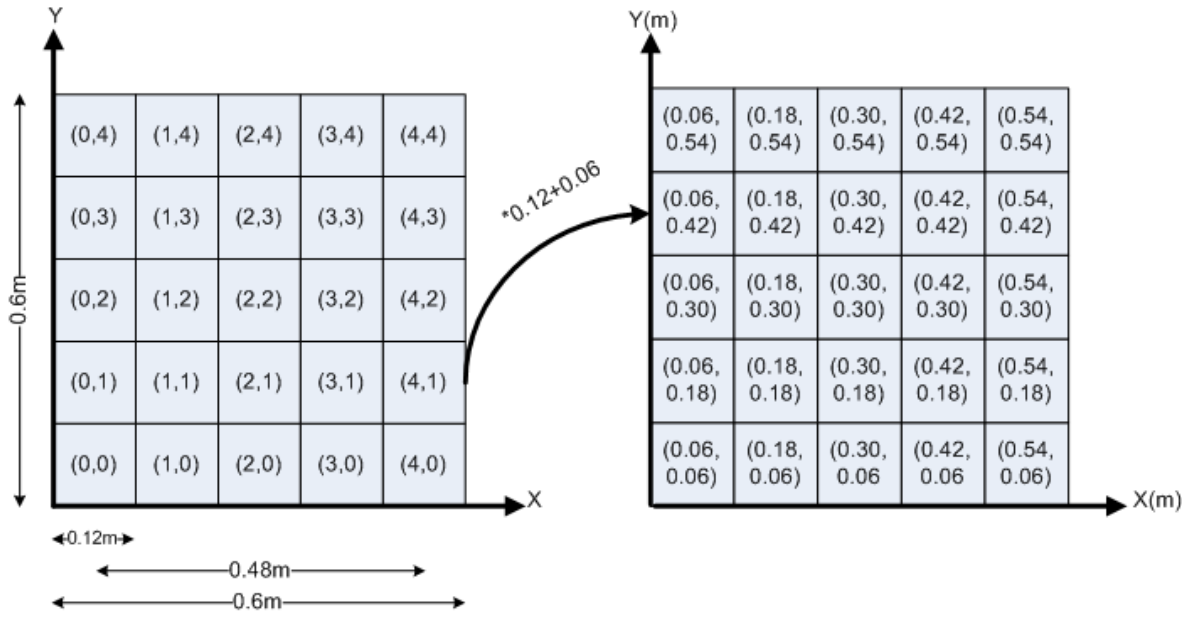


Figura 5-35. Transformación de coordenadas a posiciones en metros para los cuadros cercanos al origen del sistema de referencia

El código de la Figura 5-36 transforma las coordenadas de los cuadros del mapa en las posiciones de sus respectivos nodos en unidades de metros. La ruta a seguir es guardada como un arreglo de pares ordenados en la variable *Ruta a Seguir.*

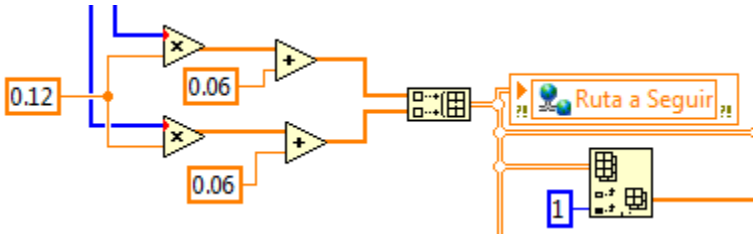


Figura 5-36. Segmento de código para transformar de coordenadas a posiciones en metros

De esta manera, si el robot parte de la posición inicial: (0.42,0.3)[m], con una orientación de 0°, y se dirige a la meta cuya posición es: (1.74,5.7)[m], el algoritmo A* traza la ruta mostrada en la Figura 5-37.

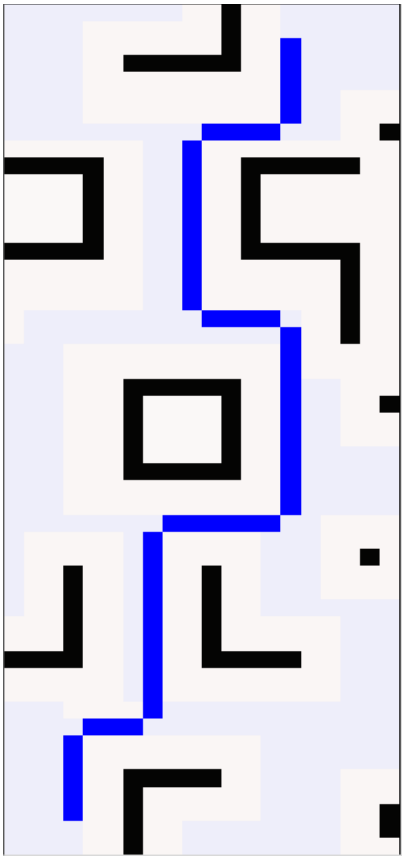


Figura 5-37. Ruta de menor coste trazada por el algoritmo A*

El programa completo para el cálculo de ruta más corta se muestra en la Figura 5-38.

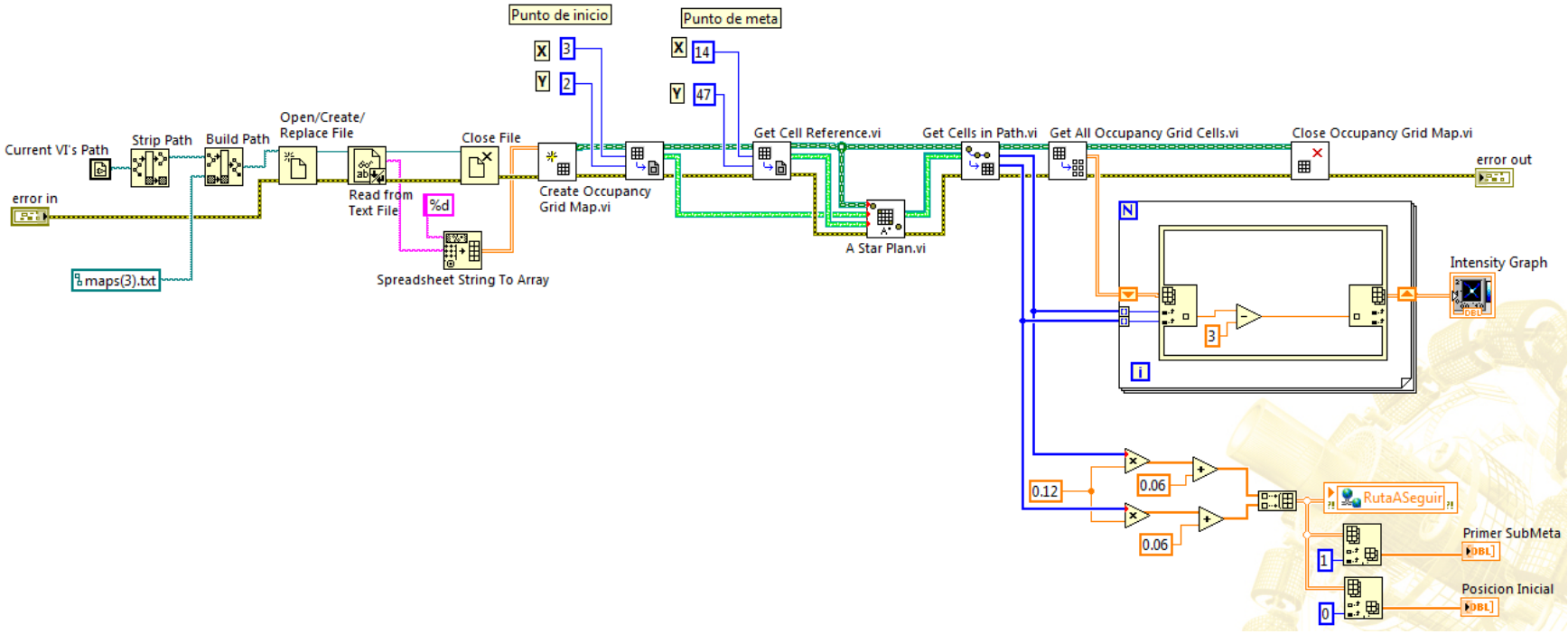


Figura 5-38. Programa para la planeación de la ruta de menor coste

El programa de la Figura 5-38 se nombró *PlanearRutaA_asterisco* cuya apariencia en el diagrama de bloques del VI principal se muestra en la Figura 5-39, este sub VI a diferencia de todos los demás procesos o subrutinas se ejecuta una sola vez antes de los demás procesos en el VI principal mostrado en la Figura 5-3, de manera que el robot cuando inicia su recorrido ya tiene planeada la ruta que debe seguir.

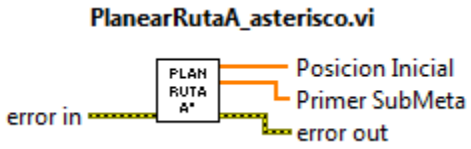


Figura 5-39. Apariencia de todo el programa encapsulado en un VI

5.4.2 Metas inmediatas

El hecho de que la ruta a seguir este compuesta de diferentes posiciones es aprovechado para dividir la ruta en diferentes *submetas* o *metas inmediatas***,** de manera que el robot DaNI debe ir alcanzando las posiciones de estas submetas para llegar a la posición final que corresponde a la *meta global*. Para esto se debe definir un *objetivo inmediato* el cual establece las coordenadas de la submeta actual o meta inmediata (a la cual el robot se debe dirigir) utilizando las posiciones de los diferentes nodos que componen la ruta calculada por el algoritmo A* la cual corresponde al “camino a seguir” (ver Figura 5-40).

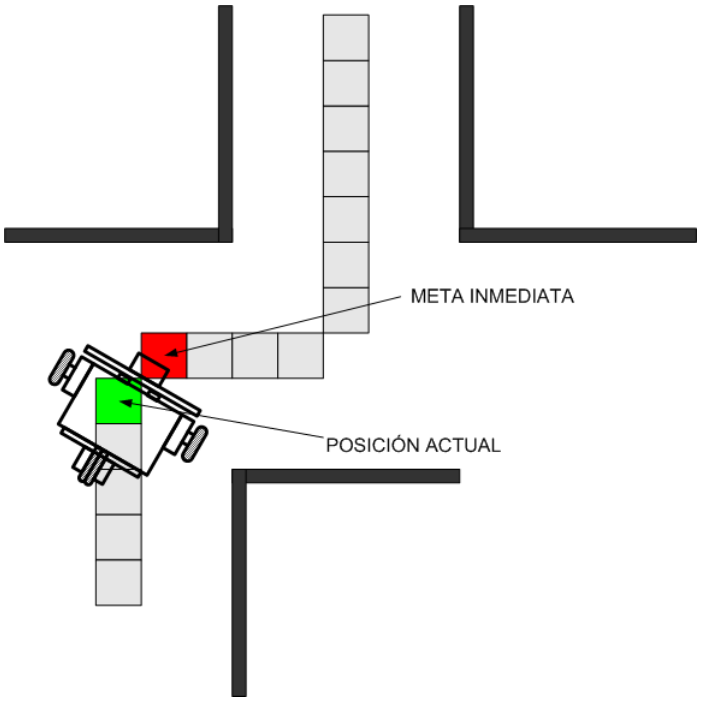


Figura 5-40. Los nodos que componen la ruta son vistos como metas inmediatas

Las coordenadas de la meta inmediata siguiente se obtienen cada vez que el robot DaNI ha logrado acercase lo suficiente a la posición de meta inmediata anterior. Para lograr esto se compara la posición actual del robot con la posición de meta inmediata con

un cierto valor de tolerancia, mientras no coincidan las posiciones el objetivo inmediato se considera no cumplido, en el momento en que las dos posiciones coinciden se considera cumplido el objetivo y se obtiene una nueva meta inmediata (ver Figura 5-41).

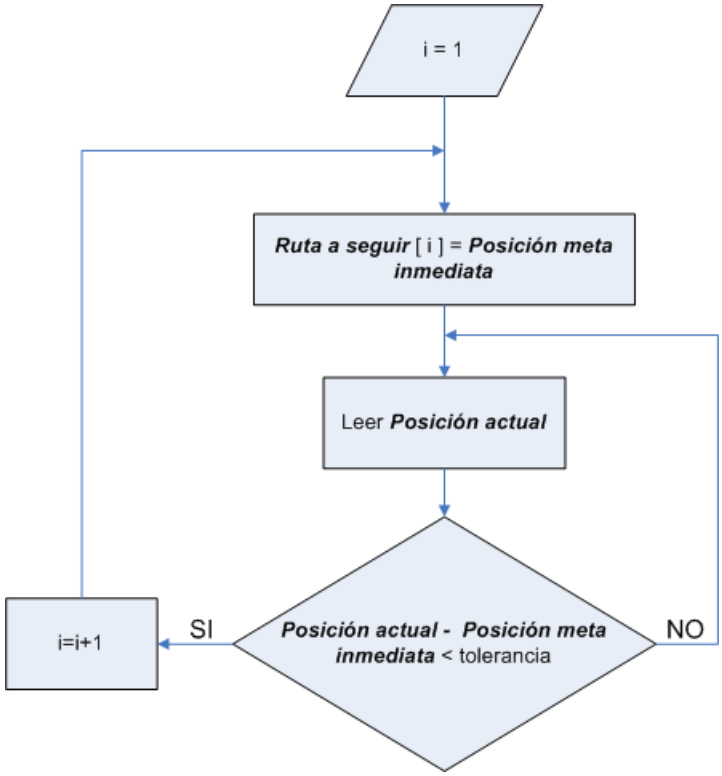


Figura 5-41. Algoritmo para ir estableciendo las posiciones de las metas inmediatas

Para la implementación del algoritmo de la Figura 5-41 se hicieron dos VIs. El VI de la Figura 5-42 se encarga de comparar la posición actual con la posición de meta inmediata, para decidir si la meta ha sido alcanzada.

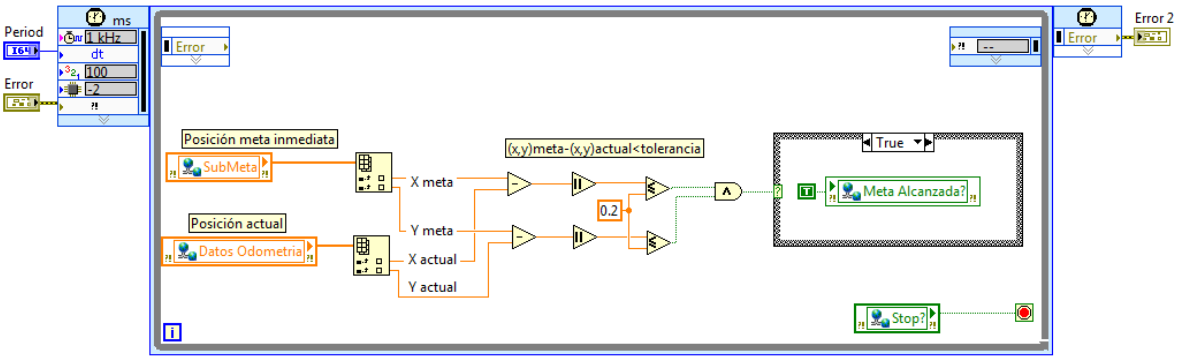


Figura 5-42. VI que compara la posición actual con la de la meta inmediata

El VI que establece si la meta inmediata fue alcanzada se nombró *MetaAlcanzada* y su apariencia en diagrama de bloques se observa en la Figura 5-43, este VI se ejecuta de manera paralela a los demás procesos del VI principal.

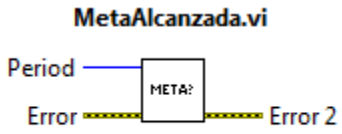


Figura 5-43. Apariencia como bloque del VI de la Figura 5-

El VI que se observa en la Figura 5-44 se encarga de establecer la posición de la siguiente meta inmediata, una vez que la meta inmediata anterior fue alcanzada.

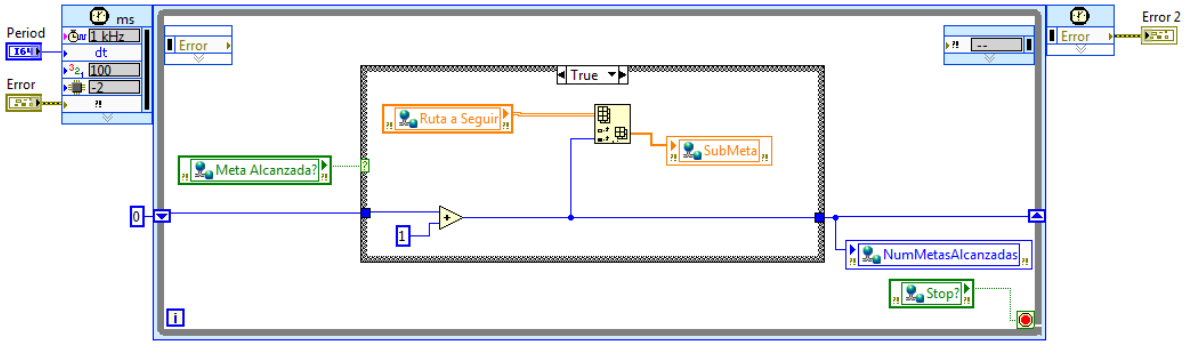


Figura 5-44. VI que entrega la posición de la meta inmediata siguiente

La variable *NumMetasAlcanzadas* que se observa en la Figura 5-44 se utiliza para definir el paro general (ver Figura 5-5) una vez que el robot DaNI ha recorrido todas las posiciones de la ruta a seguir.

Este VI se nombró *NuevaMeta* como se observa en la Figura 5-45, y de igual manera se ejecuta de forma paralela y asíncrona a los demás procesos del VI principal.

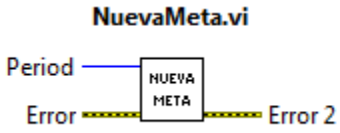


Figura 5-45. Apariencia como bloque del VI de la Figura 5-

5.4.3 Comandos de movimiento

Los comandos de movimiento se refieren a las instrucciones que son enviadas a los motores del robot para realizar el movimiento adecuado para recorrer la ruta que lleva a la meta final establecida. Se tienen tres comandos en la forma $[x' \quad y' \quad \emptyset']$ en el sistema de referencia local del robot que corresponden al movimiento hacia delante, hacia los lados y el de orientación respectivamente (ver Figura 5-46). Recordando que en el direccionamiento diferencial no existe desplazamiento de forma lateral debido a la

restricción holónoma (como se explica en el tema 2.1), el primer comando es igual a cero, el segundo comando corresponde a la velocidad lineal máxima del vehículo, por lo que el comando que en realidad se tiene que calcular es el de velocidad angular del robot.

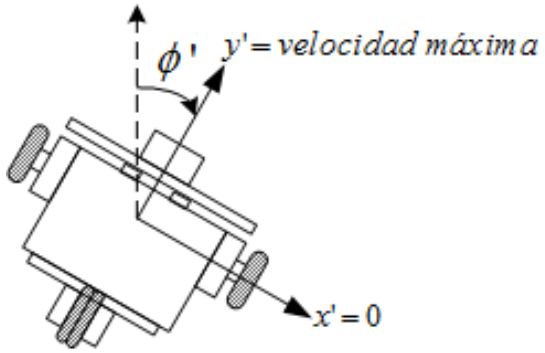


Figura 5-46. Comandos de movimiento en el robot DaNI. Los tres comandos son relativos al sistema de referencia local del robot

El proceso que genera este comando de orientación se basa en el algoritmo de *Histograma de Campo Vectorial* o *VFH* por sus siglas en inglés (*Vector Field Histogram*).

5.4.3.1 Histograma de campo vectorial (VFH)

Es un método que se implementa en robots móviles utilizado principalmente para la evasión de obstáculos en tiempo real, que permite evitar colisiones mientras simultáneamente dirige al robot hacia la meta u objetivo establecido. Este algoritmo utiliza un *histograma cartesiano* con la información de todo el espacio de trabajo el cual después es reducido a un *histograma polar* con la información local alrededor del robot.

Los algoritmos VFH disponibles en el módulo de robótica de LabVIEW (Figura 5-47) utilizan sólo el histograma polar.

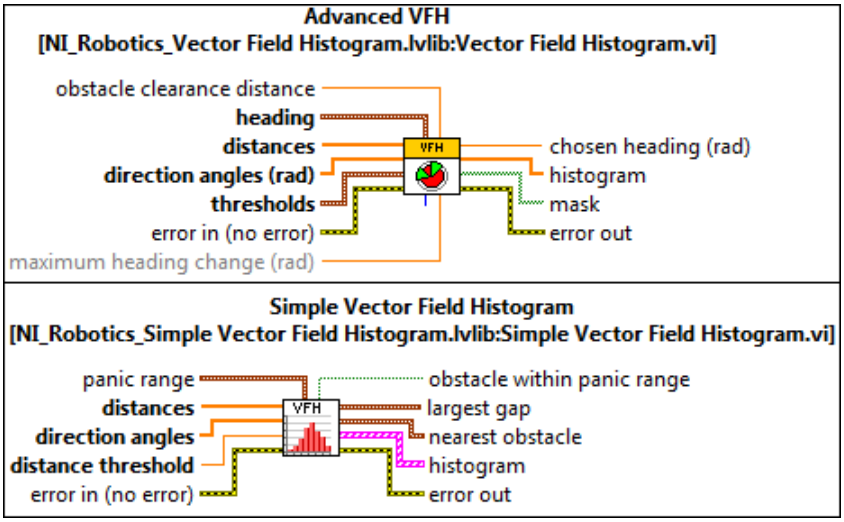


Figura 5-47. Algoritmos de histograma de campo vectorial en LabVIEW

El histograma se construye con ayuda del sensor ultrasónico del robot DaNI y contiene la información del área frente al robot identificada por el barrido del sensor ultrasónico. El histograma polar representa la *densidad polar de obstáculos* en diferentes direcciones, el algoritmo VFH se encarga de seleccionar la dirección con una menor *densidad polar de obstáculos*, en [3] se puede conocer a detalle las características del método VFH.

En la Figura 5-48-A) se muestra una ejemplo donde se pueden observar los valles (zonas con baja densidad polar de obstáculos) frente al robot, de los cuales se selecciona el mejor (en color verde). En la Figura 5-48-B) se puede ver la situación que genera el histograma del lado izquierdo, la dirección que mejor lleva a la posición de meta se obtiene del valle seleccionado.

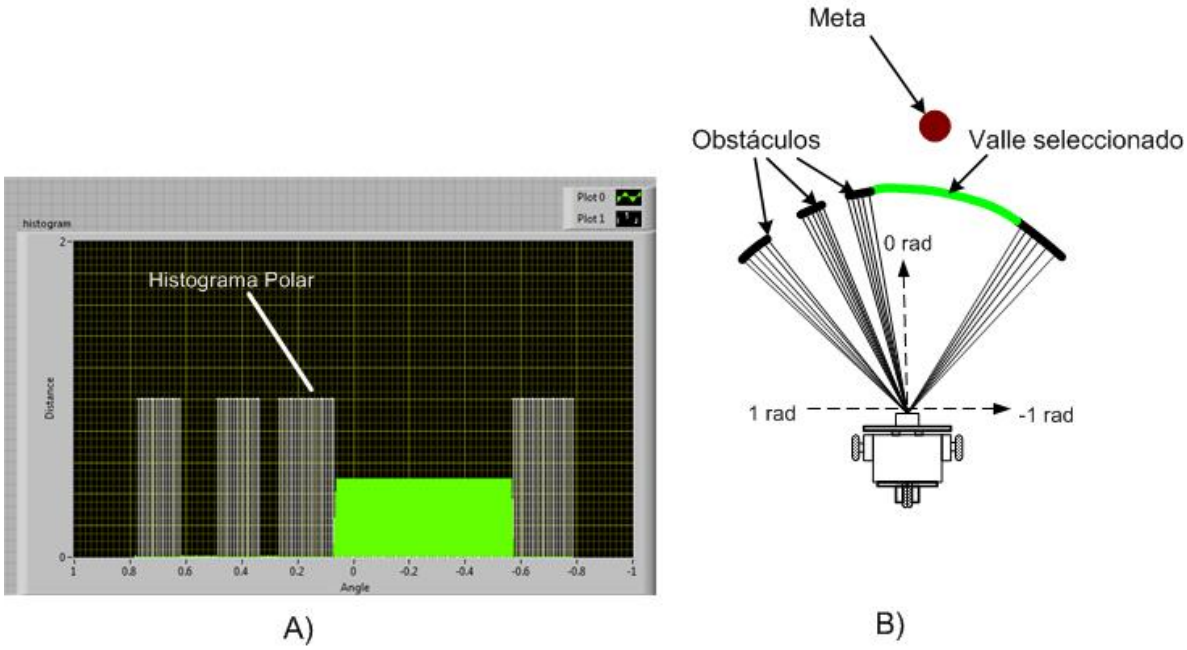


Figura 5-48. Representación de un histograma polar

Tomando esta dirección seleccionada como un comando de orientación se puede alinear el robot con la dirección instantánea que mejor lleva a la posición de meta inmediata como se muestra en la Figura 5-49.

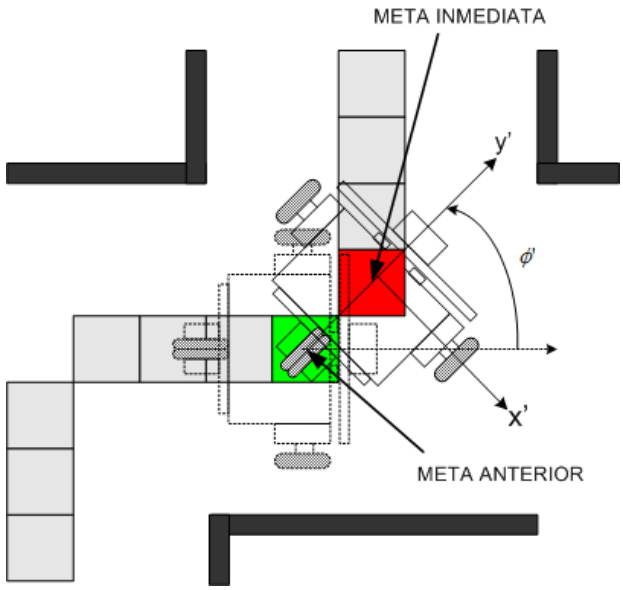


Figura 5-49. El comando ø' alinea al robot con la dirección instantánea que mejor lleva a la meta inmediata, mientras el comando y' hace avanzar al robot hacia adelante este proceso se repite una y otra vez mientras el robot va navegando

El proceso encargado de calcular los comandos de movimiento con el método VFH se integra en el VI *comandos de movimiento*, cuyo diagrama de bloques se muestra a en la Figura 5-50.

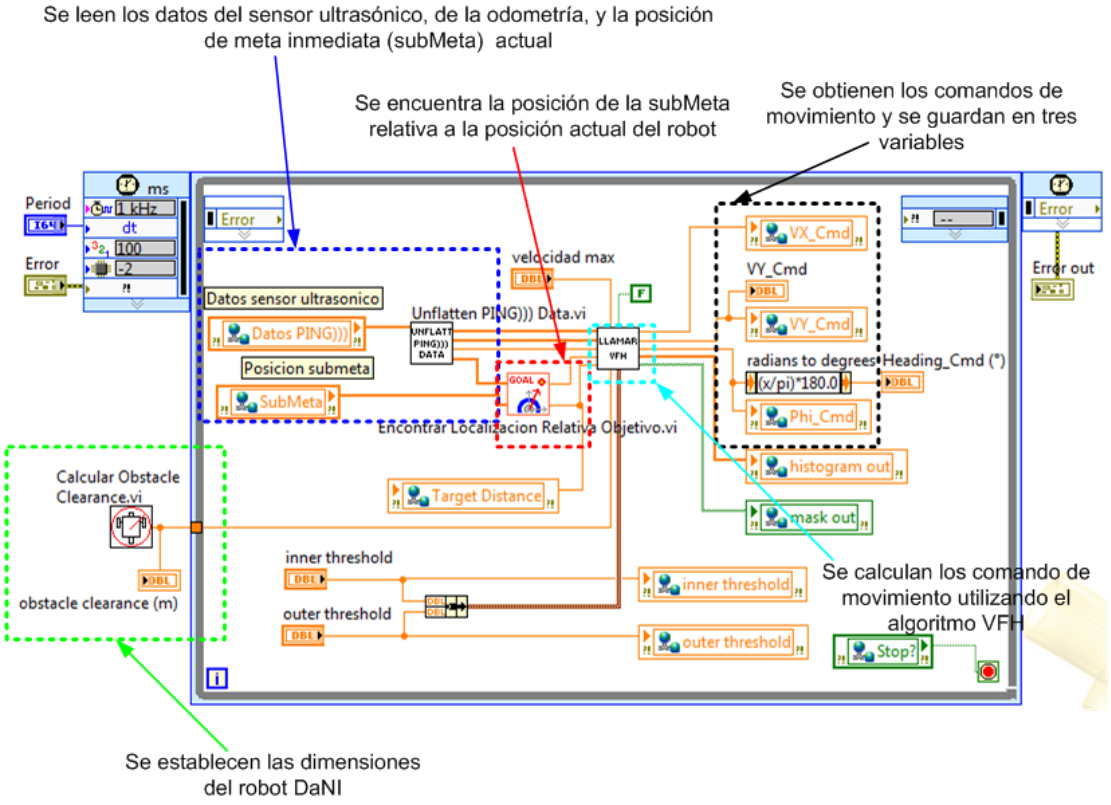


Figura 5-50. Código del VI *Comandos de Movimiento*

Dentro del VI *comandos de movimiento* se tiene otro subVI nombrado *Llamar VFH* (Figura 5-51), dentro de este subVI se calcula el comando de orientación utilizando la utilidad *Advanced VFH* de LabVIEW.

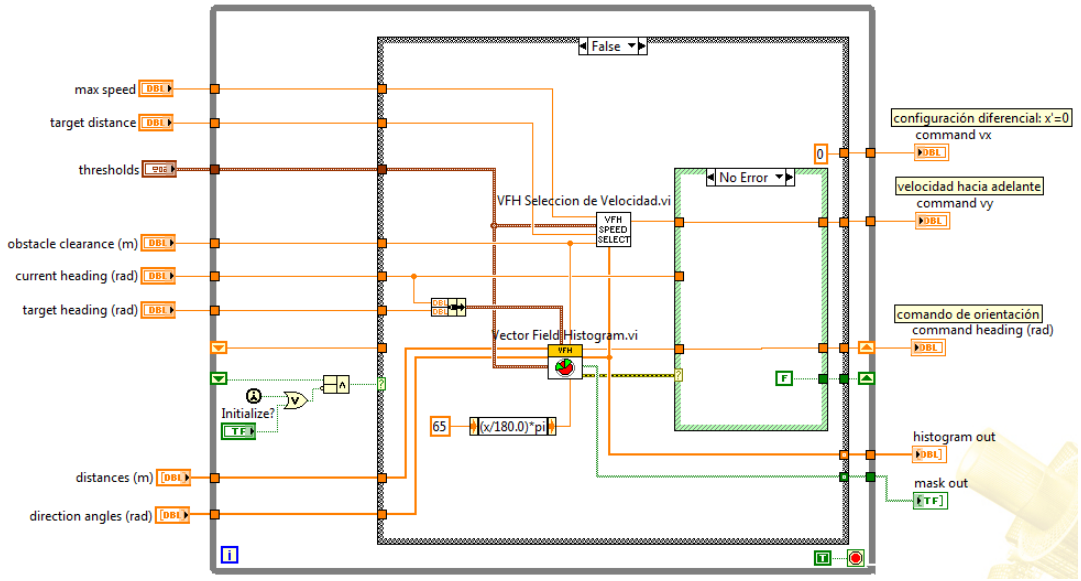


Figura 5-51. Código del VI *Llamar VFH*

El VI *Vector Field Histogram* calcula la orientación que mejor lleva a la posición de meta inmediata que se le indique a su vez que evade obstáculos. Esta función entrega un error si no existe alguna dirección libre para transitar, misma que se aprovechó para girar al robot DaNI 90° de manera que pudiera evitar obstáculos imprevistos, como se observa en la Figura 5-52.

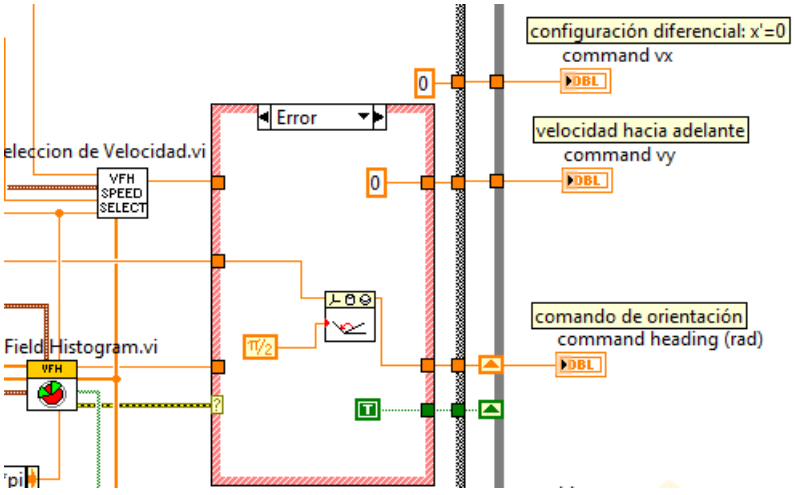


Figura 5-52. En caso de que no exista una dirección libre (error en el algoritmo VFH) el robot gira $\boldsymbol{\pi/2}$

En el subVI *Llamar VFH* se tiene otro subVI nombrado *VFH Selección de Velocidad* (Figura 5-53) el cual desacelera al robot si este se acerca a la posición de meta actual o a algún obstáculo inesperado.

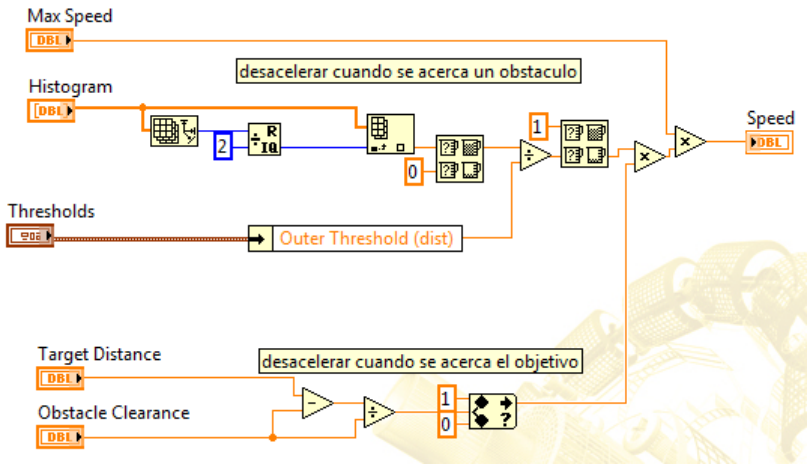


Figura 5-53. Para desacelerar al robot se utiliza la información del histograma (obstáculos) y la distancia hasta el objetivo actual

El VI *comandos de movimiento* guarda los comandos en las variables *VX_Cmd, VY_Cmd,* y *Phi_Cmd* (ver Figura 5-50). El proceso para el cálculo de los comandos de movimiento (Figura 5-54) también se ejecuta de forma paralela y asíncrona a los demás procesos.

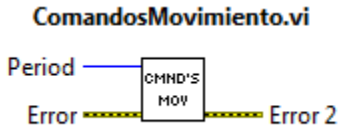


Figura 5-54. Apariencia del bloque del VI para el calculo de los comandos de movimiento.

5.5 Reacción

Finalmente una vez que se tienen todas las instrucciones para guiar al robot es necesario convertirlas de manera que se puedan accionar los motores para lograr el movimiento deseado.

5.5.1 Accionamiento de los motores

Para accionar los motores se creó el VI mostrado en la Figura 5-55, donde los comandos de movimiento están guardados en la variable *velocidad steering frame* (ver Figura 5-58) esta variable indica la velocidad lineal y angular que debe tener el robot.

Al escribir el valor del comando de orientación en el VI *Apply Velocity to Motor,* este comando es convertido a un comando de velocidad angular, por ejemplo si el algoritmo VFH entrega una dirección de $\pi/6\ [rad]$, este valor de orientación se convierte al comando de velocidad angular $\pi/6\ [rad/s]$, lo cual indica que el robot debe girar 30° en un segundo.

Utilizando el VI *Apply Velocity to Motor* se transforman las instrucciones de velocidad lineal y angular del robot a instrucciones de velocidad angular de las ruedas. Finalmente se hace girar los motores del robot a esos valores de velocidades angulares

escribiendo sus valores en el VI *Write DC Motor Velocity Setpoint,* con lo que se logra mover al robot DaNI con la velocidad lineal y angular indicadas por los comandos.

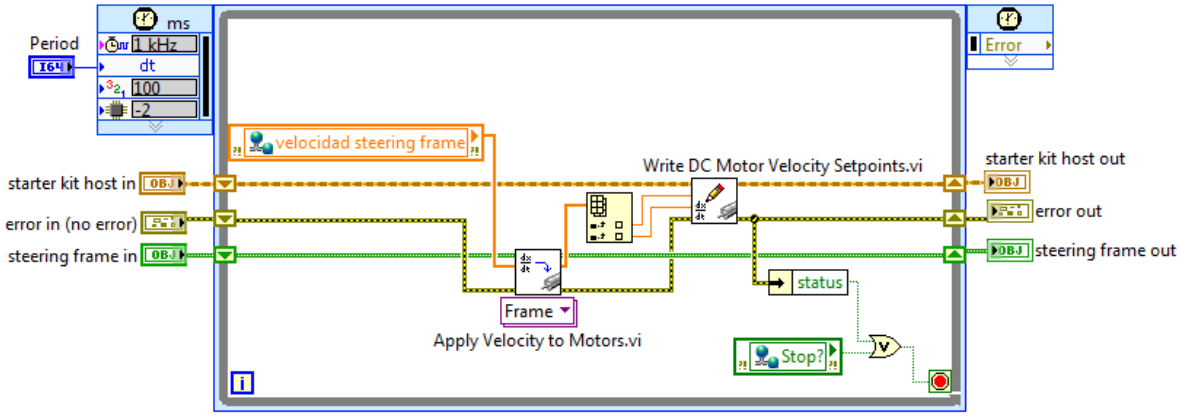


Figura 5-55. Código del VI creado para el accionamiento de los motores del robot DaNI

5.5.2 Control de la orientación

Al hacer las pruebas ubicando una meta en línea recta justo hacia en frente del robot se observó que a medida que se acercaba a la meta el robot, este empezaba a oscilar en su movimiento. Esto mismo ocurría si se colocaba un obstáculo en su trayectoria, después de evadir el obstáculo las oscilaciones crecían a medida que se acercaba a la meta el robot, adicionalmente en este caso las oscilaciones terminaban por desestabilizar el movimiento del robot haciéndolo girar rápidamente sin control. Este mismo fenómeno se presentó con mayor énfasis al ubicar una meta en una posición que no estuviera en línea recta frente al robot, en estos casos el robot inmediatamente empezaba a girar en su sitio rápidamente sin control en la dirección que se encontraba la meta. Por ejemplo si la meta estaba a la derecha del robot este empezaba a girar sin control en el sentido de las manecillas del reloj

Al analizar esta situación se llegó a la conclusión de que el problema tenía que ver con la orientación instantánea del robot. Para solucionar esto fue necesario implementar en el robot un control de orientación. El controlador que se implementó fue un control PID, debido a que es relativamente sencillo de implementar, además de que el método utilizado en este trabajo para sintonizarlo se basa en pruebas experimentales.

El diagrama de bloques del controlador PID implementado se muestra en la Figura 5-56, este es un esquema modificado del PID básico denominado PI-D como se menciona en [4], en este esquema la acción derivativa opera sólo en la trayectoria de realimentación actuando solo sobre la variable del proceso y no en la señal de referencia.

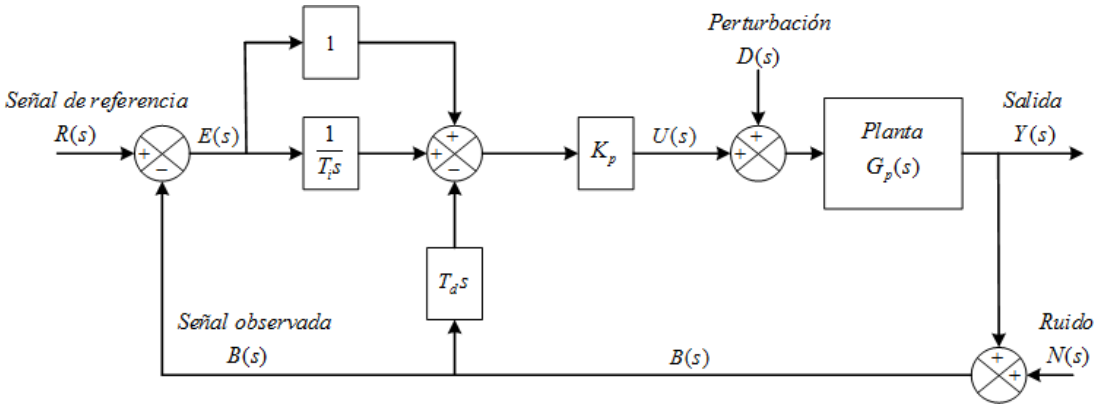


Figura 5-56. Diagrama de bloques de un sistema controlado por un controlador PI-D

El VI con el código del controlador PI-D programado en LabVIEW se muestra en la Figura 5-57.

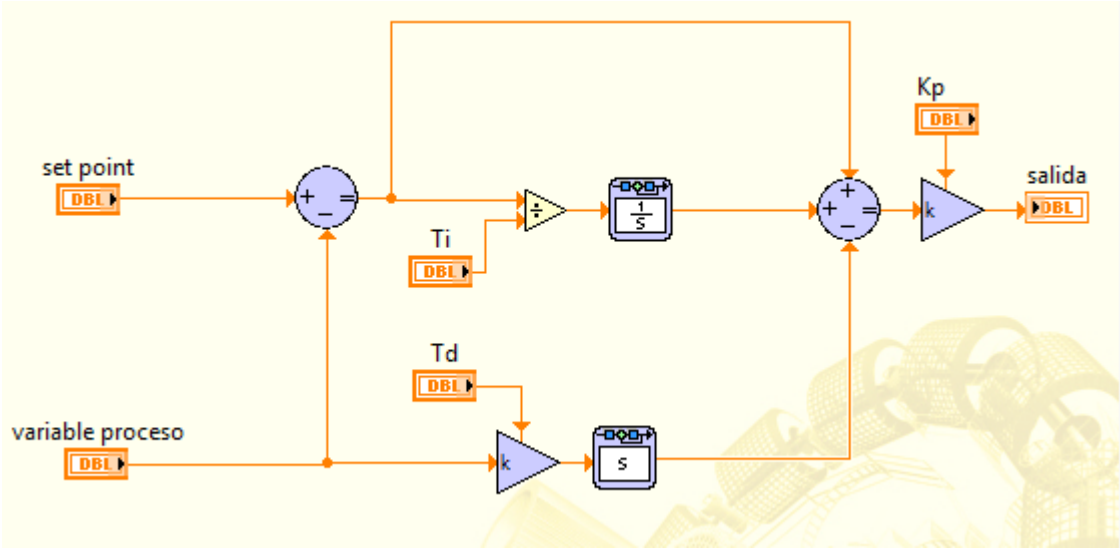


Figura 5-57. Diagrama de bloques del controlador PI-D en LabVIEW

El código del controlador PI-D está incorporado dentro de otro VI desde donde es llamado para hacer los cálculos cada que se tienen los datos de las variables necesarias. Este VI se muestra en la Figura 5-58.

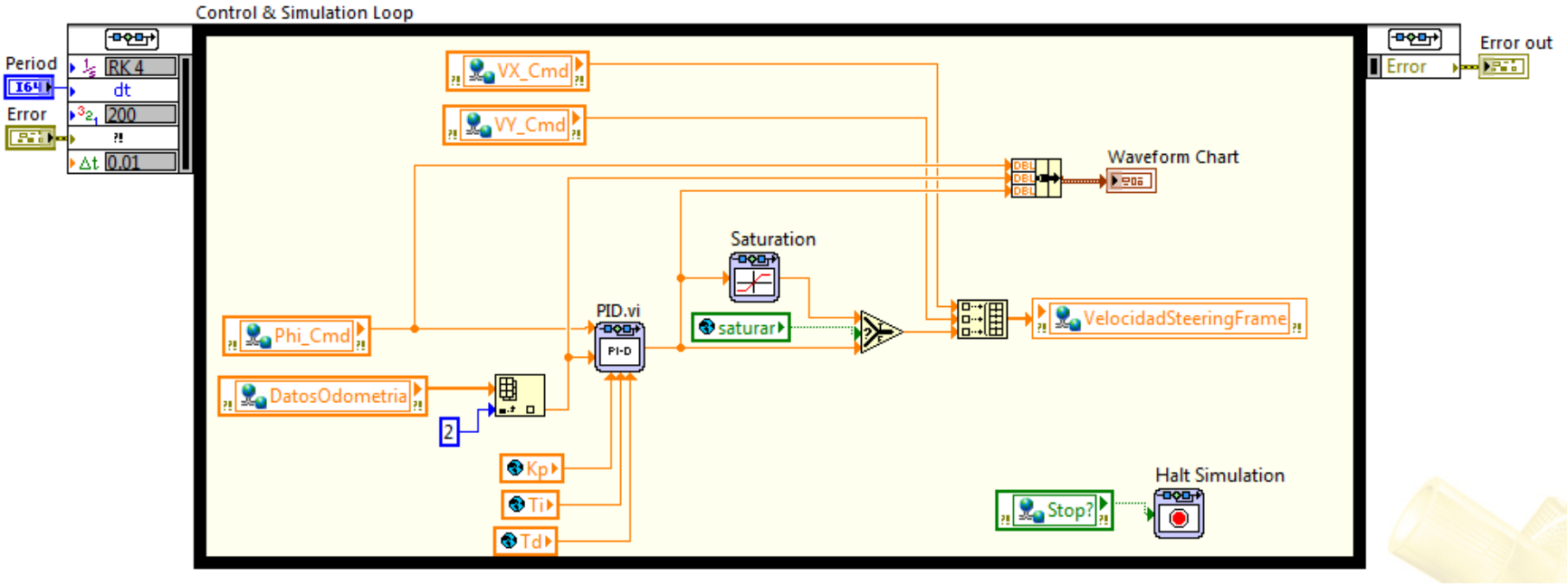


Figura 5-58. Diagrama de bloques del programa completo para el control de la orientación del robot DaNI

En el programa de la Figura 5-58, la variable a controlar es la orientación que se lee de la estimación odométrica (variable *Datos Odometría*). La señal de control se junta con las variables *VX_Cmd* y *VY_Cmd* que son los otros comandos de movimiento y se envían al proceso de accionamiento de los motores utilizando la variable *VelocidadSteeringFrame.*

De esta forma, con el programa de la Figura 5-58 el robot logra controlar su orientación y puede alinearse lo más cercano posible con cualquier dirección de meta inmediata. El proceso experimental para sintonizar el controlador PID se describe en el siguiente capítulo.

El programa creado se nombró *ControlOrientación*, este VI es llamado como subrutina desde el VI principal al igual que lo demás procesos paralelos. Su apariencia como bloque se muestra en la Figura 5-59.

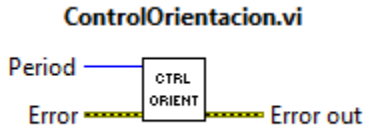


Figura 5-59. Apariencia como bloque del VI de control de orientación

Capítulo 6

Pruebas experimentales y resultados

En este capítulo se describen las pruebas realizadas en el robot DaNI, para ajustar parámetros de manera que el objetivo se cumpla cabalmente.

6.1 Ajuste del periodo del ciclo del modelo odométrico

Como se mencionó en el tema 2.5, el modelo odométrico proporciona una estimación de la posición y de la orientación del robot por lo que fue necesario determinar el periodo adecuado de ejecución del proceso de estimación odométrica, de manera que los valores arrojados por el modelo odométrico fueran lo más cercanos posibles a los reales.

Para esto se realizó la prueba ilustrada en la Figura 6-1 donde el robot DaNI a partir de la posición 1 en el origen del sistema de referencia y con una orientación de -45° avanzó hacia enfrente hasta detenerse en el instante en que alguna de sus coordenadas en x o en y alcanzara el valor de 1.2 (m). En esa posición 2 (x_f, y_f) se comparó la posición y orientación finales estimadas, con la posición y orientación finales reales, las cuales se midieron físicamente. Las pruebas se hicieron a una velocidad lineal del robot de 0.2 [m/s].

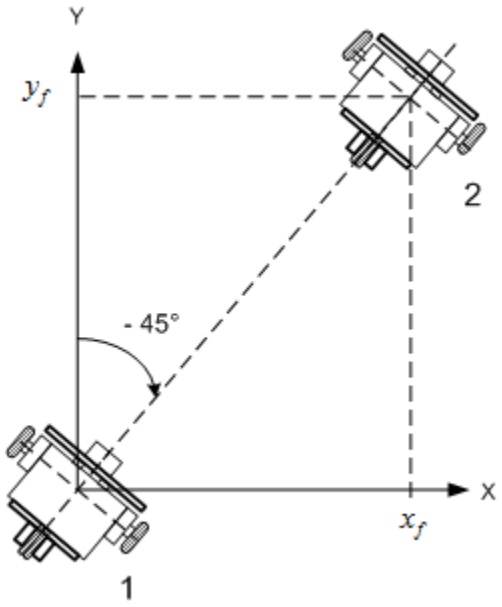


Figura 6-1. Prueba para determinar el periodo de ejecución del ciclo odométrico

Los resultados se muestran en Tabla 6-1.

Tabla 6-1. Resultados prueba periodo de ejecución ciclo odométrico

	Odometría		Mediciones		Error relativo (%)		
Periodo del ciclo odométrico (ms)	X (m)	Y (m)	X (m)	Y (m)	Ex (%)	Ey (%)	Promedio errores (%)
15	1.224	1.336	1.953	2.085	37.33	5.50	26.25
14	1.234	1.306	1.840	1.880	32.93	3.33	22.26
13	1.234	1.323	1.792	1.678	31.14	5.51	19.27
12	1.249	1.245	1.667	1.553	25.07	3.21	16.04
11	1.246	1.341	1.499	1.510	16.88	5.14	11.07
10	1.248	1.271	1.308	1.405	4.59	6.77	6.96
9	1.253	1.366	1.245	1.285	0.64	0.67	2.54
8	1.266	1.311	1.323	1.255	4.31	6.97	5.25
7	1.279	1.391	0.958	1.035	33.51	2.12	23.34
6	1.298	1.433	0.910	0.917	42.09	4.15	34.17
5	1.817	1.725	0.875	0.942	107.66	19.05	69.94

En la Tabla 6-1 se puede observar que para los valores de 9 (ms) y 8 (ms) se obtiene el menor valor de error relativo, al obtenerse errores sumamente parecidos se realizaron mediciones adicionales para estos dos valores, los resultados se observan en la Tabla 6-2.

Tabla 6-2. Resultados segunda prueba periodo de ejecución ciclo odométrico

	Odometría			Mediciones		Error relativo		
Periodo del ciclo odométrico (ms)	X (m)	Y (m)	Φ (°)	X (m)	Y (m)	Ex (%)	Ey (%)	Promedio del error (%)
9	1.253	1.318	-42.10	1.132	1.285	10.69	2.57	4.42
9	1.252	1.347	-42.92	1.251	1.265	0.08	6.48	2.19
9	1.245	1.378	-42.40	1.190	1.112	4.62	23.92	9.51
9	1.246	1.341	-43.10	1.244	1.250	0.16	7.28	2.48
9	1.253	1.366	-44.39	1.245	1.285	0.64	6.30	2.32
9	1.252	1.313	-43.44	1.181	1.225	6.01	7.18	4.40
9	1.250	1.295	-42.92	1.145	1.255	9.17	3.19	4.12
9	1.248	1.417	-40.85	1.245	1.278	0.24	10.88	3.71
9	1.249	1.481	-38.42	1.300	1.347	3.92	9.95	4.62
9	1.249	1.301	-40.84	1.153	1.262	8.33	3.09	3.81
8	1.266	1.311	-43.27	1.323	1.255	4.31	4.46	2.92
8	1.155	1.240	-44.13	1.155	1.240	0.00	0.00	0.00
8	1.259	1.316	-43.53	1.114	1.155	13.02	13.94	8.99
8	1.264	1.359	-42.92	1.110	1.200	13.87	13.25	9.04
8	1.270	1.273	-44.57	1.080	1.135	17.59	12.16	9.92

8	1.256	1.363	-43.53	1.210	1.191	3.80	14.44	6.08
8	1.254	1.374	-40.33	1.180	1.145	6.27	20.00	8.76
8	1.260	1.335	-41.02	1.117	1.155	12.80	15.58	9.46
8	1.263	1.441	-39.20	1.147	1.245	10.11	15.74	8.62
8	1.256	1.377	-43.44	1.160	1.135	10.28	21.32	9.87

Promediando los errores de los resultados mostrados en la Tabla 6-2 se concluye que un periodo de 9 (ms) en el ciclo de la estimación odométrica da mejores resultados.

6.1.1 Efecto de la velocidad lineal y angular del robot en el modelo odométrico

Para observar el efecto de la velocidad lineal del robot en el modelo odométrico se repitió la prueba ilustrada en la Figura 6-1, pero en esta ocasión el parámetro a variar fue la velocidad lineal. Según las especificaciones técnicas del robot, la velocidad máxima de los motores es $15.7\ [rad/s]$, lo cual se traduce en una velocidad lineal máxima del robot de $0.79[m/s]$. Los resultados se muestran en la Tabla 6-3.

Tabla 6-3. Resultados del efecto de la velocidad lineal en el modelo odométrico

	Odometría		**Mediciones**		**Errores**		
Velocidad (m/s)	X (m)	Y (m)	X (m)	Y (m)	Error relativo X (%)	Error relativo Y (%)	Promedio errores (%)
0.05	1.209	1.477	1.275	1.300	5.18	13.62	6.26
0.10	1.227	1.253	1.171	1.210	4.78	3.55	2.78
0.15	1.238	1.376	1.258	1.260	1.59	9.21	3.60
0.20	1.243	1.414	1.270	1.305	2.13	8.35	3.49
0.25	1.260	1.388	1.265	1.265	0.40	9.72	3.37
0.30	1.267	1.400	1.260	1.302	0.56	7.53	2.69
0.35	1.238	1.333	1.225	1.260	4.73	5.79	3.51
0.40	1.279	1.364	1.264	1.267	1.19	7.66	2.95
0.45	1.301	1.340	1.230	1.234	5.77	8.59	4.79
0.50	1.303	1.498	1.315	1.305	0.91	14.79	5.23
0.55	1.308	1.447	1.290	1.290	1.40	12.17	4.52
0.60	1.317	1.440	1.305	1.240	0.92	16.13	5.68
0.65	1.358	1.391	1.240	1.278	9.52	8.84	6.12
0.70	1.361	1.580	1.343	1.380	1.34	14.49	5.28
0.75	1.391	1.566	1.370	1.365	1.53	14.73	5.42
0.80	1.344	1.671	1.354	1.430	0.74	16.85	5.86

De los resultados obtenidos en la Tabla 6-3 se observa que a medida que se aumenta la velocidad lineal del robot aumenta el error en el modelo odométrico, en el rango de 0.10 a 0.4 (m/s) se obtuvieron los menores errores, derivado de esto se selecciono la velocidad de 0.2 (m/s) como la velocidad lineal máxima a la que se movería el robot. Ya que para fines del proyecto se requieren estimaciones y movimientos lo más precisos posible.

Para observar el efecto ahora de la velocidad angular (cuando el robot gira) en el modelo odométrico se diseñó la rutina ilustrada en la Figura 6-2. El robot partió de la posición 1 en el origen de coordenadas con una orientación de 0°, avanzó a la posición 2 a una distancia de 1.5 (m), realizó un giro en sentido de las manecillas del reloj y volvió a avanzar ahora una distancia de 1.2 (m) a la posición 3.

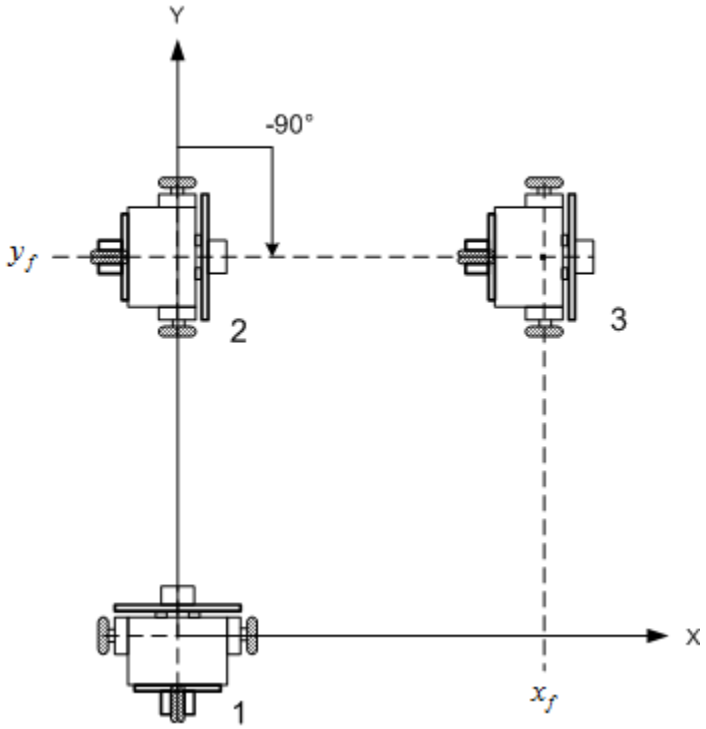


Figura 6-2. Prueba para observar el efecto de la velocidad angular en el modelo odométrico

Los resultados de esta prueba se muestran en la Tabla 6-4.

Tabla 6-4. Resultados del efecto de la velocidad angular en el modelo odométrico

	Odometría			**Mediciones**			**Error relativo**			
Velocidad angular (rad/s)	X (m)	Y (m)	Φ (°)	X (m)	Y (m)	Φ (°)	X (%)	Y (%)	Phi (%)	Promedio errores
0.2618	1.287	2.199	-96.14	1.205	2.250	-84.120	6.80	2.27	14.29	3.02
0.5236	1.279	1.708	-103.84	1.205	1.820	-92.799	6.14	6.15	11.90	4.10
0.7854	1.276	1.383	-113.28	1.338	1.385	-106.237	4.63	0.14	6.63	1.59
1.0472	1.275	1.270	-111.90	1.230	1.486	-96.730	3.66	14.54	15.68	6.06
1.309	1.260	0.764	-130.59	1.362	1.084	-116.437	7.49	29.52	12.15	12.34
1.5708	1.249	0.213	-142.96	1.600	0.692	-124.530	21.94	69.22	14.80	30.39
1.8326	1.241	-0.393	-151.53	1.765	0.115	-133.167	29.69	441.74	13.79	157.14

De los resultados obtenidos en esta prueba se observa que para velocidades angulares mayores a 1.0472 [rad/s] o de 45 [°/s] el error del modelo odométrico aumenta considerablemente, esto sugiere que los cambios de orientación no deben ser mayores a 45°. Esto es correcto ya que en la planeación de trayectorias al tener que realizar un giro de 90° se realiza primero uno de 45° y luego otro del mismo valor. Esto se ilustra en la Figura 6-3.

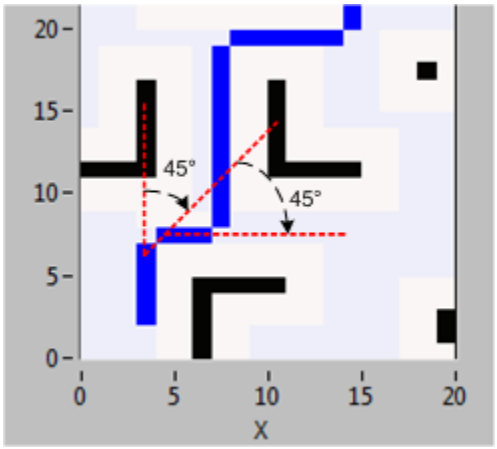


Figura 6-3. Forma en que se dividen los giros de ángulos rectos

6.2 Sintonización del control PID de la orientación

Para sintonizar el controlador PID se utilizó como punto de partida el segundo método de Ziegler-Nichols, el cual propone reglas para determinar los valores de la ganancia proporcional K*p*, del tiempo integral Ti y del tiempo derivativo Td.

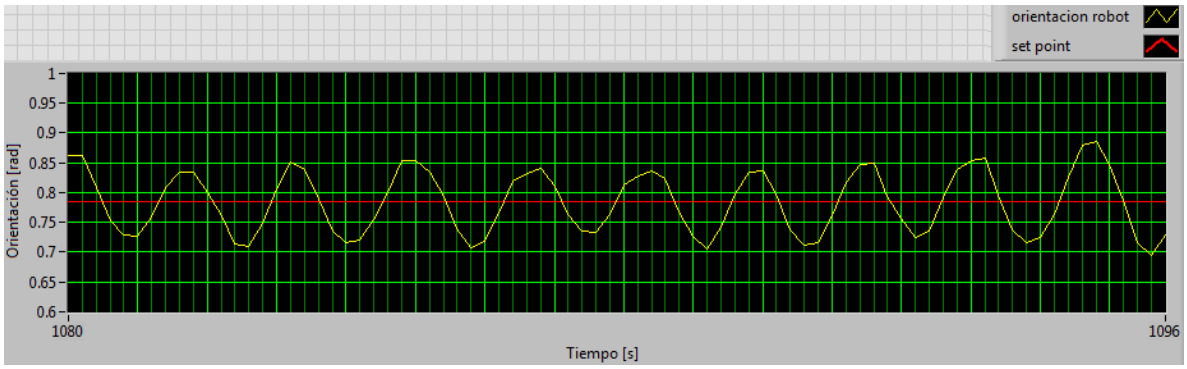
Inicialmente se usa sólo la acción proporcional del control fijando $T_i = \infty$ y $T_d = 0$, y se incrementa K_p desde 0 hasta un valor crítico K_{cr}, para el cual la salida presenta oscilaciones sostenidas. De esta manera se obtiene experimentalmente K_{cr} y el periodo P_{cr} de las oscilaciones sostenidas. Este método sugiere establecer los valores de los parámetros K_p, T_i, T_d según se muestra en la Tabla 6-5.

Tabla 6-5. Reglas de sintonía segundo método de Ziegler-Nichols

Tipo de controlador	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Para encontrar el valor de K_{cr} que provoca oscilaciones sostenidas en la orientación del robot se aplicó una entrada escalón que corresponde a cambio de orientación de $\pi/4\,[rad]$, lo que indica que a partir de una orientación de 0°, el robot debe girar hasta 45° en sentido contrario a las manecillas del reloj, esto se hace mientras el robot sigue su curso hacia adelante. Esta prueba se realizó en movimiento, debido a que los cambios de orientación del robot se hacen al mismo tiempo que este va avanzando hacia la meta. Lo que significa que la acción de velocidad lineal y angular están acopladas.

De esta manera, el valor para el cual el robot mostró oscilaciones sostenidas en su orientación fue de $K_{cr} = 3.5$ y el periodo de dichas oscilaciones fue de $P_{cr} = 1.7[s]$. En la Gráfica 6-1 se puede ver la gráfica obtenida de la oscilación.



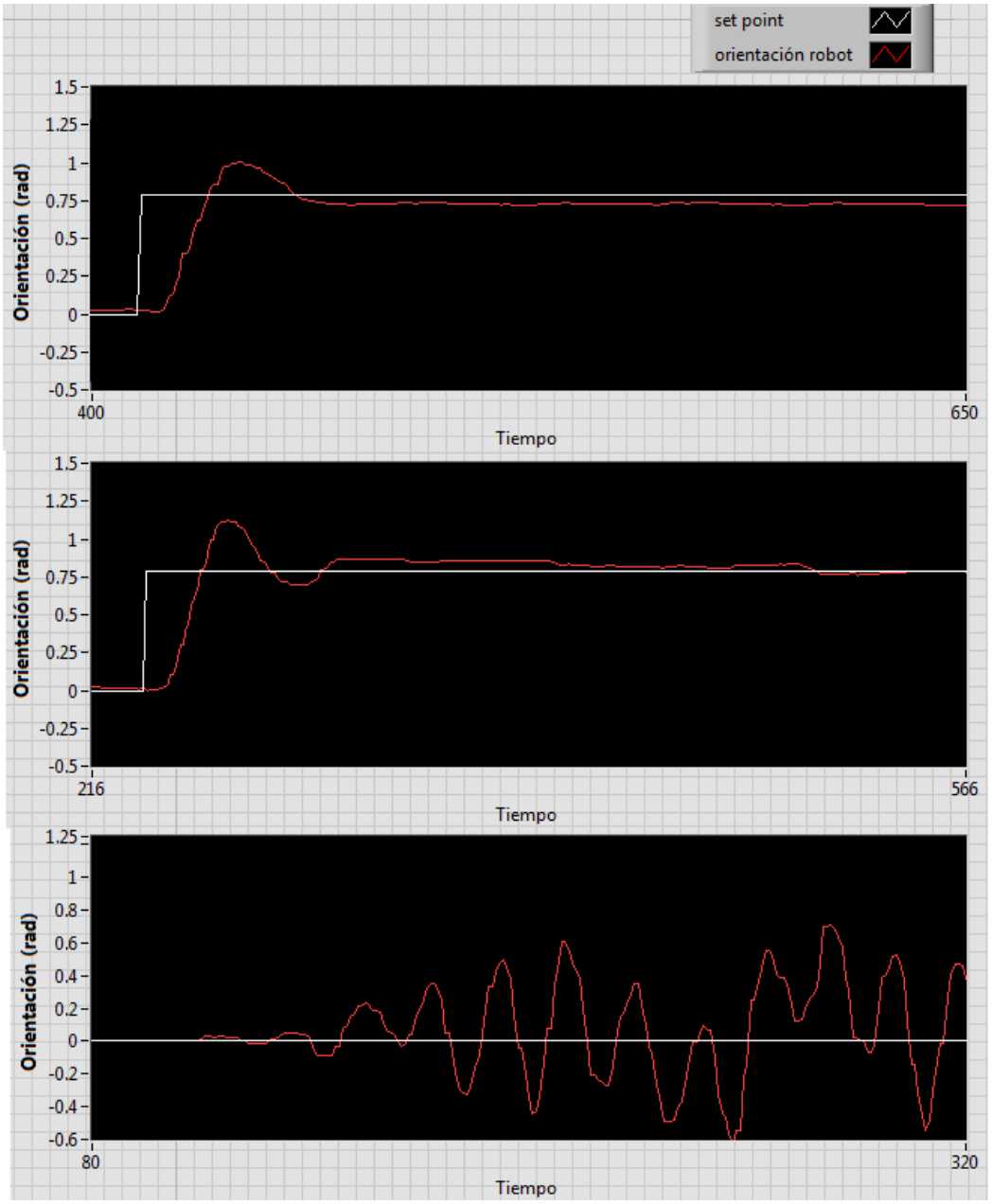
Gráfica 6-1. Oscilaciones sostenidas en la orientación del robot

Al sustituir en la Tabla 6-5 los valores de K_{cr} y P_{cr}, se obtienen los valores mostrados en la Tabla 6-6.

Tabla 6-6. Parámetros sugeridos para el controlador por el método de Ziegler-Nichols

Tipo de controlador	K_p	T_i	T_d
P	1.75	∞	0
PI	1.575	1.4167	0
PID	2.1	0.85	0.2125

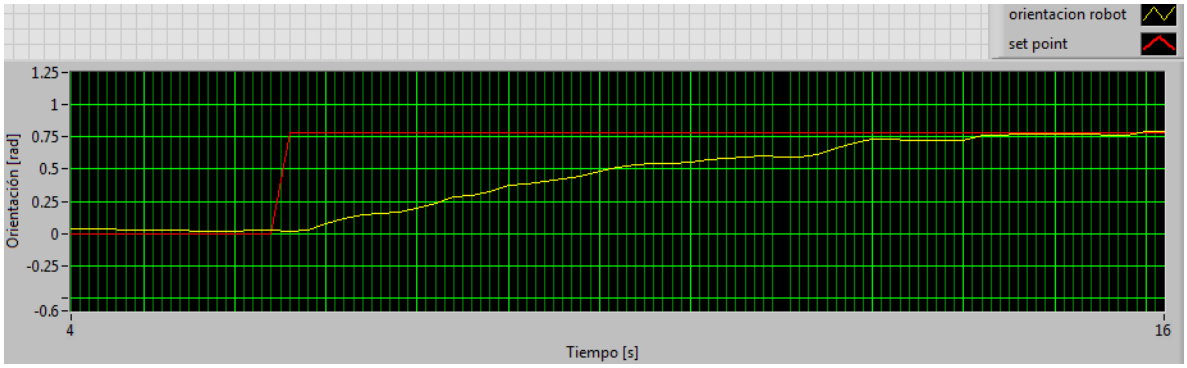
En la Gráfica 6-2 se muestra la respuesta del sistema ante una entrada escalón, con un controlador P, un PI y un PID utilizando los valores de la Tabla 6-6.



Gráfica 6-2. Comparación de la respuesta a escalón con los controladores de la Tabla 6-6

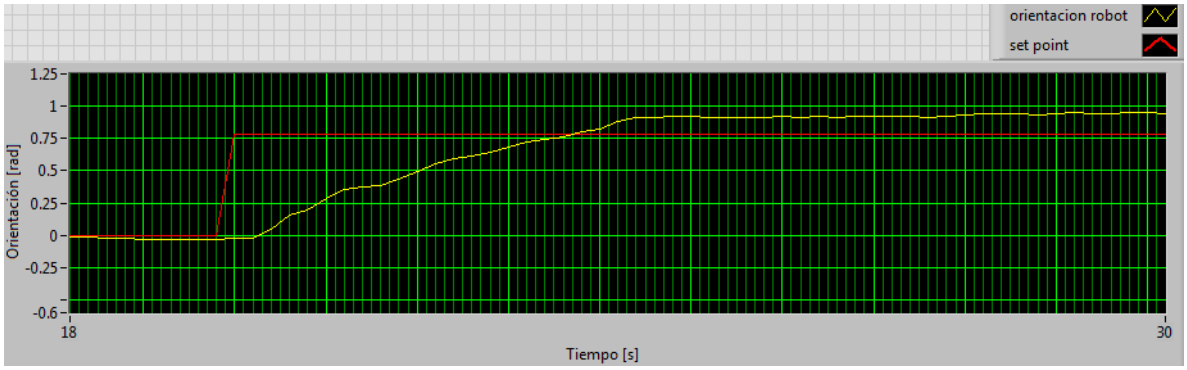
Con el control proporcional se tuvo un sobrepaso de aproximadamente 1 [rad], con el control PI el sobrepaso aumento ligeramente y mejoró el error en estado permanente, y con el control PID se tuvieron oscilaciones en estado estacionario.

Se observó que las oscilaciones con el controlador PID eran rápidas, debido a esto se disminuyó el valor de K_p hasta 0.2 para hacer más lenta la respuesta del sistema. De esta manera y ante una entrada escalón el sistema con el controlador PID tuvo la respuesta mostrada en la Gráfica 6-3.



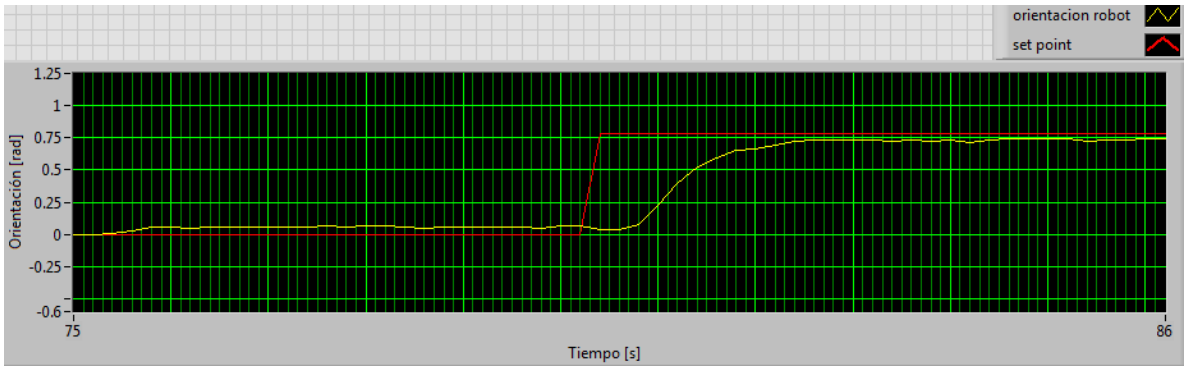
Gráfica 6-3. Respuesta a escalón del controlador PID con K_p = 0.2

Debido a que esta respuesta es muy lenta se fue aumentando poco a poco el valor de K_p, pero al aumentarlo el sobrepaso crecía (Gráfica 6-4).



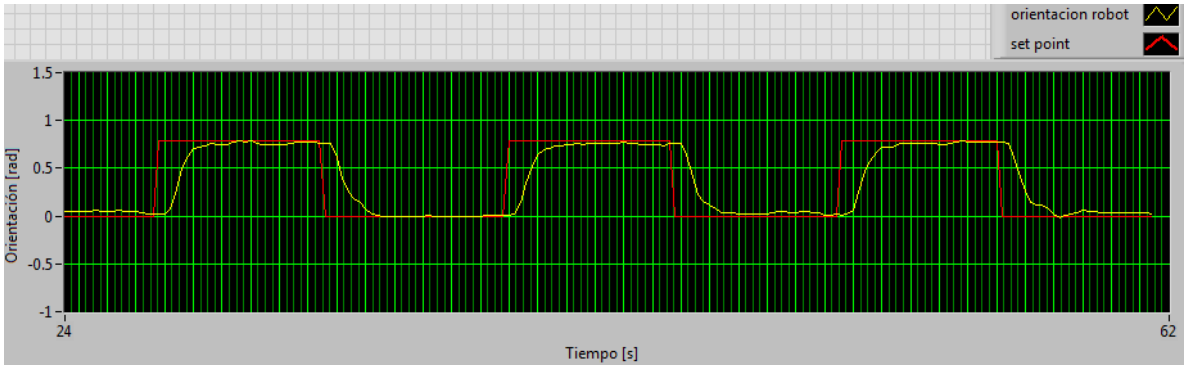
Gráfica 6-4. Respuesta a escalón con sobrepaso del controlador PID

Para disminuir el sobrepaso se aumentó el valor de T_i hasta 5.85. A su vez se fue disminuyendo poco a poco el valor de T_d hasta dar con el valor de 0.05 para disminuir las oscilaciones a medida que se iba alcanzando el set point. Dada la naturaleza del experimento se evitó a toda costa tener oscilaciones en el movimiento del robot, ya que esto a su vez produce errores en la estimación de la posición y orientación del modelo odométrico. La respuesta ante una entrada escalón con estos parámetros se muestra en la Gráfica 6-5.



Gráfica 6-5. Respuesta a escalón sin sobrepaso del controlador PID

Finalmente se aumentó de nuevo K_p hasta el valor de 1.5 para tener una mayor rapidez de respuesta, pero no demasiado para evitar tener nuevamente sobrepasos y oscilaciones. En la Gráfica 6-6 se muestra la respuesta del sistema ante varias entradas escalón con los nuevos parámetros resultantes de este ajuste.



Gráfica 6-6. Respuesta ante varios escalones del controlador PID ajustado

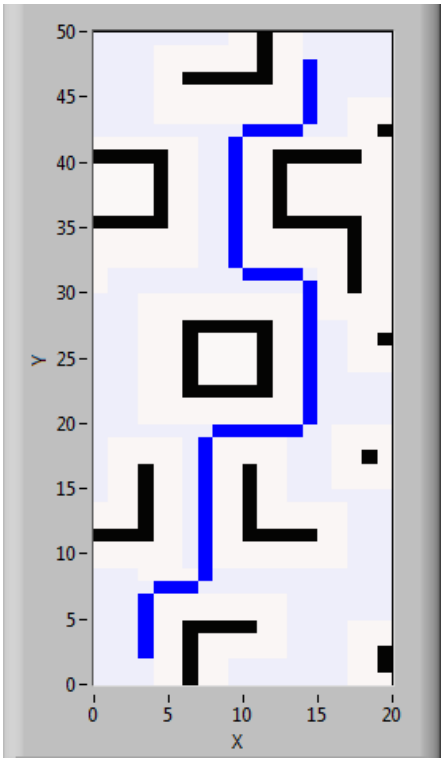
6.3 Pruebas con planeación previa de trayectorias

Con el juego de parámetros obtenido para el controlador se procedió a hacer las pruebas en el espacio de trabajo con la planeación de ruta. En primera instancia se analizó el desempeño del robot al seguir la ruta planeada sin paredes, esto para ajustar lo más posible los valores del controlador de la orientación.

Se hicieron 20 pruebas con el trayecto en el espacio de trabajo para observar el desempeño del robot, cada prueba se calificó como satisfactoria si el robot llegaba a la posición de meta sin atravesar ninguno de los espacios correspondiente a donde se ubican las paredes, y como insatisfactoria si llegaba a atravesar aunque sea por muy poco el espacio correspondiente a las paredes.

La ruta a seguir es la que se muestra en la Gráfica 6-7, donde la posición inicial está en la parte inferior y la posición de meta en la parte superior de la gráfica.

Los resultados de las pruebas con el primer juego de parámetros del controlador se muestran en la Tabla 6-7.

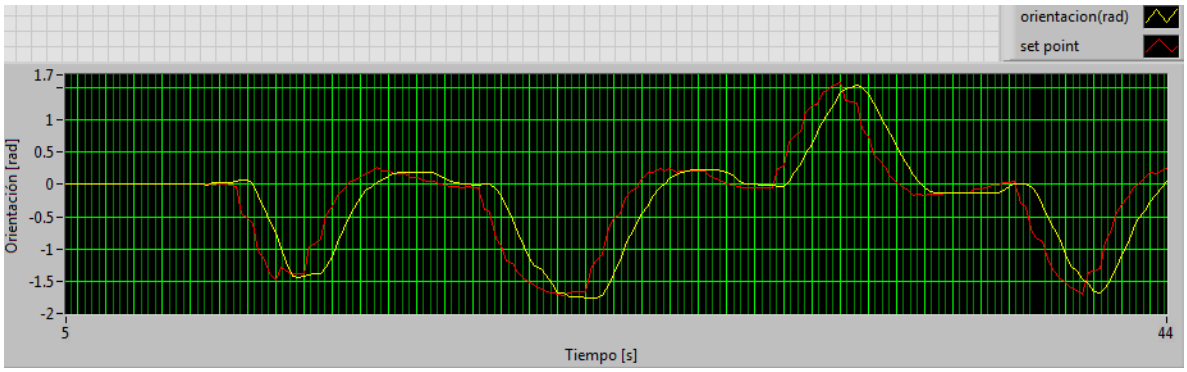


Gráfica 6-7. Ruta a seguir calculada por el algoritmo A*

Tabla 6-7. Resultados primer prueba de seguimiento de ruta

# Prueba	¿Satisfactoria?
1	No
2	Si
3	Si
4	No
5	No
6	No
7	No
8	No
9	No
10	No
11	Si
12	Si
13	No
14	Si
15	Si
16	Si
17	No
18	No
19	No
20	Si

En la Gráfica 6-8 se muestra el cambio de orientación a medida que el robot DaNI sigue la ruta calculada, para una de las pruebas de la Tabla 6-7. La señal *set point* es la orientación indicada por los comandos de movimiento y la señal *orientación(rad)* es la orientación instantánea del robot.



Gráfica 6-8. Cambio de la orientación en el seguimiento de la ruta

De estas pruebas se concluye que se tiene una eficacia del 40%, la cual es significativamente baja. Del análisis cualitativo del desempeño del robot, se observó que las oscilaciones que hace el robot a medida que avanza provocan errores de posición que hacen que en algunas ocasiones el robot gire antes de lo debido, este es el principal error que se detecto que provoca que el robot atraviese por donde estarían ubicadas las paredes.

Para mejorar el desempeño se volvieron a ajustar experimentalmente los parámetros del controlador de la orientación de manera que el trayecto sin paredes se hiciera correctamente la mayor cantidad de veces posible. En este caso se fue disminuyendo el valor de Td hasta 0.01.

Cabe señalar que debido a la *distancia de franqueamiento de obstáculos* (*obstacle clearance distance*). La distancia de franqueamiento de obstáculos específica la distancia mínima a la cual el punto de guiado del vehículo puede acercarse a un obstáculo o a la meta en cuestión. En el caso de que la distancia de franqueamiento de obstáculos contemple toda la estructura del robot sería el frente del vehículo el que alcanzaría la posición de meta y no el punto de guiado, como se ilustra en la Figura 6-4.

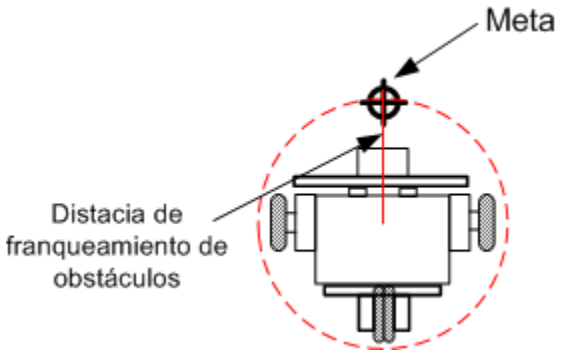


Figura 6-4. Representación de la distancia de franqueamiento de obstáculos

Para lograr un mejor seguimiento de la trayectoria se disminuyó lo mayormente posible el valor de la *distancia de franqueamiento,* de esta manera el punto de guiado del robot se acerca más a la posición de meta en cuestión.

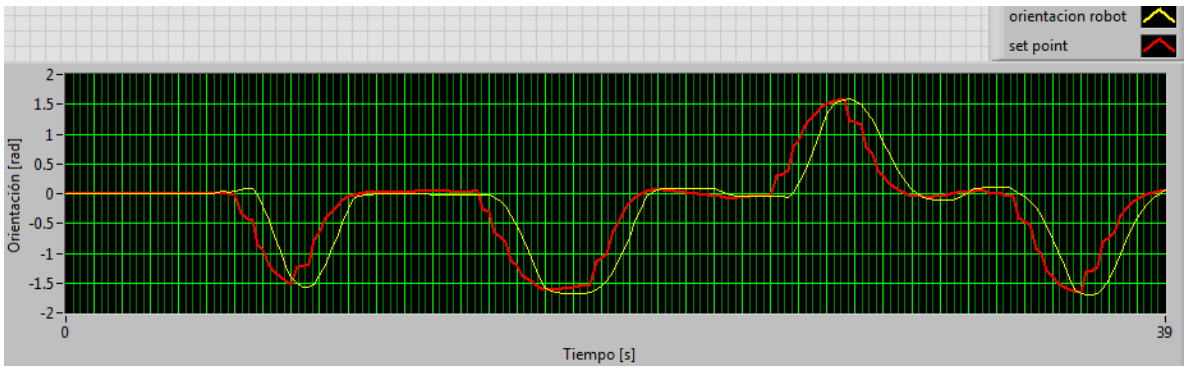
Con estos ajustes tanto del seguimiento de la trayectoria como del control de la orientación, se obtuvieron los resultados mostrados en la Tabla 6-8.

Tabla 6-8. Resultados segunda prueba de seguimiento de ruta

# Prueba	¿Satisfactoria?
1	Si
2	Si
3	No
4	Si
5	Si
6	No
7	Si
8	No
9	No
10	Si
11	Si
12	Si
13	Si
14	Si
15	No
16	Si
17	No
18	No
19	No
20	Si

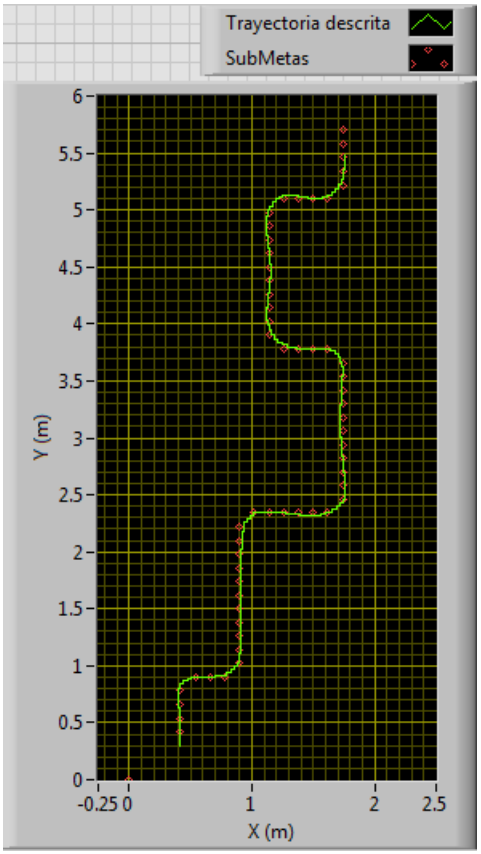
De los resultados obtenidos en la Tabla 6-8 se observó que el desempeño del robot mejoró con respecto a los resultados de la prueba mostrada en la Tabla 6-7.

En la Gráfica 6-9 se muestra el cambio de orientación del robot DaNI (*orientación robot)* a medida que el comando de orientación (*set point*) cambia debido a la ruta calculada por el algoritmo A*, para una de las pruebas (satisfactorias) de la Tabla 6-8. Comparando la Gráfica 6-9 con la Gráfica 6-8 se puede observar que las oscilaciones (señal *orientación robot*) son menores.



Gráfica 6-9. Cambio de la orientación del robot en el seguimiento la ruta, con los nuevos parámetros del controlador PID

En la Gráfica 6-10 se observa la trayectoria descrita por el punto de guiado del robot DaNI en su avance hacia la meta, resultado de la estimación odométrica.



Gráfica 6-10. Trayectoria descrita por el punto de guiado con los parámetros finales del controlador de orientación

Posteriormente se procedió a realizar la prueba pero ahora con las paredes que forman los pasillos en el espacio de trabajo como se observa en la Figura 6-5.

Figura 6-5. Espacio de trabajo con paredes

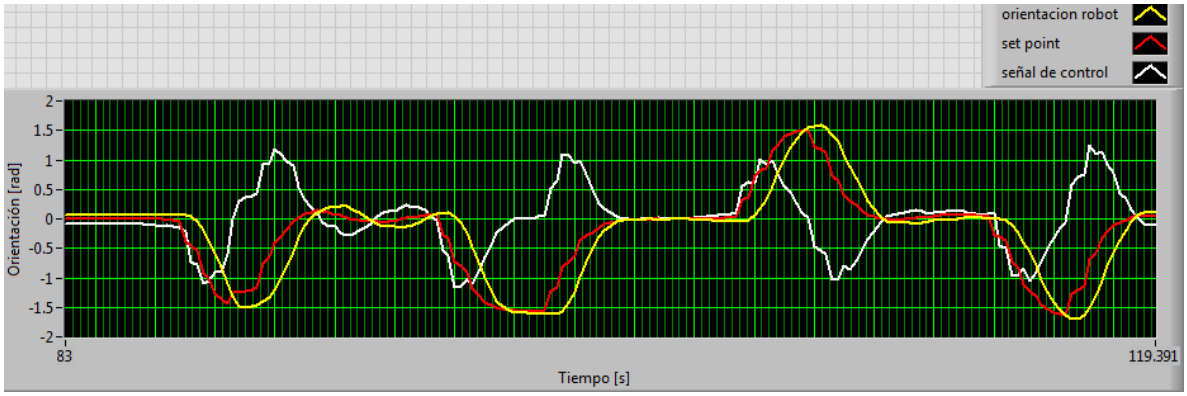
El programa que se encarga de desacelerar al robot a medida que se acerca a una meta (ver Figura 5-53) se modifico de manera que solo desacelerara en presencia de un obstáculo, esto se hizo debido a que el movimiento se volvió ligeramente intermitente por la presencia de los obstáculos, de esta manera el movimiento volvió a ser continuo. Los resultados se muestran en la Tabla 6-9.

Tabla 6-9. Resultados del seguimiento de ruta en la prueba con paredes

# Prueba	¿Satisfactoria?
1	Si
2	Si
3	No
4	Si
5	Si
6	No
7	Si
8	Si
9	Si
10	Si
11	Si
12	No
13	Si
14	Si
15	Si
16	Si
17	Si
18	No
19	Si
20	Si

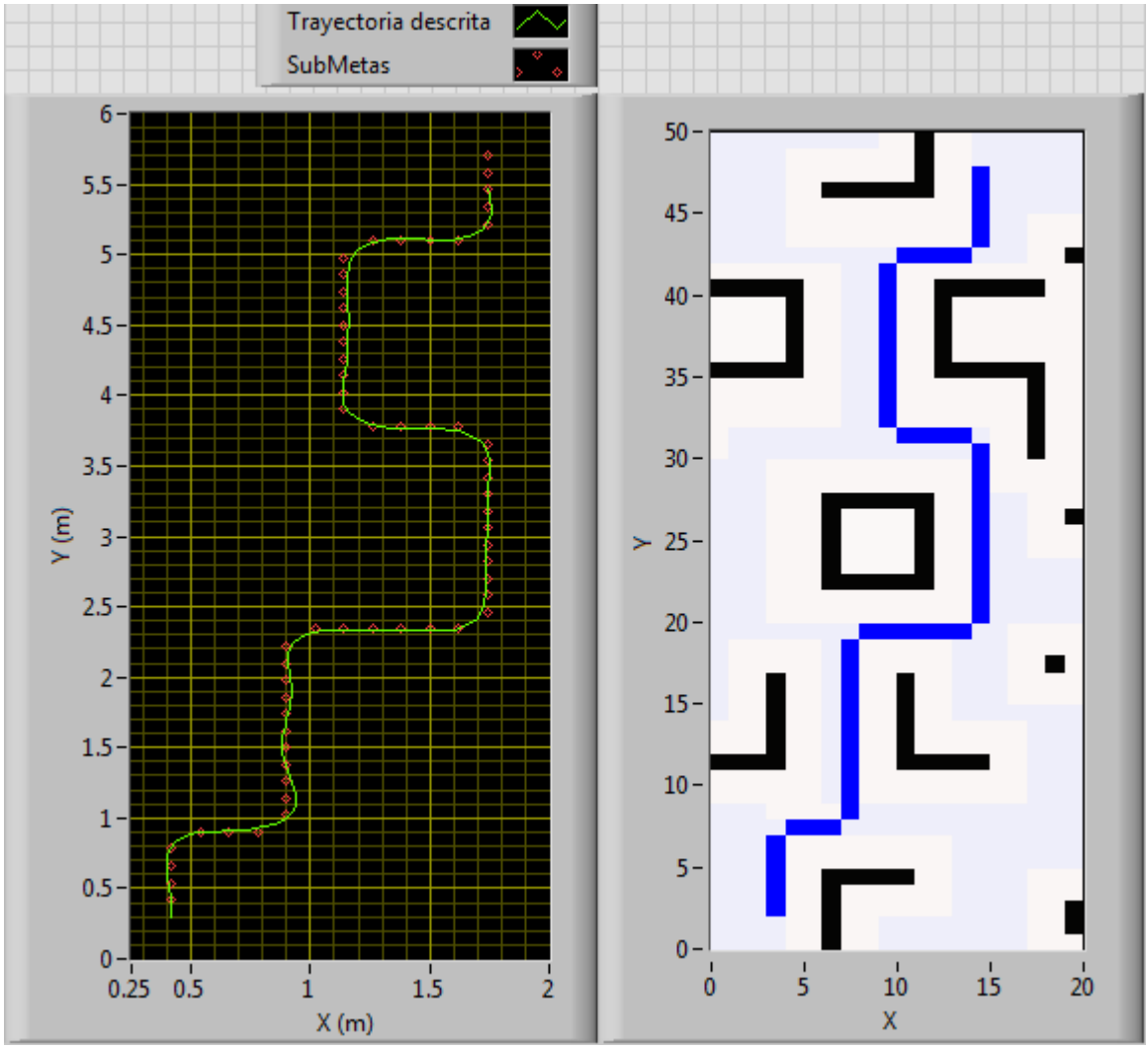
Los resultados mostrados en la Tabla 6-9 muestran una eficiencia del 70% por parte del robot al seguir la ruta en el espacio de trabajo. El error presente en el modelo odométrico es el responsable de que existan ocasiones en las cuales no se realice el seguimiento de la ruta planeada de forma exacta.

En la Gráfica 6-11 se muestra el cambio de orientación del robot DaNI (*orientación robot*) a medida que el comando de orientación (*set point*) cambia, también se muestra la señal de control, esta señal es la instrucción que es enviada a los motores para lograr el movimiento de orientación necesario para seguir la trayectoria planeada por el algoritmo A*, todo esto para una de las pruebas (satisfactorias) de la Tabla 6-9.



Gráfica 6-11. Variación de la orientación del robot, del comando de orientación y de la señal de control

Una vez más se muestra la trayectoria descrita por el punto de guiado al seguir la ruta planeada (ver Gráfica 6-12).



Gráfica 6-12. Trayectoria descrita por el punto de guiado y ruta planeada, para la prueba final de seguimiento de ruta

Para intentar mejorar aún más el desempeño, después de observar cómo el robot DaNI se movía conforme avanzaba a la meta, la estimación del ángulo de orientación se sustituyó por una medición directa del ángulo de orientación utilizando un giroscopio, en este caso y por practicidad se utilizó el giroscopio incorporado en un dispositivo iPod, los resultados obtenidos fueron prácticamente los mismos que con la estimación odométrica, es por eso que no se incluyó en el esquema final, aún así y derivado de esto, en el apéndice B se incluye la manera de cómo leer el valor de la orientación del giroscopio de un ipod y mandar este valor al robot DaNI para su incorporación en el esquema de navegación autónoma propuesto. Derivado de esto último y como una aportación adicional, en el apéndice C se muestra como controlar manualmente al robot DaNI utilizando el giroscopio del dispositivo iPod.

6.4 Pruebas con obstáculos no previstos

Una de las principales ventajas que tiene este esquema de control autónomo del robot DaNI, es que la ruta a seguir no está sujeta a giros de ángulos rectos, ya que puede ser de cualquier forma, e incluso realizar evasión de obstáculos no previstos en la planeación previa de la ruta a seguir.

Para poder hacer la evasión de obstáculos no previstos fue necesario ajustar los *umbrales de detección de obstáculos*, estos umbrales determinan el rango en el cual el algoritmo VFH identifica un objeto y a partir de que distancia lo considera un obstáculo.

El *umbral interno* especifica la distancia entre el sensor y un obstáculo a la cual el algoritmo VFH considera la dirección bloqueada. El *umbral externo* especifica la distancia entre el sensor y un obstáculo a partir de la cual el algoritmo VFH ya no considera la dirección bloqueada. Los umbrales son medidos a partir de la distancia de franqueamiento de obstáculos alrededor del vehículo (ver Figura 6-4).

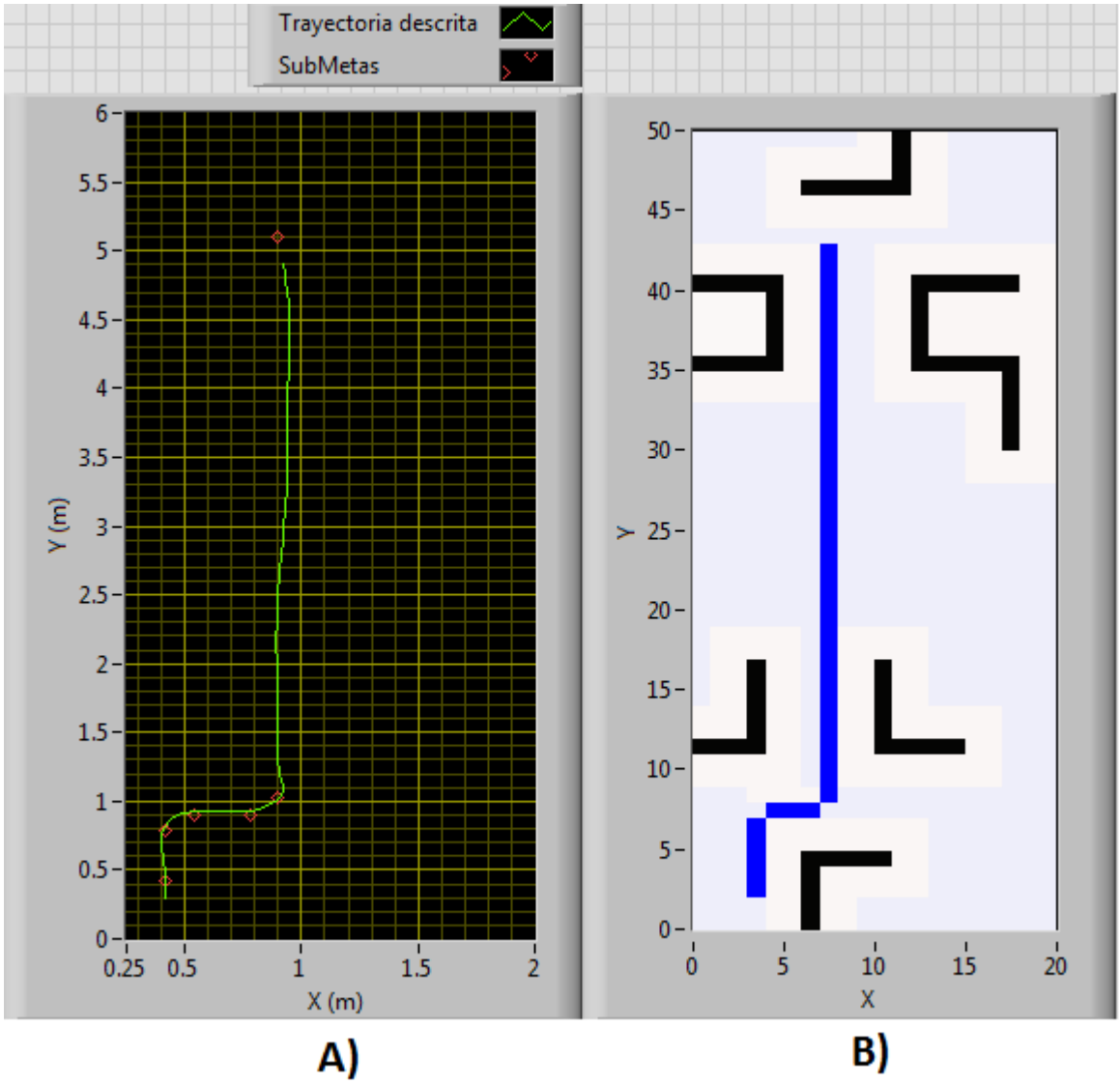
Los valores para la distancia de franqueamiento, el umbrale interno y externo fueron de 0.3312, 0.2649 y 0.5299 [m] respectivamente, con estos valores se logra la evasión de obstáculos imprevistos.

Véase en la Figura 6-6-A) el espacio de trabajo, y en la Figura 6-6-B) el mismo espacio de trabajo pero con un obstáculo no previsto.

Figura 6-6. Espacio de trabajo sin obstáculo y con un obstáculo no previsto en la planeación previa de trayectorias

De esta manera la ruta a seguir sería la mostrada en la Gráfica 6-13-B), como se puede observar en esta gráfica, el obstáculo no está contemplado en la ruta previa. En la

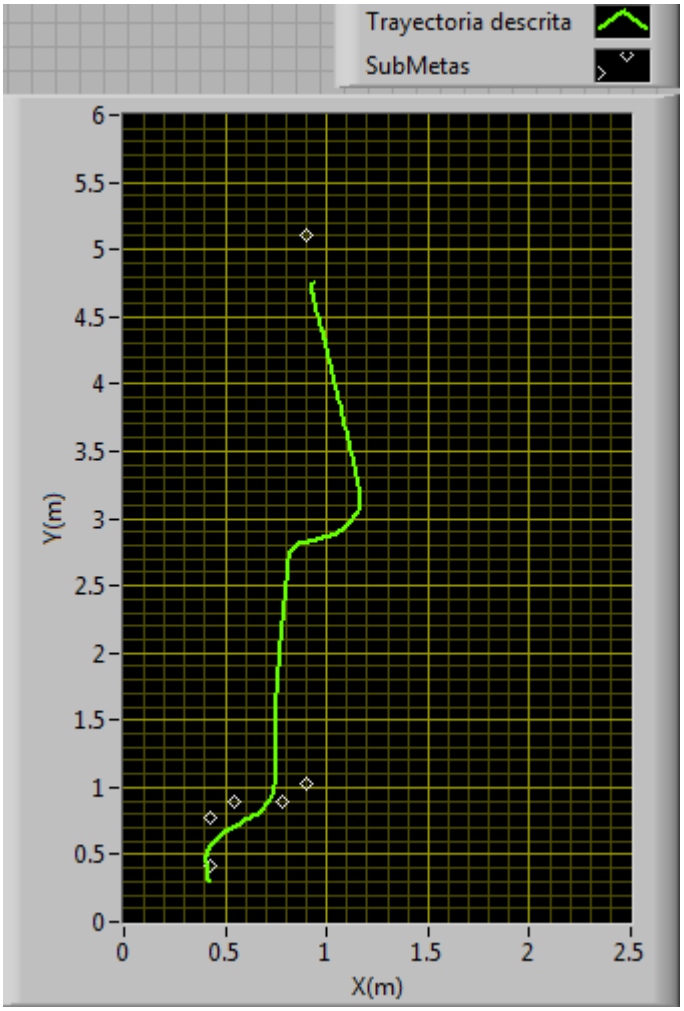
Gráfica 6-13-A) se observa la trayectoria que describe el punto de guiado al seguir la ruta sin obstáculos imprevistos.



Gráfica 6-13. Trayectoria descrita sin obstáculo y ruta a seguir sin contemplar el obstáculo

Entonces, se hizo un ligero ajuste en la forma que se compone la ruta calculada por el algoritmo A*, de manera que se tengan las metas mínimas y suficientes para componer la ruta completa como se muestra en la Gráfica 6-13-A), los rombos son las submetas mínimas que componen la ruta a seguir de la Gráfica 6-13-B).

Finalmente, de esta manera el robot DaNI puede evadir el obstáculo imprevisto no contemplado por la planeación de ruta del algoritmo A*, mientras al mismo tiempo se dirige a la dirección de meta indicada (ver Gráfica 6-14).



Gráfica 6-14. Trayectoria descrita por el punto de guiado al evadir un obstáculo no previsto

En la Gráfica 6-14 se observa la trayectoria descrita por el punto de guiado del robot DaNI, la curva que aparece a mitad del recorrido corresponde a la evasión del obstáculo no previsto en la planeación previa de ruta.

Conclusiones

El objetivo planteado en este trabajo se cumplió satisfactoriamente, ya que se logró hacer que el robot recorriera la ruta más corta entre una posición inicial y la meta indicada, en el espacio de trabajo con obstáculos para una velocidad lineal máxima de 0.2 m/s. Esto gracias a la implementación de diversos algoritmos como él A* para la planeación de rutas, el histograma de campo vectorial para evasión de obstáculos, y la teoría de control para el control del movimiento del robot. Una de las principales ventajas de este esquema de navegación autónoma es el hecho de que se maneja tanto la planeación y seguimiento de rutas como la evasión de obstáculos, dándole una mayor autonomía de movimiento al robot. Además el seguimiento de rutas no está limitado a giros en ángulos rectos (como en [9]), que si bien fue el experimento en el que se probó el robot por ser el que presenta el mayor reto en el direccionamiento, las rutas pueden ser de cualquier forma, gracias a la forma en que se direcciona al robot. Adicionalmente este trabajo servirá de referencia para el desarrollo de aplicaciones en robótica móvil en el laboratorio de instrumentación virtual.

Así mismo, se observó la necesidad de conocer los fundamentos teóricos que sustentan el funcionamiento del robot, por ejemplo el caso del modelo odométrico al tomar en cuenta solo la cinemática del robot presenta un error inherente que crece a medida que avanza el tiempo; otro ejemplo es el relacionado con los sensores y los componentes mecánicos, los cuales al tener un rango de operación y ciertas características de resolución (en el caso de los sensores), están limitados a cierto tipo de aplicaciones. Es por ello que el comportamiento del robot no es ideal, pero es posible obtener un buen desempeño en el sistema de robótica dentro del objetivo propuesto; es debido a todo esto que es necesario conocer la teoría, limitaciones y principios de funcionamiento del robot y todos sus componentes mecánicos y electrónicos.

Una función de gran ayuda fue el simulador incorporado en el módulo LabVIEW Robotics, ya que permitió probar algunos segmentos del programa antes de ser implementados en el laboratorio con el robot real cuando no se podía trabajar con este.

Respecto a los algoritmos implementados de control, planeación de rutas y de evasión de obstáculos, se observó que es necesario conocer su principio de funcionamiento para saber interpretar sus resultados derivados y aplicarlos de forma correcta.

El hecho de contar únicamente con dos tipos de sensores: el sensor ultrasónico y los codificadores ópticos, permitió explotar sus capacidades y visualizar las mejoras que traería consigo el hecho de agregar otro tipo de sensores. Por ejemplo, si se agregara un sensor laser o una cámara para la detección de objetos, el robot podría elaborar por sí mismo el mapa del espacio de trabajo volviéndose más autónomo. Otra mejora a realizar es la correspondiente al error presente en el modelo odométrico, ya que este es el responsable de que en algunas ocasiones el robot se acercara demasiado a las paredes

del espacio de trabajo, una manera sencilla de mitigar esto es implementar un modelo para este error como se explica en [1]. Un forma más adecuada sería complementar la odometría con un sistema de visión montado de forma que se observe todo el espacio de trabajo, de esta manera se conocería con certeza la posición y orientación del robot; lo que permitiría implementar un control de posición en el seguimiento de las rutas planeadas como se describe en [8].

La robustez de la estructura del robot es un factor importante en la implementación de algoritmos de planeación y evasión de obstáculos, ya que el robot no está exento de colisiones con objetos a medida que se van haciendo pruebas, es por eso que se debe revisar periódicamente el buen funcionamiento de los componentes fiscos. En este trabajo el mal funcionamiento de un motor al final de los experimentos causó, a su vez, un mal funcionamiento en la rutina que anteriormente realizaba correctamente, fue hasta que se detectó esta anomalía, que se cambió el motor y se corrigió el problema.

Personalmente la realización de este trabajo, si bien fue demandante, resultó sumamente interesante, ya que involucra la aplicación de diversas disciplinas y temas aprendidos a lo largo la carrera, desde conceptos básicos de cinemática hasta la implementación de teoría de control en un robot móvil, pasando por la implementación de diversos algoritmos computacionales (en este caso en LabVIEW). También me permitió comprender la importancia y complejidad que tienen este tipo de sistemas móviles de robótica.

Finalmente, la experiencia de haber presentado este trabajo en diversas exposiciones y en un congreso, resulto muy enriquecedora y satisfactoria.

Apéndice A

Instalación del toolkit Control & Simulation en el robot DaNI

Con el robot DaNI encendido y conectado a la computadora huésped se debe abrir la aplicación *Measurement & Automation* de LabVIEW.

La tarjeta sbRIO-9632 del robot debe aparecer bajo la sección *Remote Systems* como se observa en la Figura 7-1, al expandir esta sección se hace doble click en la opción *Software.* En la opción *Software* se hace click derecho y se selecciona *Add/remove Software.*

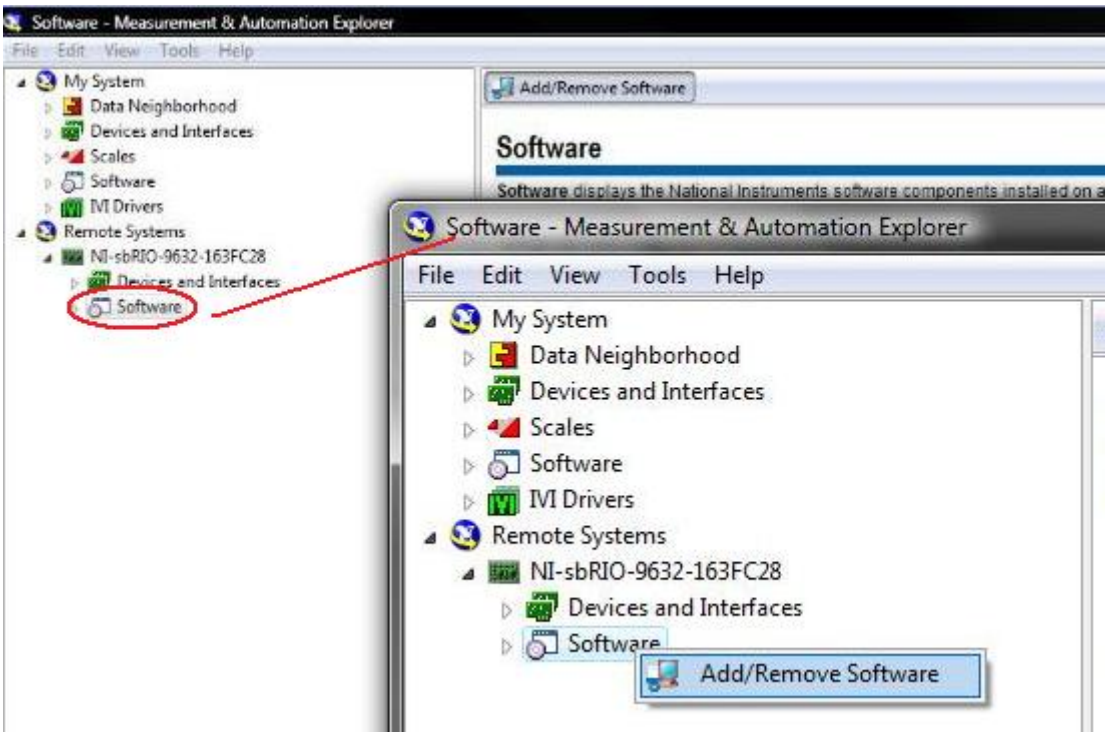


Figura 7-1. Instalación manual de software en el robot DaNI

A continuación aparecerá la ventana mostrada en la Figura 7-2, y se hace click en *Next*.

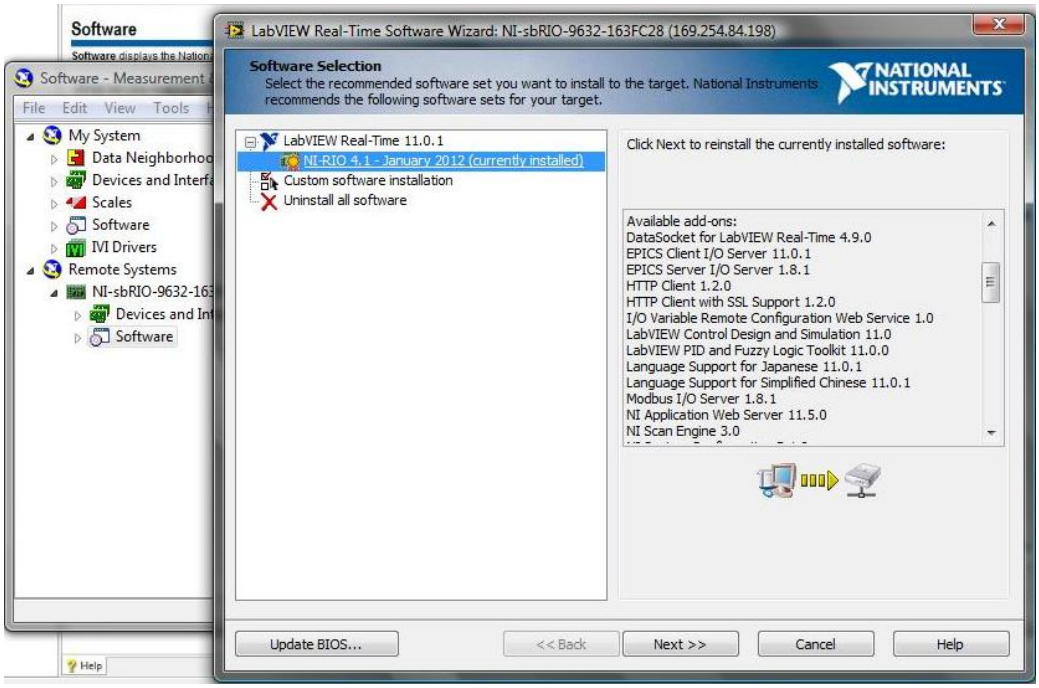


Figura 7-2. Software a instalar en el robot DaNI

En la ventana que aparece a continuación (ver Figura 7-3) se enlistan las opciones de software que pueden ser instaladas en el robot DaNI, de las cuales están seleccionadas las casillas de las opciones recomendadas, es importante no quitar las marcas de ninguna de estas casillas, ya que estas opciones incluyen el software básico necesario para la programación del robot. De las opciones no seleccionadas se marca la casilla correspondiente al toolkit *Control and Design Simulation* como se observa en la Figura 7-3, y se hace click en *Next.*

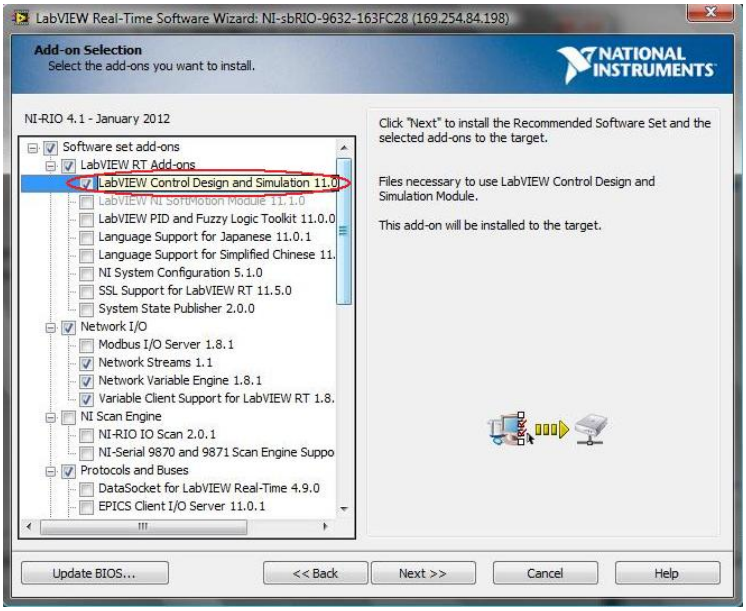


Figura 7-3. Selección de software a instalar en el robot DaNI

Posteriormente aparecerá la ventana de la Figura 7-4 donde se enlista el software que se instalará en el robot DaNI.

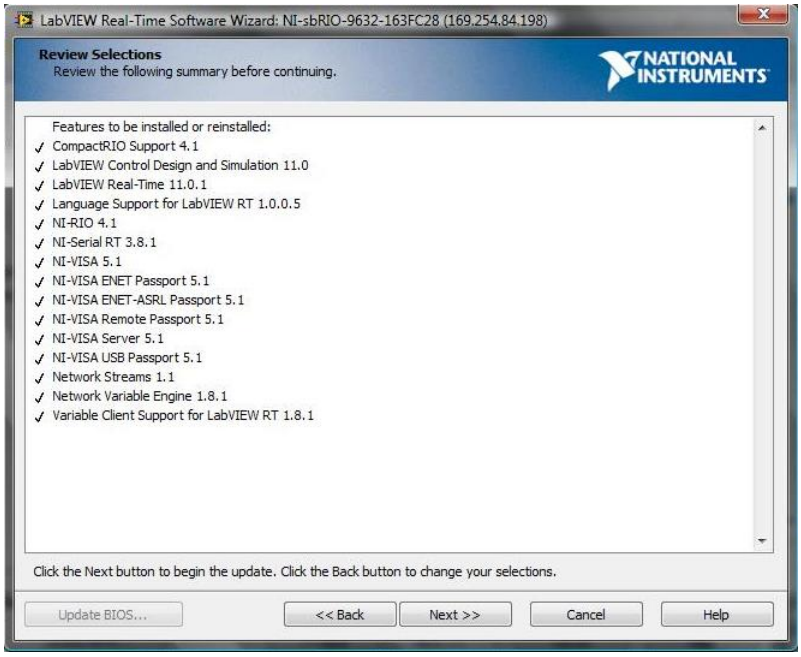


Figura 7-4. Software que se instalará en el robot DaNI

Después de dar click en *Next* comenzará la instalación del software (ver Figura 7-5).

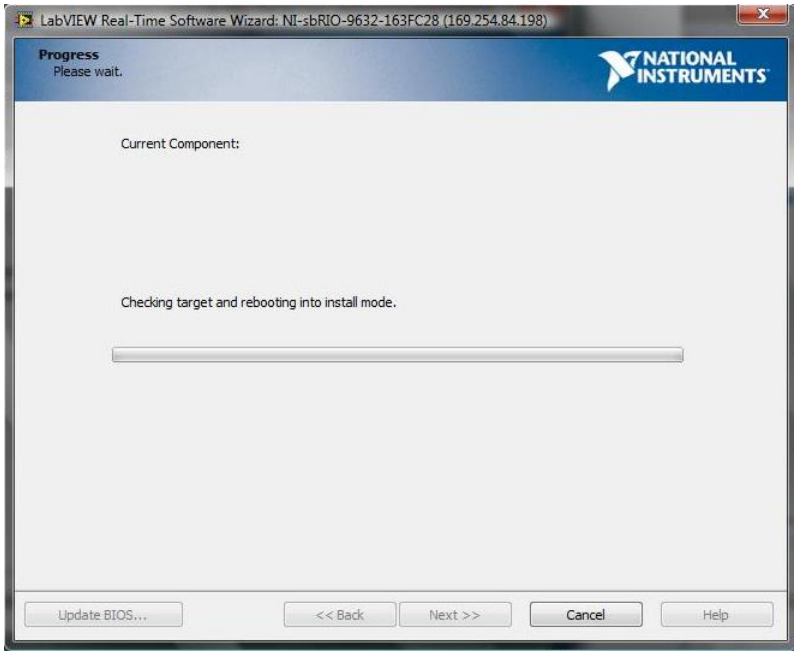


Figura 7-5. Instalando el software en el robot DaNI

Finalmente aparecerá la venta de la Figura 7-6 indicando que la actualización del software del robot DaNI fue exitosa.

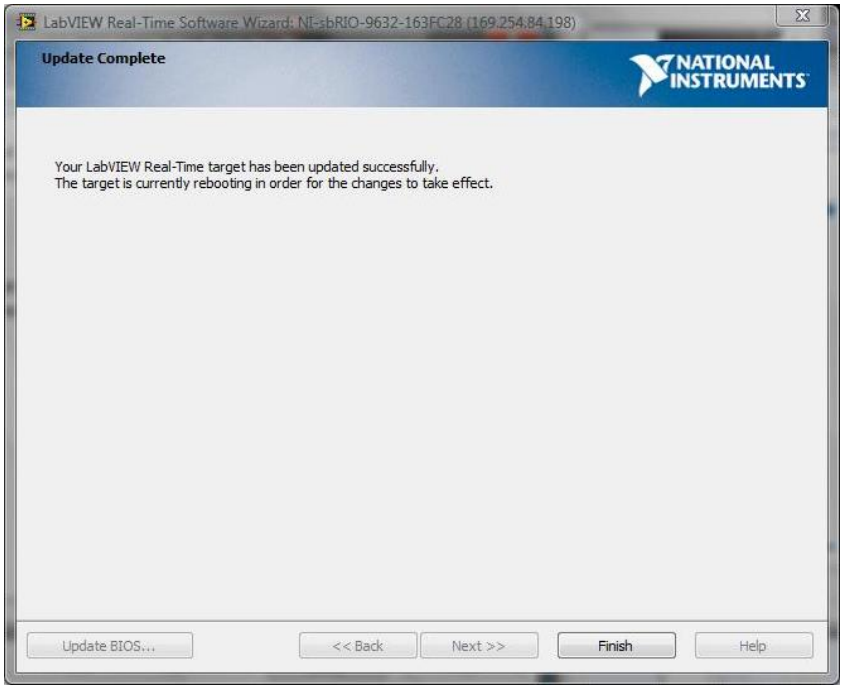


Figura 7-6. Instalación exitosa de software en el robot DaNI

Apéndice B

Medición del ángulo de orientación del robot DaNI utilizando un dispositivo iOS

Para medir el ángulo de orientación del robot DaNI utilizando un dispositivo iOS se hace uso de las funciones del toolkit *NI Open Sound Control.* Para instalar este toolkit se deben seguir los siguientes 2 pasos:

Paso 1**:** Descargar e Instalar el software *VI Package Manager (VIPM),* este software se puede descargar de la siguiente liga:

http://jki.net/vipm/download

En esta página (Figura 8-1) se selecciona la versión del sistema operativo, se introduce una dirección de correo electrónico y se presiona el botón de descarga.

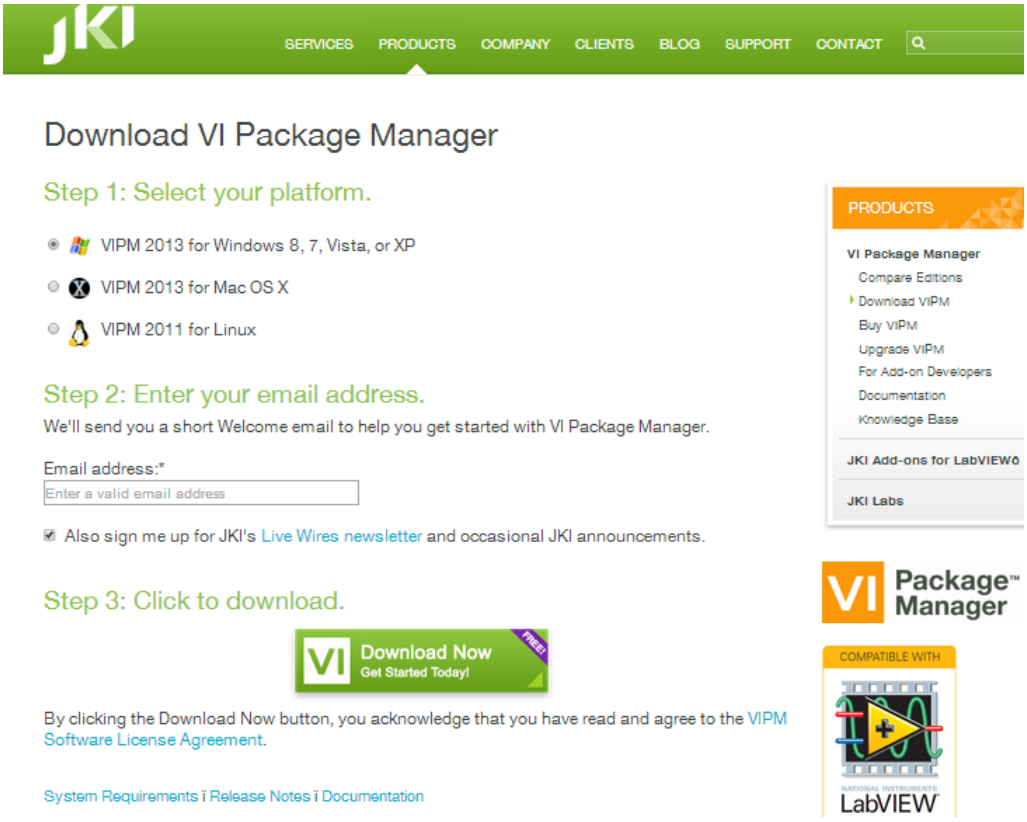


Figura 8-1. Página de descarga del software VIPM

Paso 2: Descargar e instalar el toolkit *NI Open Sound Control.*

Una vez instalado el *VIPM* se abre el programa, de la ventana que aparece (ver Figura 8-2) se selecciona la versión instalada de LabVIEW, y de la lista de toolkits se hace clic derecho sobre el toolkit *NI Open Sound Control*, y se selecciona *Install.*

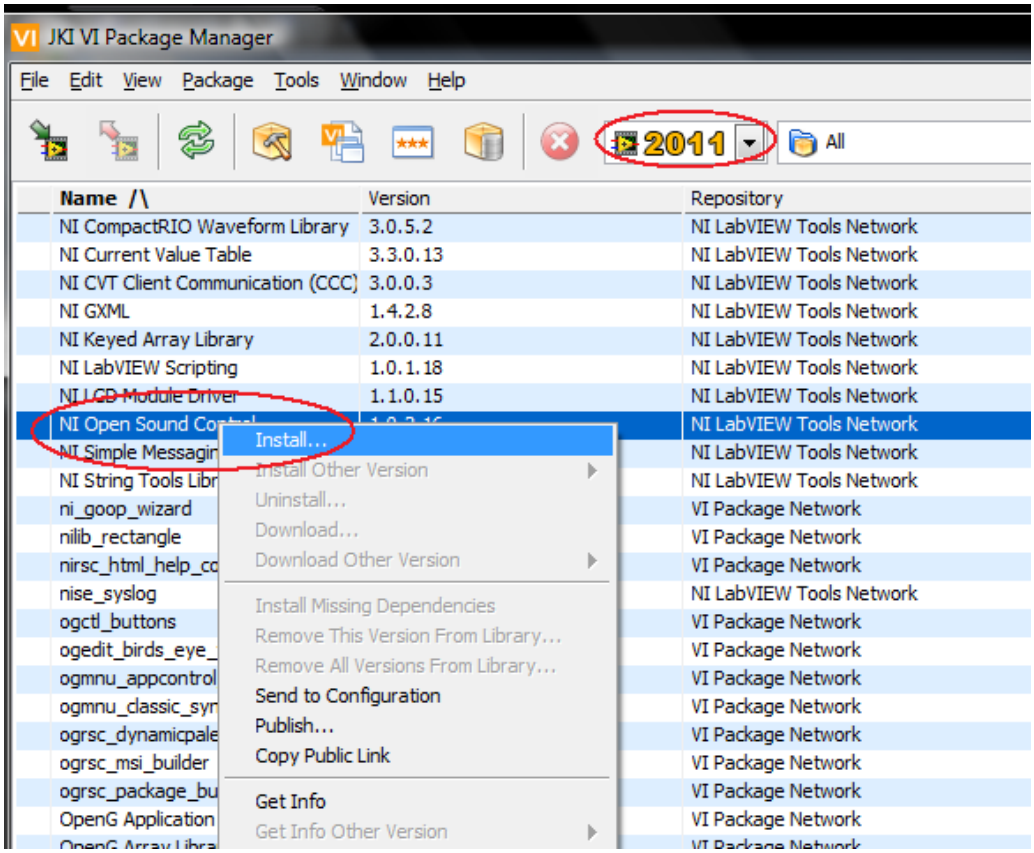


Figura 8-2. Instalación del toolkit utilizando el VIPM

Posteriormente aparecerá la ventana mostrada en la Figura 8-3 y se hace clic en el botón *Continue.*

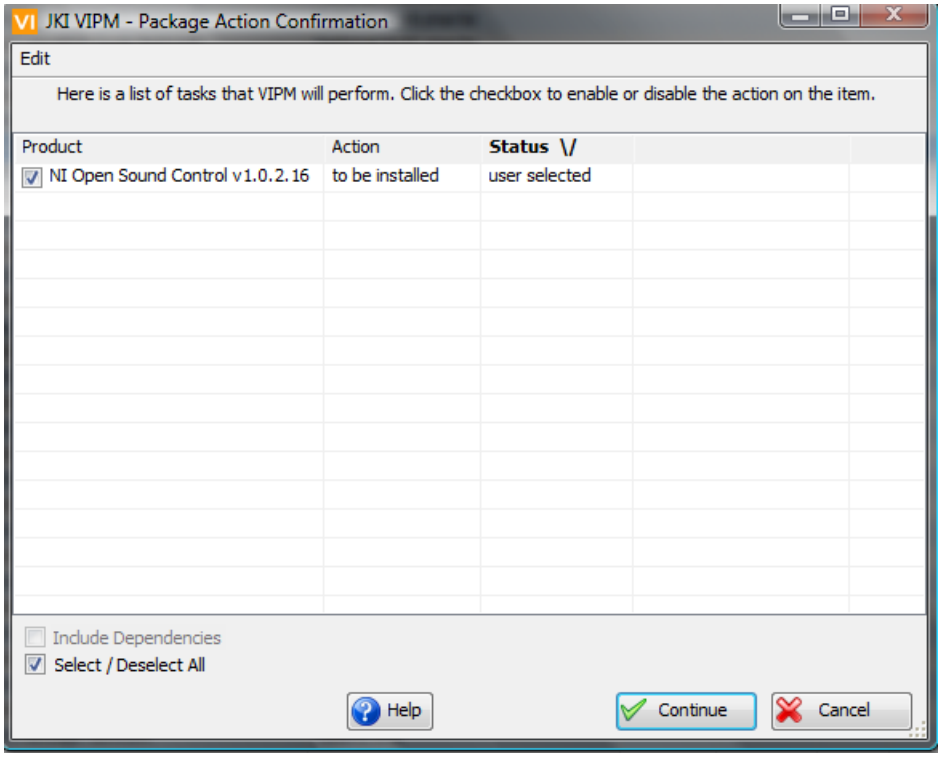


Figura 8-3. Toolkit a instalarse

Cundo el toolkit se instale exitosamente aparecerá la ventana de la Figura 8-4.

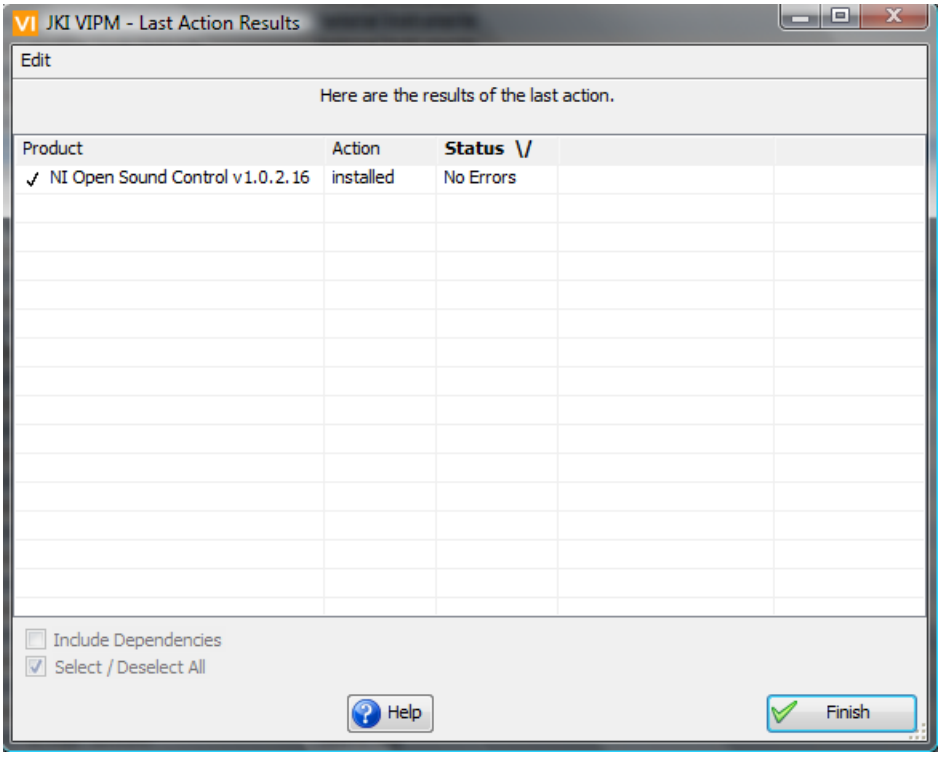


Figura 8-4. Instalación exitosa del toolkit

Después de instalar exitosamente el toolkit, en el VIPM aparecerá el icono de LabVIEW junto al nombre del toolkit (ver Figura 8-5).

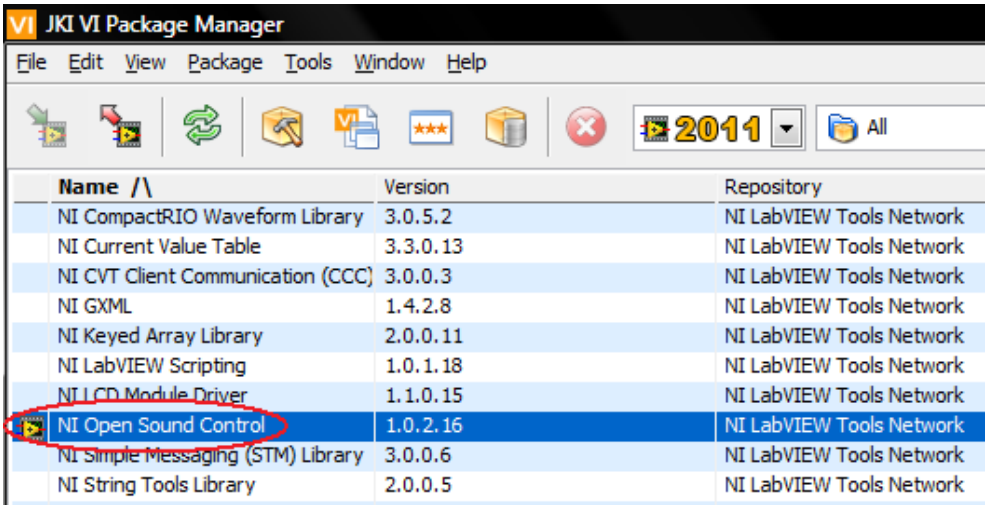


Figura 8-5. Toolkit instalado correctamente en LabVIEW

Una vez instalado el toolkit *NI Open Sound Control,* es necesario instalar la app GyrOSC (ver Figura 8-6) en el dispositivo iOS (un iPod en este caso), se puede utilizar alguna otra app similar, solo es necesario ajustar el código de LabVIEW aquí propuesto ya que la manera en que envían los datos dichas aplicaciones difieren entre si.

Figura 8-6. Icono de la app GyrOSC

La app GyrOSC permite envíar los datos de los sensores de movimiento del dispositivo iOS a otras aplicaciones (como LabVIEW) que esten en la misma red, en formato OSC (Open Sound Control) a través de WiFi. De esta manera se pueden controlar aplicaciones de LabVIEW utilizando los datos del giroscopio del dispositivo iOS.

En el dispositvo iOS, una vez instalada la app GyrOSC se debe introducir en esta, la dirección IP de la computadora huesped (host) que se encuentra en la misma red WiFi que el dispositivo iOS, y el número del puerto al que se enviaran los datos (ver Figura 8-7).

Figura 8-7. Configuración IP de destino y puerto

En la app se deben seleccionar los datos que serán enviados a tarves de la red WiFi, en este caso solo se utilizarán los datos del giroscopio, como se observa en la Figura 8-8.

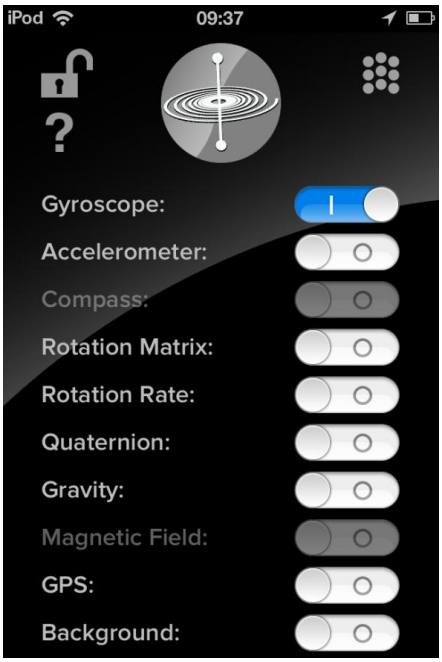


Figura 8-8. Selección de datos de los sensores del iPod que serán enviados

A continuación (Figura 8-9) se muestra el código de LabVIEW necesario para adquirir correctamente los datos provenientes del giroscopio del dispositivo iOS.

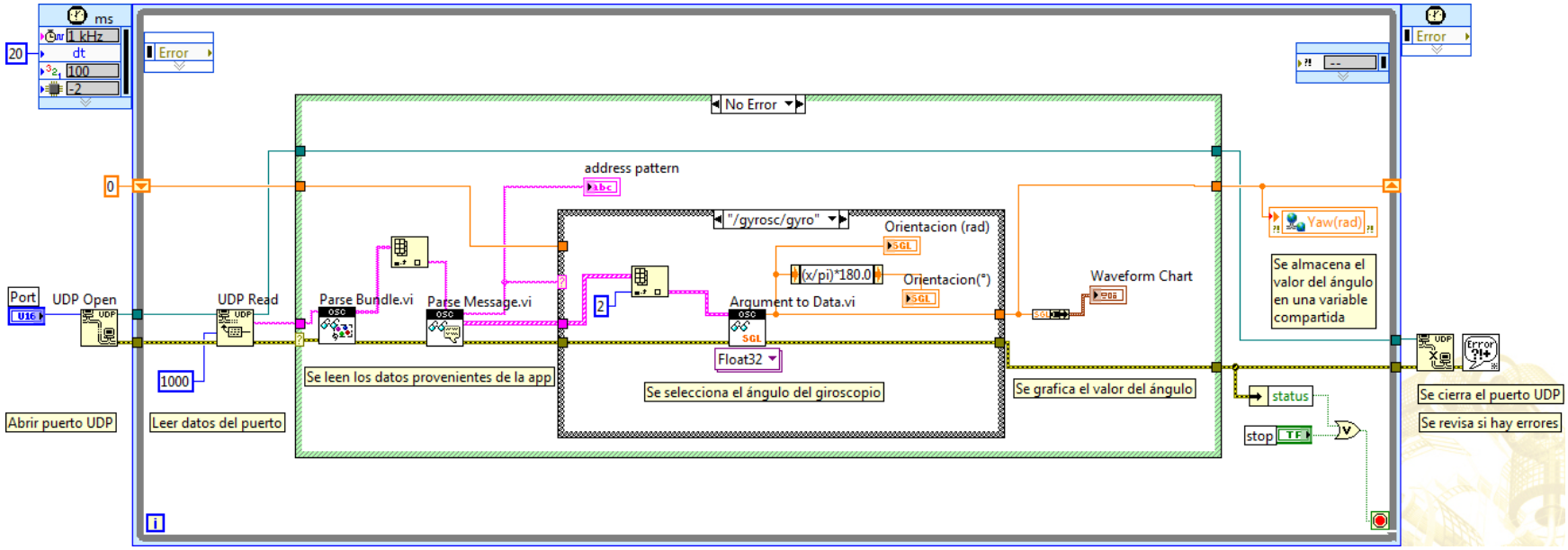


Figura 8-9. Programa para la adquisición de los ángulos del giroscopio del dispositivo iOS

En caso de que el VI *UDP Read* entregue un error de *time out* se tiene el código mostrado en la Figura 8-10.

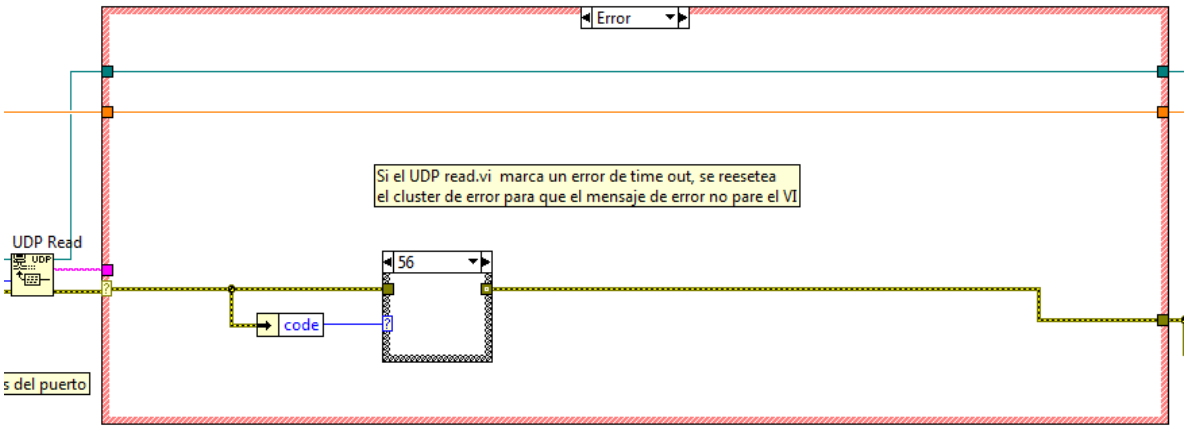


Figura 8-10. Caso para un error en el VI UDP read

Y en el caso de que los datos enviados por la app no correspondan al ángulo de *yaw* del girocopio se tiene el codigo mostrado en la Figura 8-11.

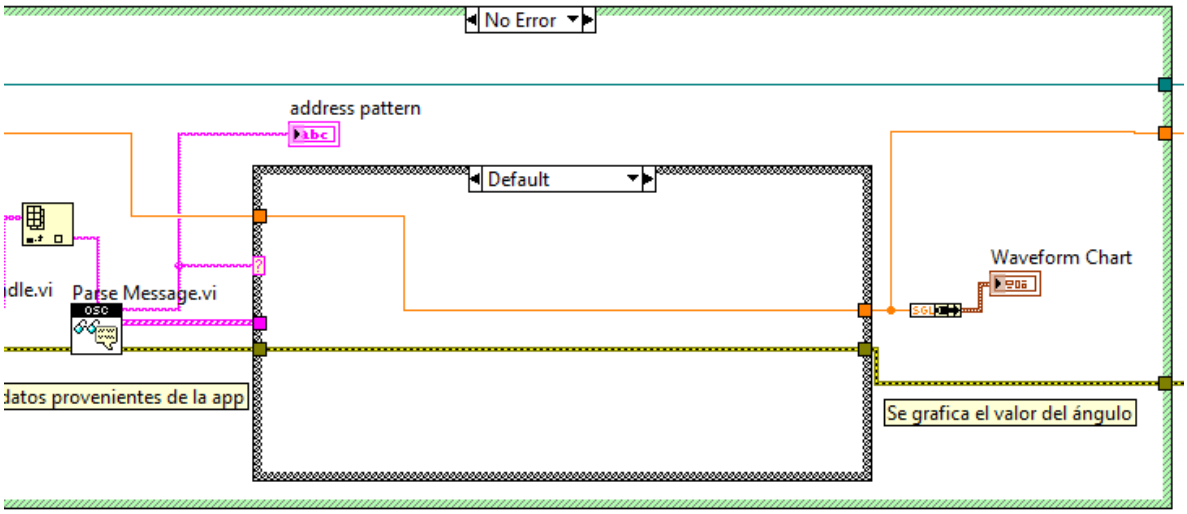


Figura 8-11. Caso para un dato diferente al ángulo de yaw del giroscopio

El ángulo de guiñada (*yaw)* corresponde al giro con respecto al eje Z en el iPod (ver Figura 8-12).

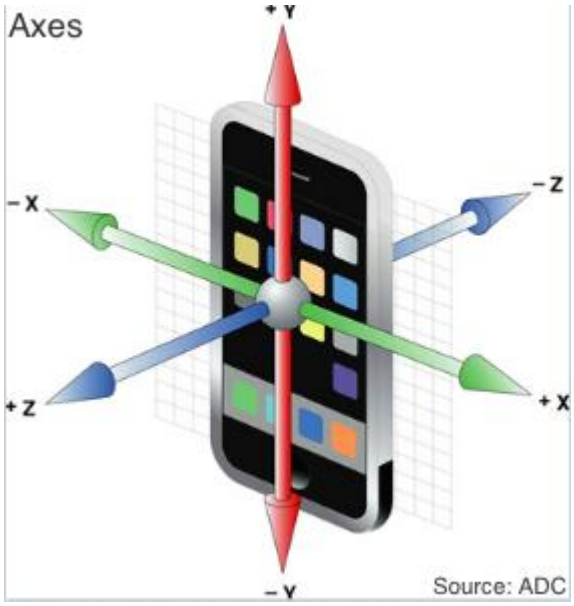


Figura 8-12. Ejes del giroscopio de un dispositivo iOS

Entonces, si el iPod se coloca horizontalmente sobre el robot DaNI el ángulo *yaw* corresponderá al mismo angulo de orientación del robot (ver Figura 8-13).

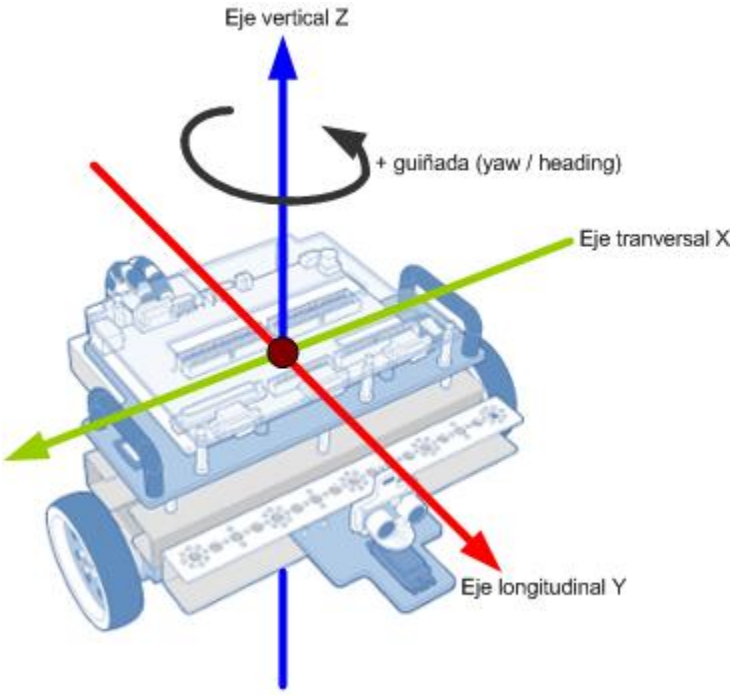


Figura 8-13. Ejes del sistema de referencia local del robot DaNI

La app GyrOSC permite establecer la posición de referencia al presionar el botón mostrado de la Figura 8-14, de esta manera se puede establecer la misma orientación de 0° para el robot y el iPod.

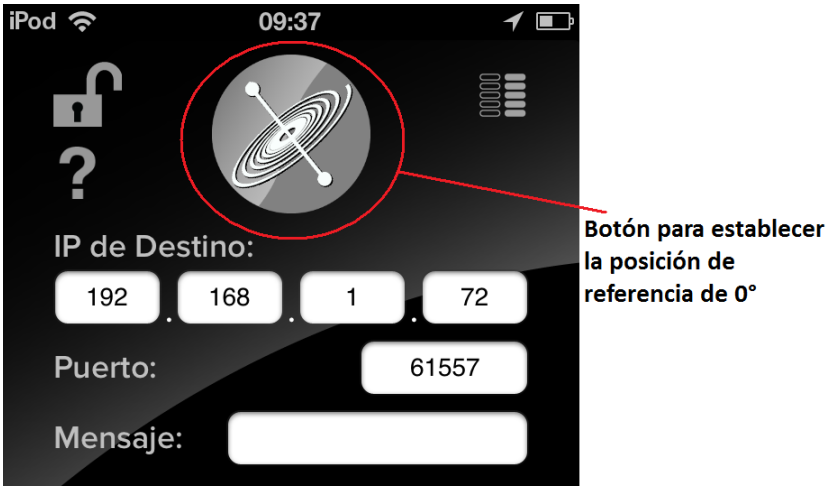


Figura 8-14. Botón para establecer la orientación de referencia en la app GyrOSC

Los datos adquiridos del giroscopio deben ser enviados al robot DaNI, debido a que no hay manera de que se envíen directamente desde el dispositivo iOS al robot (a menos que se incorpore un router inalambrico al robot DaNI), el VI que adquiere los datos del giroscopio se ejecuta en la computadora huesped (host), mientras que el VI que incorpora los valores del giroscopio del iPod en el esquema de navegación autonoma se ejecuta en el robot DaNI, adicionalmente es indispensable que el robot DaNI este conectado a la computadora huesped en todo momento para conocer los valores del giroscopio.

En la Figura 8-15 se muestra el VI que incorpora los datos del giroscopio del iPod al modelo odométrico del robot DaNI, la parte de estimación del ángulo de orientación del robot es sustituida por la lectura de la variable compartida *Yaw(rad)*, de esta manera se incorpora el ángulo de orientación medido con el IPod al esquema de navegación autonoma del robot DaNI presentado en este trabajo.

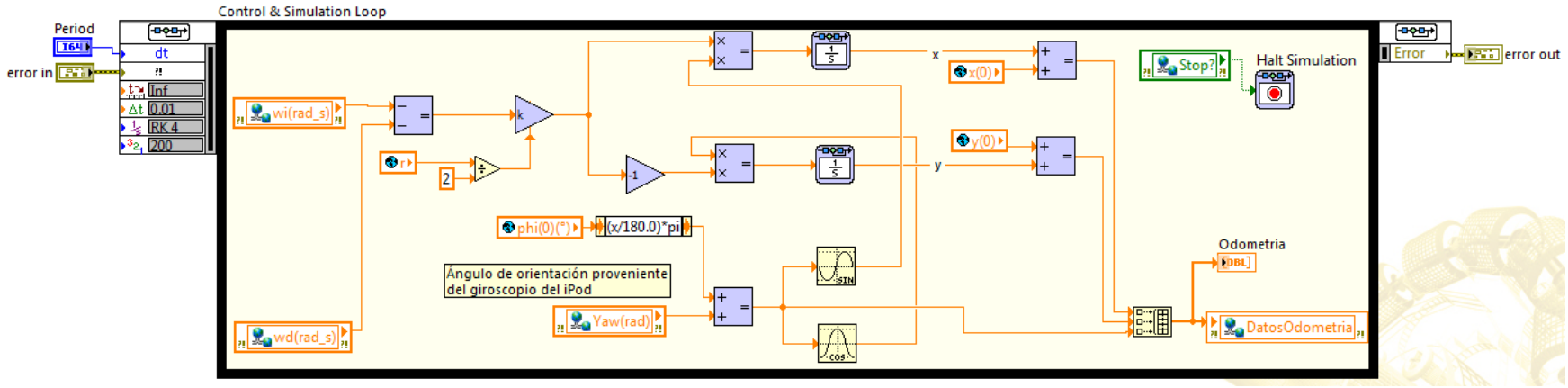


Figura 8-15. Código del modelo odométrico con el dato del ángulo de yaw del giroscopio

En la Figura 8-16 se muestra el proyecto donde el VI de la adquisición de datos del giroscopio se encuentra bajo la computadora huesped (My Computer), y el VI que lee los datos y los incorpora al robot DaNI se encuentra bajo la tarjeta sbRIO-9632.

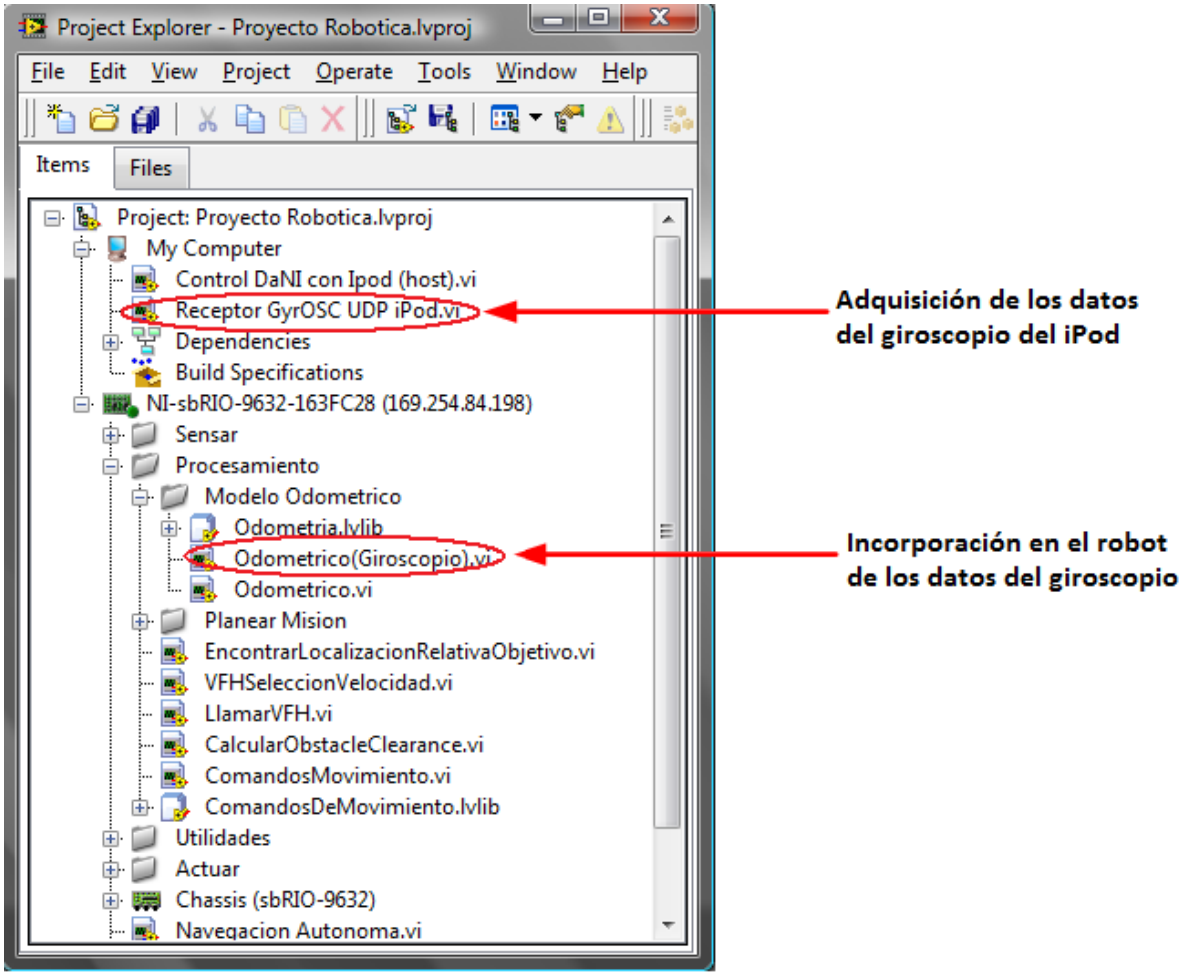


Figura 8-16. Ubicación en el proyecto de los VIs de adquisición de datos del giroscopio

Apéndice C

Control manual del robot DaNI con un dispositivo iOS

Derivado de la medición del ángulo de orientación del robot DaNI con un iPod, se diseñó un control manual del robot utilizando el acelerómetro del iPod. Esto permite controlar al robot remotamente a través de una red (WiFi en este caso).

El primer paso es instalar el toolkit *NI Open Sound Control* como se explica en el apéndice B. Posteriormente es necesario elaborar el VI de adquisición de datos del giroscopio, el cual es prácticamente el mismo que el mostrado en la del apéndice B, con la única diferencia de que además del ángulo de *guiñada (yaw),* se adquiere el ángulo de balanceo (*roll)* que corresponde al giro alrededor del eje Y en la Figura 8-12. Para adquirir estos ángulos se creó el VI mostrado en la Figura 9-2.

Una vez que se tiene el VI para la adquisición de dato del giroscopio, es necesario elaborar el programa que accionará y controlará los motores del robot DaNI con los valores del giroscopio, este programa se muestra en la Figura 9-1.

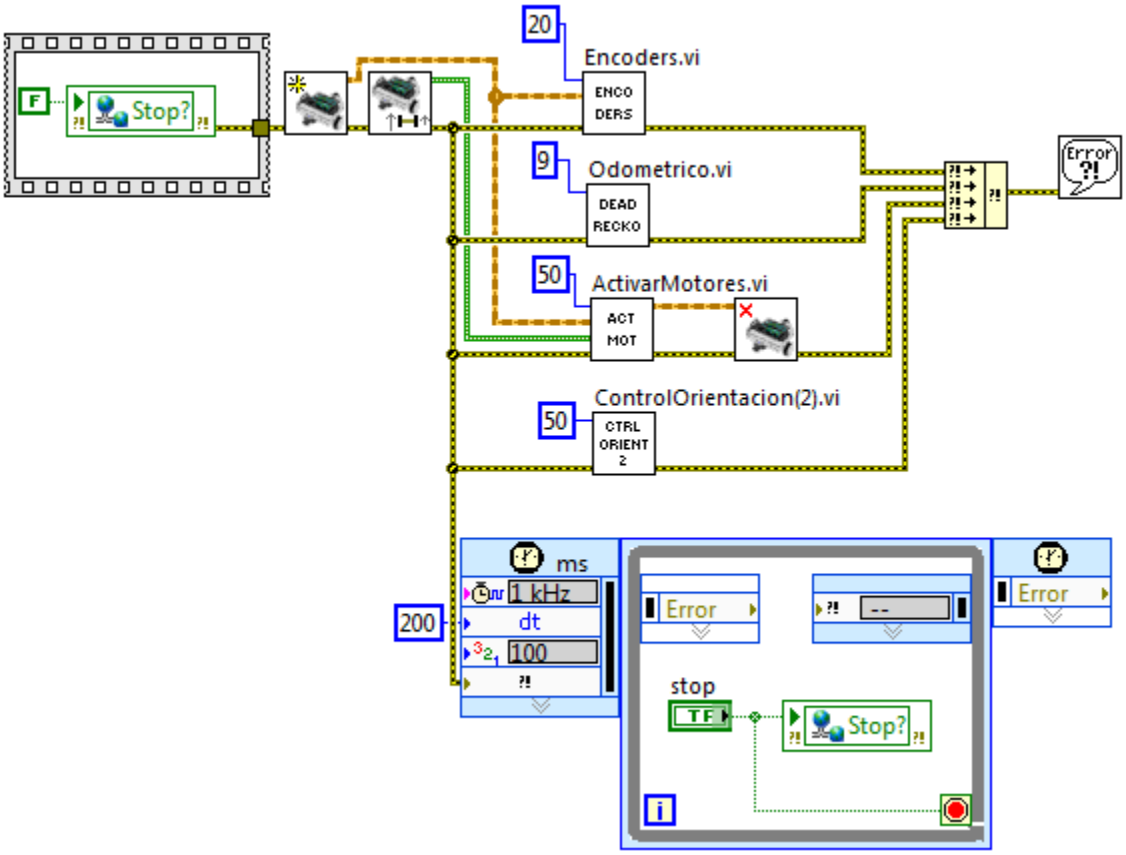


Figura 9-1. Programa para el control y accionamiento de los motores del robot DaNI

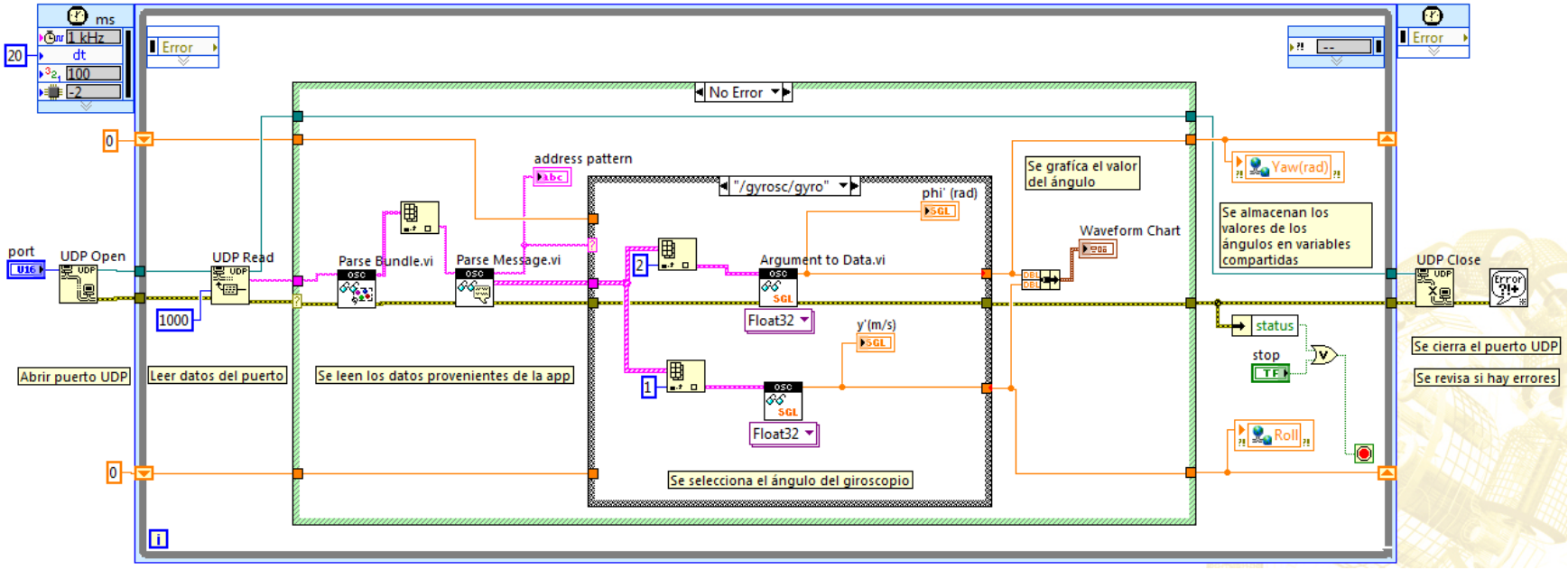


Figura 9-2. Adquisición de los ángulos de guiñada (yaw) y de balanceo (roll) del giroscopio del iPod

Los VIs: *Encoders, Odométrico* y *ActivarMotores* son los mismos utilizados en la navegación autónoma del robot DaNI (ver Figura 5-11, Figura 5-14 y Figura 5-55). El VI *PI_D Orientacion* se muestra en la Figura 9-3, en la cual el subVI *PI_D* también es el utilizado anteriormente en la navegación autónoma del robot DaNI.

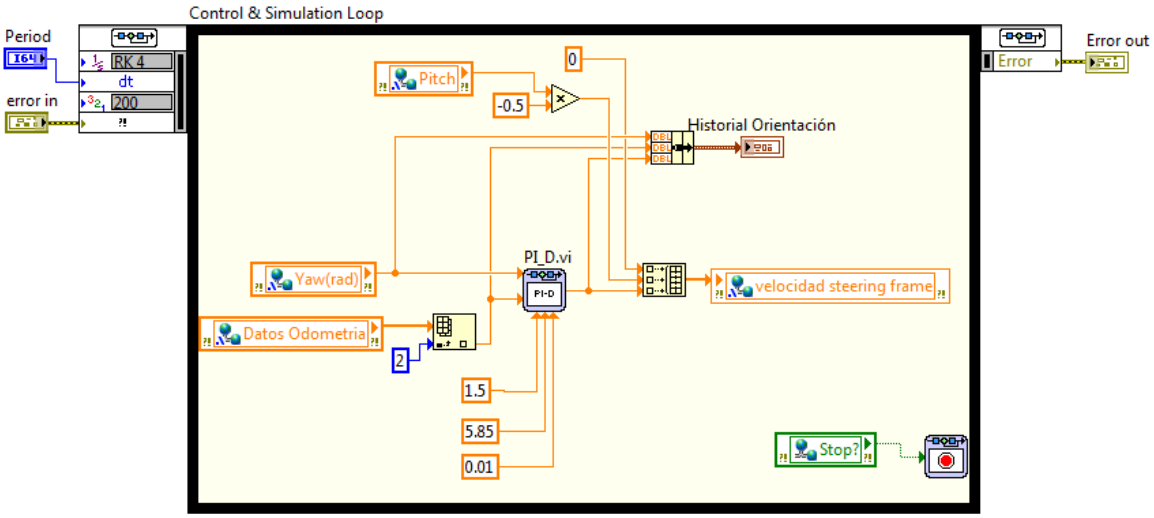


Figura 9-3. Control de orientación para el modo manual con iPod del robot DaNI

Los VIs de lectura de los encoders, del modelo odométrico y del control de orientación se utilizan para controlar los giros con los datos del giroscopio, y el VI *Activar Motores* acciona los motores.

El VI de adquisición de datos del giroscopio se ejecuta en la computadora huésped (host), mientras que los VIs de accionamiento y control de los motores se ejecutan en el robot DaNI (target), como se observa en el proyecto mostrado en la Figura 9-4.

El robot DaNI debe estar conectado en todo momento a la computadora huésped para conocer el valor del giroscopio, ya que no hay manera mandar los datos del giroscopio directamente desde el iPod hasta el robot DaNI, a menos que se tenga un router inalámbrico conectado directamente al robot DaNI.

El ángulo de guiñada (yaw) del giroscopio del iPod se utiliza para controlar la velocidad angular del robot (giro a la izquierda o derecha), y el ángulo de cabeceo (pitch) se utiliza para controlar la velocidad lineal del robot (avance hacia enfrente o hacia atrás).

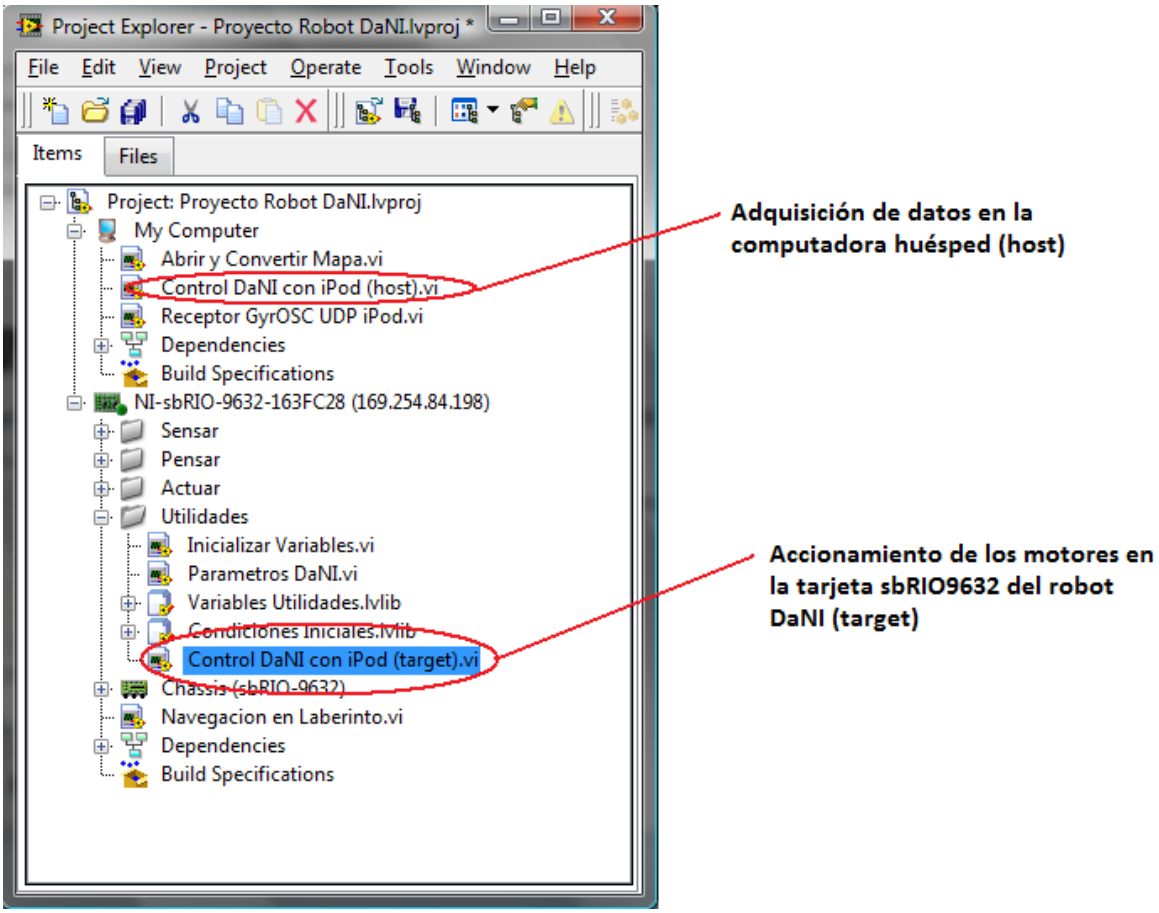


Figura 9-4. Ubicación en proyecto de los VIs para la adquisición de los datos del giroscopio y del accionamiento y control de los motores

Bibliografía y referencias

[1] R. Siegwart y N. Illah, *Introduction to Autonomous Mobile Robots*, Cambridge, Massachussets: MIT Press, 2004.

[2] A. Ollero, *ROBOTICA Manipuladores y robots móviles*, México, D.F.: Alfaomega, 2007.

[3] J. Borenstein y Y. Koren, «The Vector Field Histogram-Fast Obstacles Avoidance For Mobile Robots,» *IEEE Journal of Robotics and Automation Vol 7, No. 3,* pp. 278-288, 1991.

[4] K. Ogata, *Modern Control Engineering*, Third Edition, New Jersey, USA: Prentice Hall, 1997.

[5] P. Bolzern, R. Scattolini y N. Schiavoni, *Fundamentos de control automático*, Tercera edición, Madrid: McGraw Hill, 2009.

[6] J. Jones y A. Flynn, *Mobile Robots Inspiration to Implementation*, Wellesley, Massachusetts: A K Peters, 1993.

[7] J. Salido, *CIBERNÉTICA APLICADA Robots Educativos*, México DF: Alfaomega, 2010.

[8] J. Martinez, *Control de un robot móvil para el seguimiento de trayectorias*, México: UNAM FES Aragón, 2010, pp. 14-15.

[9] A. Alcaraz, *Desarrollo de algoritmos de control y movimiento de robots,* Cartagena: Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad de Cartagena, 2012.

[10] «mobile robotics experiments with DaNI - National Instruments,» [En línea]. Available: download.ni.com/pub/devzone/epd/mobile_robotics_experiments.pdf. [Último acceso: Diciembre 2012].

[11] «Getting Started with the LabVIEW Robotics Module - National Instruments,» [En línea]. Available: www.ni.com/pdf/manuals/375286c.pdf. [Último acceso: Diciembre 2012].

[12] «MINDS@UW,» [En línea]. Available: minds.wisconsin.edu/handle/63257. [Último acceso: Mayo 2013].

[13] «The National Australian University - Information Engineering - Serafina,» [En línea]. Available: http://serafina.anu.edu.au/. [Último acceso: Marzo 2014].

[14] «A* Pathfinding para principiantes,» [En línea]. Available: http://www.policyalmanac.org/games/articulo1.htm. [Último acceso: Diciembre 2013].

[15] «NI Single-Board RIO Embedded Control,» [En línea]. Available: http://www.ni.com/pdf/products/us/cat_sbRIO_96xx.pdf. [Último acceso: Noviembre 2013].

[16] «Sabertooth dual 10A motor driver for R/C,» [En línea]. Available: http://www.dimensionengineering.com/datasheets/Sabertooth2x10RC.pdf. [Último acceso: Noviembre 2013].

[17] «TETRIX DC Motor Specs,» [En línea]. Available: http://www.patronumbots.com/uploads/1/3/1/6/13161577/tetrix_dc_motor_specs.pdf. [Último acceso: Noviembre 2013].

[18] «Uranus an omni-directional mobile robot,» [En línea]. Available: http://www.cs.cmu.edu/~gwp/robots/Uranus.html. [Último acceso: Marzo 2014].

[19] «NI LabVIEW Robotics Starter Kit - Data Sheet,» [En línea]. Available: http://sine.ni.com/ds/app/doc/p/id/ds-217/lang/es. [Último acceso: Enero 2013].

[20] «Community: An Introduction to A* Path Planning (using LabVIEW),» [En línea]. Available: https//decibel.ni.com/content/docs/DOC-8983. [Último acceso: Noviembre 2013].

[21] «PING))) Ultrasonic Distance Sensor - Parallax Inc.,» [En línea]. Available: http://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf. [Último acceso: Marzo 2014].

[22] «RIObotics,» [En línea]. Available: http://riobotics.blogspot.mx/2011/03/using-labview-to-acquire-iphone.html. [Último acceso: Noviembre 2013].