UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

PROTOTIPO DE ROBOT MÓVIL PARA FINES DE EXPLORACIÓN TERRESTRE.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Ingeniero Eléctrico Electrónico

PRESENTA:

Héctor Edoardo Morales Aguilar

DIRECTORA DE TESIS

Ing. Gloria Mata Hernández

A Dios,
por conducirme en este camino y las enseñanzas obtenidas durante el viaje.
A mis padres,
ya que este trabajo es fruto de su apoyo y enseñanzas.
A mi hermano,
que espero y este trabajo te inspire a hacer uno mil veces mejor.

Agradecimientos

A la Universidad Nacional Autónoma de México, mi *Alma Mater*; por las experiencias y enseñanzas ofrecidas por esta.

A la Facultad de Ingeniería, que día a día prepara ingenieros para enfrentar los desafíos y problemas de la sociedad actual y considero como mi segunda casa.

A la Ing. Gloria Mata Hernández, por la confianza otorgada para elaborar este proyecto, por las oportunidades ofrecidas de participar como ponente en un congreso y dar algunas clases. Muchas gracias profesora.

A mi Padre, por proponerme siempre retos y estar disponible en los momentos que en los que necesité.

A mi Madre, por ser la persona en la que siempre puedo confiar, que me inspira a superar mis límites y me motiva a dar lo mejor de mi.

A mi Tía "Marcia", que siempre me cuidó como uno de sus hijos y ha estado presente en los momentos importantes.

A mi hermano, que es la principal persona que me motiva a ser mejor porque se que el puede ser mejor que yo.

A Joana Gabriela, mi amor: gracias por ser la persona que soporta mis malos momentos y a pesar de eso, sigues a mi lado, por ser mi editora en este trabajo y ser quien aclara mi mente en los momentos en los que me siento indeciso.

A mis maestros, aquellos que me motivaron (en especial a la Miss Lupita y a la profesora Tlanesis) y a los que no en elegir este camino, todo se los debo a ustedes.

A mis amigos, que sin ellos este camino no hubiera sido el mismo, a los de Puebla, a los Scouts, a los de la Universidad, a los "*Controlerros*".

Gracias totales.

"Si el conocimiento puede crear problemas, no es a través de la ignorancia con lo que podremos resolverlos."

Isaac Asimov

Índice general

Índice de figuras

Indice de tablas

Justificación

La necesidad del ser humano por explorar lugares o zonas en las cuales el acceso es limitado, riesgoso o imposible, lo ha llevado a idear nuevas formas de superar las barreras físicas impuestas por la naturaleza. Las investigaciones y desarrollos realizados en este campo han resultado en el nacimiento de la robótica, con la cual se han logrado llevar a cabo investigaciones en lugares en los que era inimaginable tener acceso debido a las restricciones que estos ambientes imponen, como es el caso en zonas marinas en donde, debido a la presión submarina, es imposible la presencia de seres humanos; en regiones con altos índices de contaminación en los que sería de alto riesgo la exposición para personas o bien en zonas extraterrestres. Es por ello que existen robots diseñados para llevar a cabo tareas de recolección de datos de variables ambientales y del propio entorno, para ser analizados, procesados y presentados en diferentes formatos e imágenes de la zona estudiada.
Teniendo en consideración las tareas necesarias para las cuales esta clase de robots son empleados, se pretende desarrollar un prototipo base de un robot móvil teleoperado, el cual sea capaz de adquirir datos acerca del entorno bajo estudio y presentar la información en diferentes despliegues e imágenes para la mejor visualización e interpretación del entorno. Este prototipo tendrá como objetivo final el poder servir de estudio, experimentación, desarrollo e investigación para diversas asignaturas de la carrera de Ingeniería Eléctrica Electrónica, así como sentar bases para contar con desarrollos tecnológicos nacionales.

Objetivo

Desarrollar el prototipo de un robot móvil con configuración diferencial para la exploración terrestre, que sea capaz de emitir video y audio del lugar a explorar, así como distintas características de ese entorno. Este robot será capaz de distinguir su posición usando un marco de referencia conocido y sus coordenadas terrestres empleando el Sistema de Posicionamiento Global (GPS). De la misma manera se contará con herramientas para el procesamiento del video enviado por el robot.

Prólogo

Para la realización del proyecto, se dividió en tres etapas. Las dos primeras fueron planteadas para su ejecución durante esta tesis y una tercera proyectada para un trabajo futuro. La primera de estas consistió en el conocimiento del robot y la implementación de un control para la teleoperación, la segunda en la incorporación de sensores y la última en el robustecimiento del sistema (Véase capítulo 5).

La organización y descripción de cada uno de lo capítulos contenidos en esta tesis es la siguiente:

En el Capítulo 1 se hace una descripción del estado actual de la robótica, en donde se mencionan las distintas clasificaciones de robots que existen en la actualidad, así como las configuraciones empleadas en los robots móviles. Por último se mencionan algunos ejemplos de los robots móviles terrestres comerciales dedicados a la exploración.

El Capítulo 2 describe el robot que se empleó en el proyecto, los sensores que se agregaron para poder obtener datos durante la exploración, la manera en que se realizó la red en la que se comunicarán los distintos dispositivos y las herramientas de software empleadas en el proyecto.

En el Capítulo 3 se menciona cuales fueron los pasos que se siguieron en la realización del proyecto, las características y la implementación de los sensores empleados y las distintas aplicaciones que fueron desarrolladas para el funcionamiento del robot. Mientras que en el Capítulo 4 se explica cómo se probó el funcionamiento de cada una de las actividades que se programaron para el robot y se muestra su operación conjunta en el robot.

El Capítulo 5 se describe cuales fueron las conclusiones a las que se llegaron después de la realización del proyecto, así como el posible trabajo a futuro a desarrollarse en esta línea de estudio.

Por último, en la sección de Anexos se incluye información adicional de interés general, la cual se obtuvo durante la realización de esta tesis. Se presenta una ley de control diseñada para el seguimiento de trayectorias, la forma en que se implementó la red informática diseñada en este proyecto, las distintas posibilidades de configuración del FPGA con el que cuenta el robot, ejemplos de las oraciones que emplea la norma NMEA 0183, el código con el que se desarrolló la aplicación para el procesamiento de video, así como pruebas individuales para la verificación de cada uno de los elementos que integran al robot en conjunto.

1

Introducción a la Robótica

Desde tiempos inmemoriales el ser humano siempre ha tenido la idea de dar vida a objetos inanimados con el objetivo de que estos reemplacen a sus creadores en tareas que resultan repetitivas y/o peligrosas para nosotros, por lo cual la ciencia ha tratado de lograr este cometido mediante el desarrollo del área de investigación conocida como *'Robótica'*.

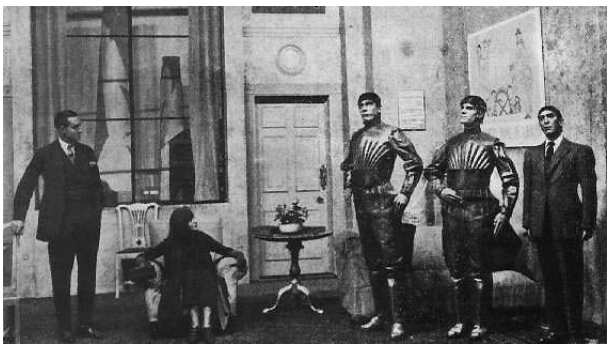


Figura 1.1 [**Fotografía de la obra *Rossum's Universal Robots***] Recuperado de: *http://www.umich.edu/∼eng415 /literature/pontee/RUR/RURsmry.html*

Este concepto fue retomado en tiempos modernos a partir de que el dramaturgo sueco Karel Capek en el año 1920, presentó la obra *Rossum's Universal Robots (R.U.R.)* (Figura 1.1). En esa ocasión, él empleó el término *robot*, el cual se deriva de la palabra de origen checo *robota*, que significa *trabajo, labor* o *servidumbre*), para denominar a unos seres artificiales que tenían como función ser los sirvientes de su creador y fue a partir de ese entonces que este concepto retomo fuerza. Otro gran aporte que se logró a través de la literatura fue el de la palabra *'Robótica'*, ya que el escritor de ciencia ficción Isaac Asimov acuñó esta palabra para definir a la ciencia que estudia a los robots, así como también fue el quien escribió *Las Leyes de la Robótica*. Las cuales ejemplifican interacción entre robots y humanos [1].

Figura 1.2 [**Robot teleoperado para desactivación de bombas**] Recuperado de: *http://www.monografias.com /trabajos93/inteligencia-artificial-aplicada-robotica/image006.jpg*

Aunque el desarrollo de los autómatas tiene su origen en tiempos de la cultura griega, el desarrollo de estos sistemas empezó a ser mas perceptible con el inicio del siglo XX. Durante esta época se empezaron a emplear distintas tecnologías para el control en tiempo real y acciones de sensado primarias; de estas innovaciones nacen los robots teleoperados (Figura 1.2), los cuales son el principio de la robótica. Estos son controlados remotamente de forma en la que el operador no

se encuentre en contacto directo con el robot; originalmente tenían la función de emplearse para el manejo de materiales radioactivos.

1.1 Clasificación de robots

Existen diversas formas de llevar a cabo la clasificación de los robots y cada una se lleva a cabo en función de algún parámetro, ya sea nivel de tecnología, uso, movilidad, autonomía, etc. En este trabajo se mencionan las clasificaciones en base a movilidad y autonomía.

1.1.1 Clasificación de robots según autonomía

La siguiente forma de clasificación (Figura 1.3) , distingue a los robots en cuanto a su autonomía. De acuerdo a esto se dividen en tres grupos, los cuales son: Teleoperados, de Funcionamiento Repetitivo y Autónomos o Inteligentes.

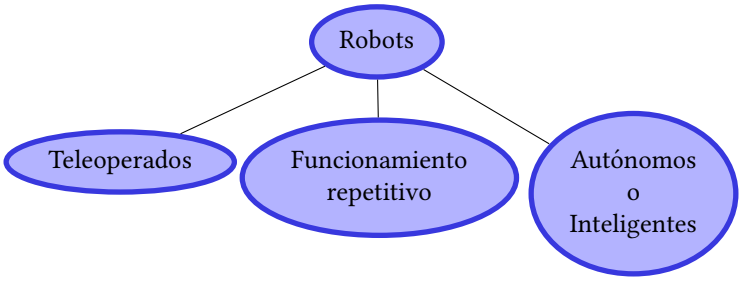


Figura 1.3 Clasificación de acuerdo a su autonomía

1.1.1.1 Robots teleoperados

En el caso de los *robots teleoperados*, las tareas de percepción del entorno, planificación y manipulación compleja son realizadas por seres humanos. Visto de otra forma, el operador es el encargado de cerrar en tiempo real el bucle de control. Estos robots son empleados para trabajos en localizaciones remotas, en tareas difíciles de automatizar y en entornos no estructurados. (Figura 1.4)

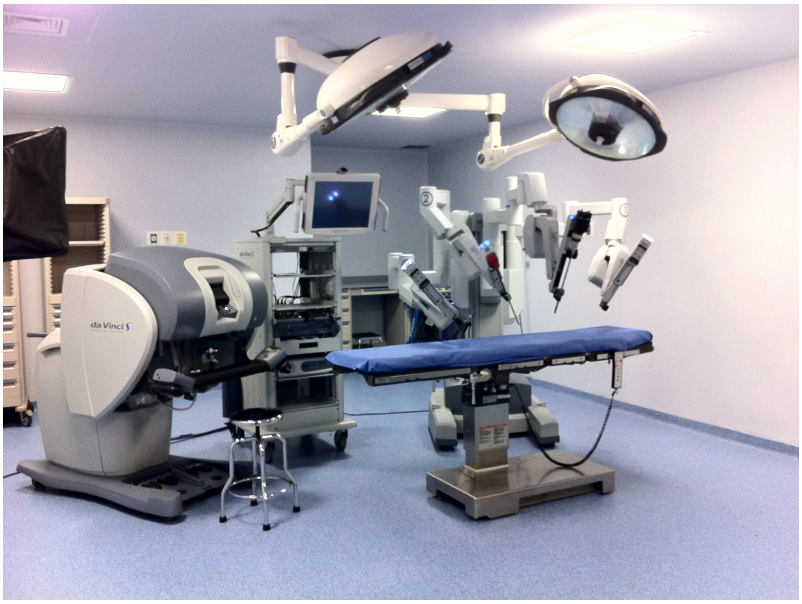


Figura 1.4 [**Robot teleoperado Da Vinci**] Recuperado de: *http://www.urologiaendourologia.com/imagenes/da %20Vinci %201.JPG*

Dado que la forma en la que se manipula el robot en estudio es la teleoperación, cabe profundizar en este tema. De acuerdo al nivel y el estilo de la arquitectura de control, los robots teleoperados se pueden conjuntar en los siguientes grupos principales (Figura 1.5):

- Control Directo: Esta arquitectura implica ninguna inteligencia o autonomía en el sistema, esto quiere decir que el movimiento del esclavo es directamente controlado por el usuario a través de la interfaz maestra.
- Control Compartido: Esta arquitectura funciona de forma que la ejecución de una tarea es compartida entre el control directo (maestro) y el sensado local (esclavo) o en su defecto que la operación del maestro se vea influida por una realidad virtual u otras ayudas automáticas.
- Control de Supervisión: Esta arquitectura comprende el nivel mas alto de inteligencia, ya que el operador introduce sentencias de alto nivel y el robot realiza su ejecución de forma autónoma.

De acuerdo con [2], en la práctica las estructuras de control empleadas tienen partes de los elementos antes mencionados.

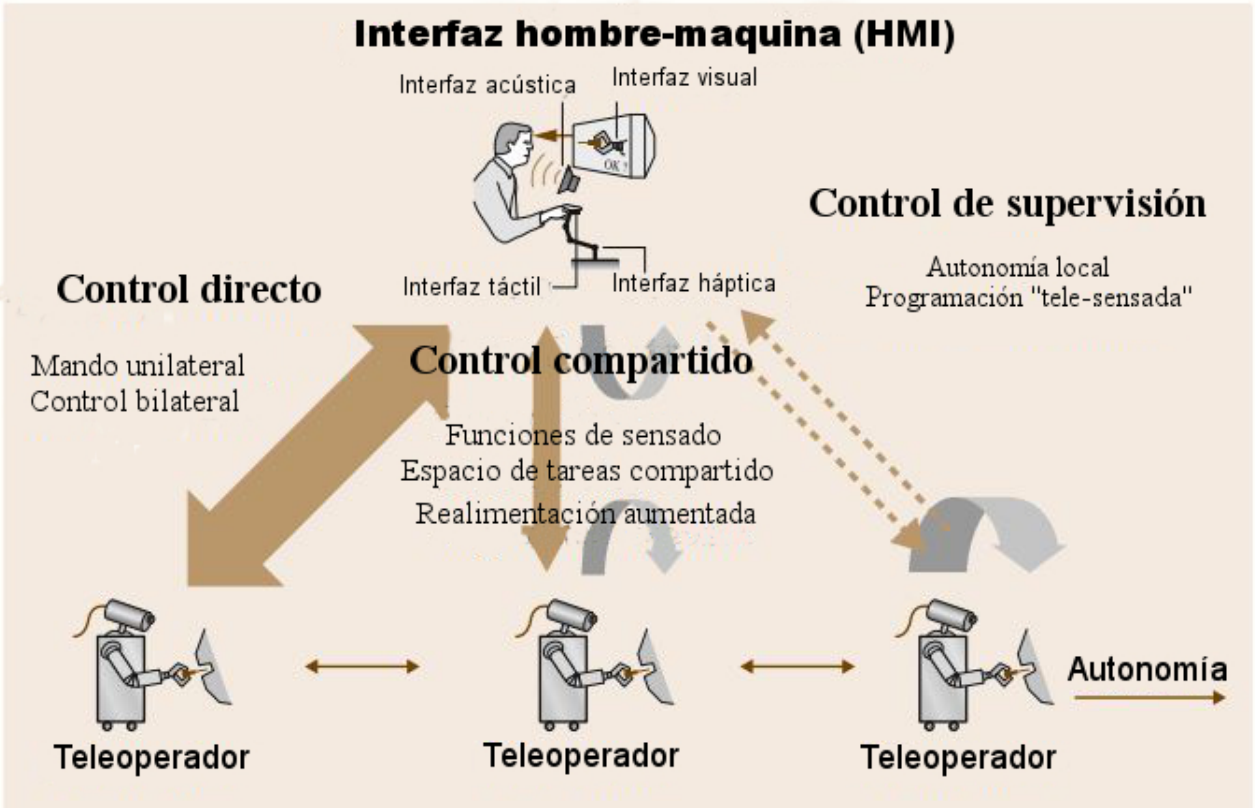


Figura 1.5 Desconocido. [**Diferentes conceptos para el control de arquitecturas en robots teleoperados**] Recuperado de: [2]

1.1.1.2 Robots de funcionamiento repetitivo

Los *robots de funcionamiento repetitivo*, son los que comúnmente se encuentran en las líneas de producción; estos trabajan normalmente en tareas predecibles e invariantes, con una limitada percepción de su entorno. Estos robots tienen una alta precisión y son relativamente rápidos. (Figura 1.7)

1.1.1.3 Robots autónomos

Por último, los *robots autónomos* son los más evolucionados. Tomando en cuenta el procesamiento de la información, estos son capaces de percibir su entorno, planificar y actuar sin la necesidad de que un ser humano intervenga. Pueden trabajar en entornos dinámicos y poco estructurados, realizando acciones en respuesta a los cambios de dicho entorno.

1.1.2 Clasificación de robots según movilidad

Esta clasificación se realiza de acuerdo al grado de movilidad del robot, por lo cual se encuentran divididos en robots fijos y robots móviles.

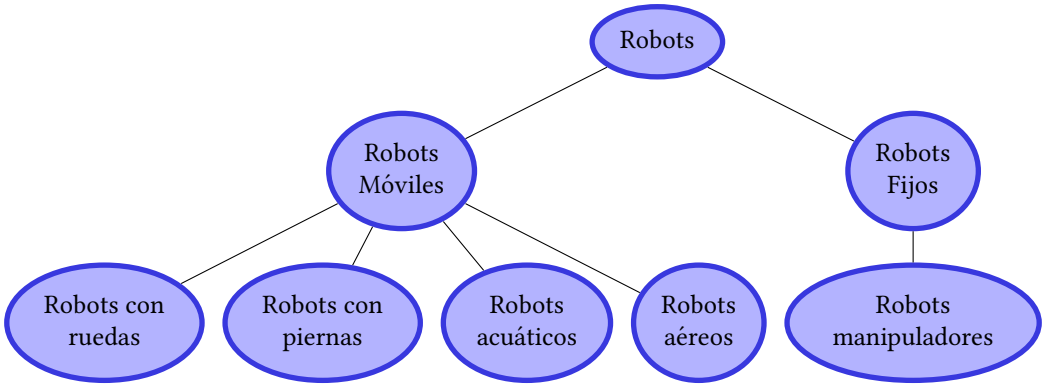


Figura 1.6 Clasificación de robots según su movilidad

1.1.2.1 Robots fijos

Por un lado se presentan los robots fijos (Figura 1.7); en esta categoría se encuentran los robots industriales o manipuladores, que son esencialmente brazos articulados. De forma mas precisa, un manipulador industrial convencional es una cadena cinemática abierta formada por un conjunto de eslabones interrelacionados mediante articulaciones o pares cinemáticos [1].
Estos robots son empleados en la mayoría de los casos en líneas de producción; se consideran fijos debido a que su base se ubica sujeta a un punto, por lo cual su movilidad se encuentra limitada a una *'Área de trabajo'*.

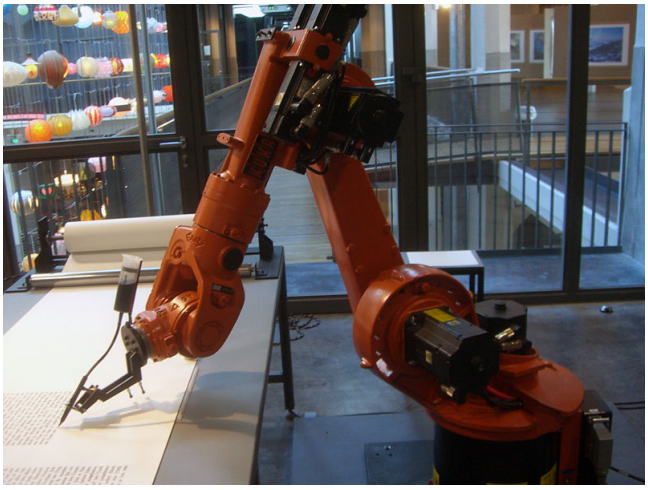


Figura 1.7 Tobias. Mirko [**Robot industrial KUKA**] Recuperado de: *https://www.flickr.com/photos/gastev/2174505811/*

1.1.2.2 Robots móviles

Por otro lado, se encuentran los robots móviles, los cuales tiene las habilidad de maniobrar libremente para el ambiente que fueron diseñados. Nacen a partir de la necesidad de añadir autonomía en los robots, limitando la intervención del ser humano al máximo.

Existen distintos robots móviles dependiendo del ambiente en que se desplazarán, así como el medio de locomoción que utilizan. En la Figura 1.6 se muestran algunos ejemplos de los robots móviles que existen. En el caso de este trabajo haremos hincapié en los robots móviles con ruedas, por lo que se mencionarán algunas de sus configuraciones.

Figura 1.8 [**Robot móvil TALON**] Recuperado de: *http://www.army-technology.com/projects/talon-tracked-military-robot/*

1.2 Tipos de configuraciones para robots con ruedas

Para el diseño de robots móviles con ruedas, se encuentran distintos tipos de configuraciones y cada uno tiene sus ventajas y desventajas propias; a continuación se enumeran las configuraciones mas populares, así como los distintos tipos de ruedas que existen.

1.2.1 Restricciones holonómicas

Antes de mencionar las distintas configuraciones existentes, es necesario explicar la definición de una palabra que es empleada comúnmente en la robótica móvil, este termino es *holónomo*. Este tiene amplios usos en varias áreas matemáticas como son las ecuaciones diferenciales, funciones y expresiones de restricción, en el caso de los robots móviles se refiere específicamente a las restricciones cinemáticas del chasís con el que cuenta el robot. Por lo que concretamente un *robot holonómico* es aquel que puede desplazarse en cualquier dirección, mientras que un *robot no holonómico* solo le es posible desplazarse paralelamente al eje de dirección del chasís.

Matemáticamente, una *restricción holónoma* puede ser expresada como una función explícita solamente de las variables de posición. Es decir, tienen la forma:

$$G_k(p,t) = 0; \qquad k = 1, \ldots, s$$

donde: p=posición, t=tiempo

Y una *restricción no holonómica* requiere una relación diferencial, como la derivación de las variables de posición. Para que una restricción sea no holonómica se exige además que no sea integrable, es decir, que no se deduzca por derivación total con respecto al tiempo de una holonómica. Debido a este último punto de vista, los sistemas no holonómicos son a menudo llamados *Sistemas no integrables*. En sistemas con restricciones no holonómicas, el número de coordenadas que describe su configuración siempre es más grande que el número de grados de libertad que poseen.

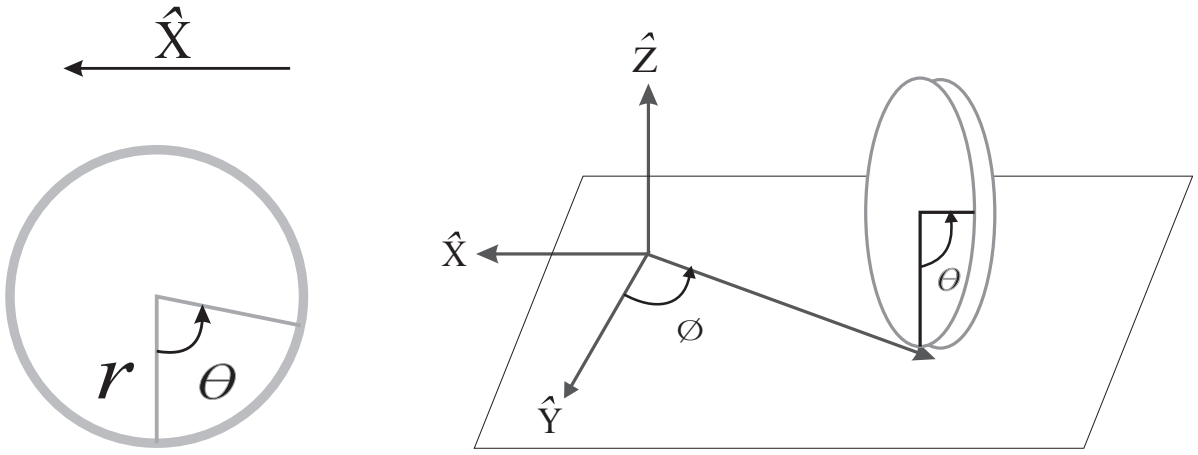
Un ejemplo para distinguir las diferencias entre las restricciones holonómicas y las no holonómicas es el estudio del movimiento de una rueda en una dimensión y en el plano.
Si se toma en cuenta el movimiento de una rueda de radio r en una dimensión, como se ilustra en la Figura 1.9a, la variable del actuador podría ser el giro θ y la variable en el espacio cartesiano la x que indica la distancia recorrida. Las dos variables obedecen a la condición de rodadura

$$x' = c\theta' \tag{1.1}$$

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

$$x - c\theta = constante \tag{1.2}$$

Por consiguiente, en este caso no existen restricciones no holonómicas. Cabe hacer hincapié en que sólo es necesaria una coordenada (x o θ) para determinar la posición de la rueda y de la misma forma existe también un solo grado de libertad.



(a) En una dimensión

(b) En el plano

Figura 1.9 Movimiento de la rueda

Por el contrario, si se estudia el movimiento de la misma rueda en el plano (Figura 1.9), se encuentran las restricciones no holonómicas. Si se considera el movimiento de la rueda de tal forma que el diámetro correspondiente al punto de contacto con el suelo este siempre en posición vertical. De esta forma se pueden emplear cuatro coordenadas para especificar completamente la posición y orientación de la rueda; las coordenadas (x, y) del punto de contacto, el ángulo θ entre la vertical y un radio de referencia, el cual indica cuanto ha girado el disco y el ángulo de orientación ϕ de la rueda. La condición de rodamiento sin deslizamiento introduce dos restricciones, ya que el espacio que el punto de contacto recorre sobre el borde de la rueda es igual al que recorre en el plano. Debido a lo anterior, si se proyecta la velocidad del punto de contacto en el plano, paralela y perpendicularmente, se obtiene:

$$-x' \sin\phi + y' cos\phi = \theta' c \tag{1.3}$$

$$x' \cos\phi + y' \sin\phi = 0 \tag{1.4}$$

Las restricciones (1.3) y (1.4) no son integrables, lo cual se comprueba verificando que el sistema de ecuaciones no es de rango completo (teorema de Frobenius), por lo tanto no se pueden obtener relaciones funcionales entre las variables (x, y, θ, ϕ) a partir de las ecuaciones de restricción.
Visto de otra forma, asignando los valores $(x_0, y_0, \theta_0, \phi_0)$, haciendo rodar la rueda sin deslizar y girándola alrededor del eje vertical, es posible llegar a cualquier otra configuración $(x_f.y_f.\theta_f, \phi_f)$, lo cual implica que no existe una relación fundamental entre estos valores. Sin embargo, las direcciones de movimiento deben satisfacer las ecuaciones anteriores y, por consiguiente, el camino no puede ser cualquiera.

1.2.2 Tipos de ruedas

El tipo de rueda empleado en la construcción de robots móviles, está íntimamente relacionado con el tipo de configuración a emplear. A continuación se enumera de forma breve la descripción de estos tipos de ruedas.

a) **Rueda estándar**:(Figura1.10a) Cuenta con dos grados de libertad: la rotación alrededor del eje de la rueda y el punto de contacto.

b) **Rueda de castor**: (Figura1.10b) Cuenta con dos grados de libertad: rotación alrededor del eje de la rueda y alrededor de un punto articulado que lo direcciona con un offset.

c) **Rueda Sueca**: (Figura1.10c) Cuenta con tres grados de libertad, rotación alrededor del eje de la rueda, alrededor de los *rodamientos"* y alrededor del punto de contacto.

d) **Rueda de balón o esférica**: (Figura1.10d) Esta rueda es omnidireccional y su implementación mecánica es difícil de realizar.[3]

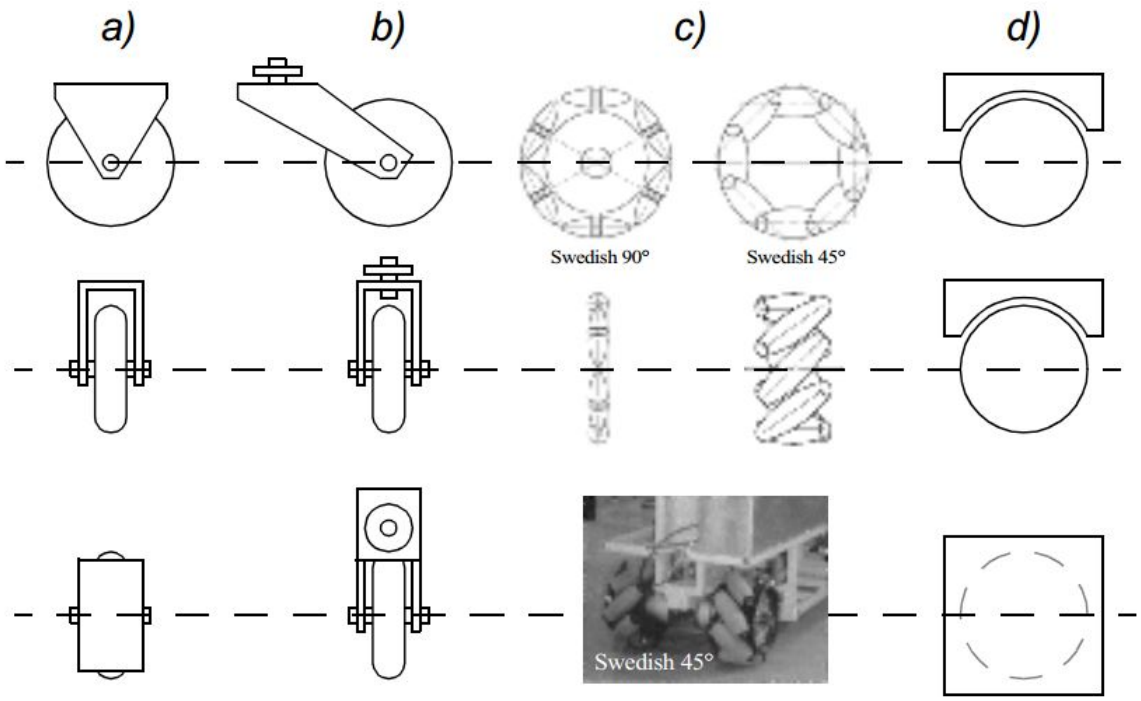


Figura 1.10 Desconocido. [**Distintos tipos de ruedas**] Recuperado de: [3]

1.2.3 Configuración Diferencial

Esta configuración (Figura 1.11) consiste en dos ruedas actuadas montadas en el lado izquierdo y derecho de la plataforma del robot. Las dos ruedas se conducen independientemente. Pueden existir una o dos ruedas sin actuar, las cuales dan estabilidad al vehículo. Es la configuración mecánica mas sencilla, debido a que no es necesaria la rotación de los ejes de dirección. Si las ruedas se mueven con la misma velocidad, estas se desplazan hacia adelante o hacia atrás; si una rueda se mueve mas rápido que la otra, el robot sigue una curva a lo largo de la trayectoria del arco del centro instantáneo de curvatura; si las ruedas giran ambas con la misma rapidez pero en distinto sentido, el robot gira alrededor del punto medio de las dos ruedas.

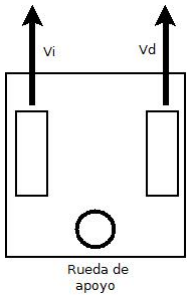


Figura 1.11 Configuración diferencial

El robot empleado para la realización de este proyecto tiene una configuración diferencial y como rueda de apoyo utiliza una rueda sueca con barriles de rodamiento a 90°

1.2.4

Configuración triciclo

Esta configuración (Figura 1.12) tiene una sola rueda, la cual es la encargada del desplazamiento y la dirección del robot. Para la estabilidad se emplean dos ruedas traseras sin actuar para poder tener siempre los tres puntos de contacto que son requeridos. La velocidad lineal y angular de la rueda están completamente desacopladas. Para un desplazamiento hacia adelante la rueda es posicionada en el punto medio y se desplaza a la velocidad deseada. Cuando la rueda forma un ángulo diferente de cero respecto al vehículo, este sigue una trayectoria curva, y en el caso particular de que la rueda forme un angulo de 90 grados, el robot girará siguiendo una trayectoria circular, el centro de ésta será el punto medio de las ruedas traseras y no el centro geométrico del robot.

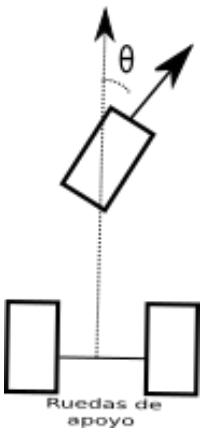


Figura 1.12 Configuración triciclo

1.2.5

Configuración Ackerman

La configuración de Ackerman (Figura 1.13) está diseñada para asegurar que el giro de las ruedas de todos los ejes tienen un punto de cruce común (centro instantáneo de rotación (ICR en inglés)), esto para evitar el deslizamiento de las ruedas. Esta configuración cuenta con las ruedas frontales actuadas y las ruedas traseras pasivas.

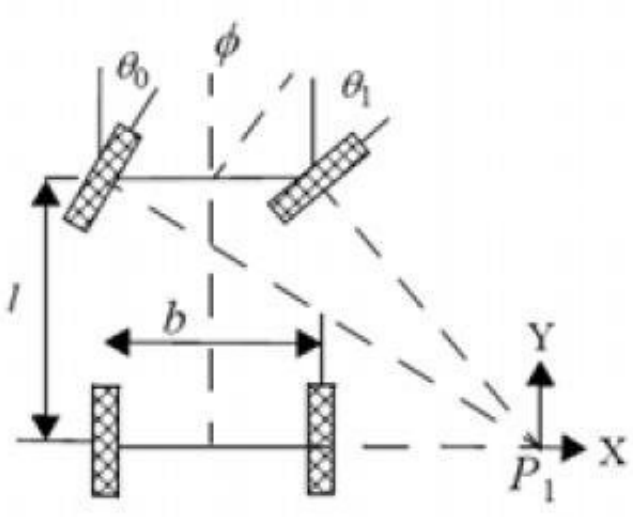


Figura 1.13 [**Configuración Ackerman**] Recuperado de: *http://www.xatakaciencia.com/robotica/robots-moviles-i*

1.2.6

Configuración asíncrona

Esta configuración (Figura 1.14) contiene tres o más ruedas que están unidas mecánicamente de tal forma que rotan en la misma dirección con la misma velocidad. Este desplazamiento síncrono mecánico puede ser realizado de distintas formas, por ejemplo mediante cadenas, cinturones o engranes. Esta configuración puede considerarse como cerca de ser holonómica, ya que cuenta con la posibilidad de desplazarse en cualquier dirección.

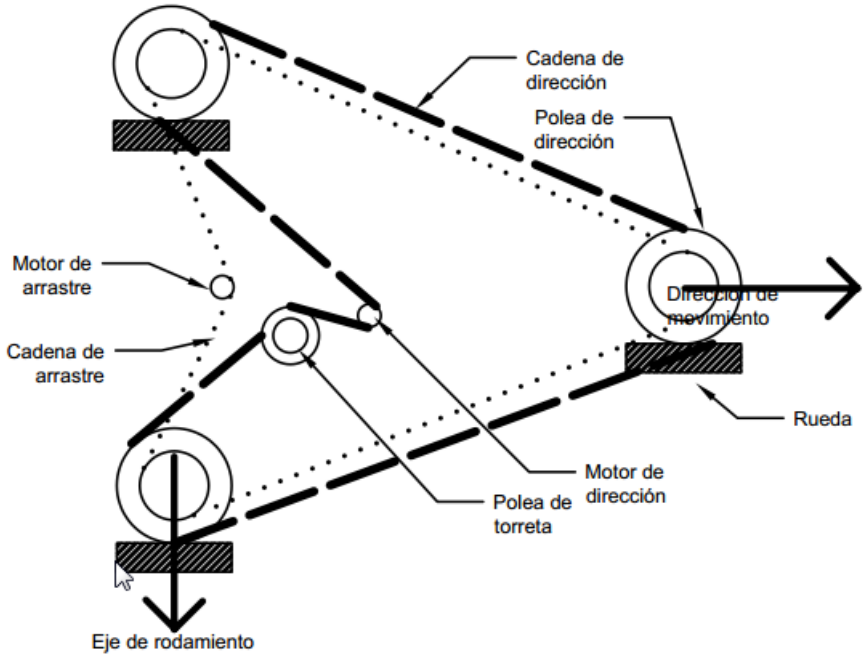


Figura 1.14 Configuración asíncrona

1.2.7

Configuración Omnidireccional

Esta configuración (Figura 1.15) puede ser lograda usando tres, cuatro o más ruedas omnidireccionales. Los robots móviles con tres ruedas usan ruedas suecas con barriles a $90°$, y los que contienen cuatro cuentan con dos tipos distintos de ruedas, dos son conocidas como *zurdas* y las otras como *diestras*. Las ruedas zurdas tienen sus barriles de rodamiento en un angulo $\alpha = 45°$ y de forma opuesta, las diestras con un angulo $\alpha = -45°$.

Figura 1.15 [Robot Carnegie Mellon Uranus, un robot omnidireccional] Recuperado de: *http://www.cs.cmu.edu/∼gwp/robots/Uranus.html*

1.3 Estado del arte

En la actualidad existen distintos tipos de robots que se especializan en tareas de exploración, y debido a la variedad de tareas para las que son programados, se presentarán cuáles son los robots que existen a nivel comercial, cabe hacer notar que los que se mencionan no son los únicos, pero si los que tienen funciones más globales para la exploración.

1.3.1 Robot SUMMIT

Este robot (Figura 1.16) es fabricado por la compañía Robotnik, el cual puede navegar de forma autónoma en el caso de tener un completo conocimiento del ambiente (posición de los obstáculos) y puede ser operado de forma remota a través de una conexión Wi-FI. Tiene implementado la configuración Ackerman y cuenta con dos motores sin escobillas para su control; ofrece la posibilidad de conexión mediante puertos USB, RS232, GPIO, RJ45 y sus aplicaciones son: Investigación, Seguridad, Monitorización Remota y Acceso a Zonas Peligrosas.

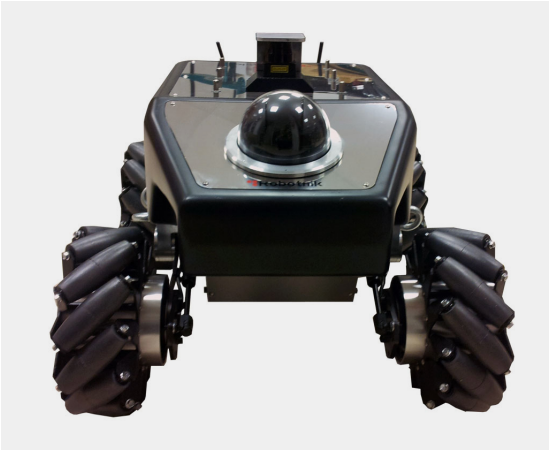


Figura 1.16 [**Robot SUMMIT**] Recuperado de: *http://www.robotnik.eu/mobile-robots/summit-xl-hl/*

1.3.2 Robot Throwbot-XT

El robot Throwbot-XT (Figura 1.17) es un robot teleoperado producido por ReconRobotics; es empleado para la transmisión de vídeo (en blanco y negro) y audio. Cuenta con una cámara infrarroja para situaciones en que la luz es insuficiente. Tiene una configuración Diferencial, es ligero, resistente a golpes y a prueba de agua.

Figura 1.17 [**Robot Throwbot-XT**] Recuperado de: *http://www.reconrobotics.com/products/Throwbot_XT_audio.cfm*

1.3.3

Robot FirstLook

Este robot (Figura 1.18a) es fabricado por la compañía iRobot; es un modelo pequeño con una configuración de tracción mediante orugas de deslizamiento, cuenta con cuatro cámaras situadas al frente, atrás, izquierda y derecha, es a prueba de agua y golpes. Es teleoperado a través de RF y existen variantes de este robot, las cuales tienen sensores dedicados a tareas especificas. En el caso de vigilancia, cuenta con una cámara actuada, o para tareas de manipulación de materiales peligrosos, cuenta con brazos robóticos.

(a) Robot (b) Mando del robot

Figura 1.18 [**Robot FirstLook**] Recuperado de: *http://www.army-technology.com/projects/irobot-110-firstlook-robot/*

1.3.4

Robot OSCAR

El robot OSCAR (Figura 1.19) es producido por la compañía COBHAM; es empleado para tareas de búsqueda en zonas de difícil acceso (derrumbes, cuevas, etc.) y alto riesgo. Es controlado de forma teleoperada y cuenta con una cámara infrarroja, una cámara de alta definición con módulo de iluminación, una cámara térmica de alto alcance y un láser para la medición de distancias. Tiene una configuración basada en orugas de deslizamiento y existe la posibilidad de agregar módulos de sensado.

Figura 1.19 [**Robot OSCAR**] Recuperado de: *http://www.cobham.com/about-cobham/mission-systems/unmanned-systems/products-and-services/remote-controlled-robotic-solutions/oscar-observation-surveillance-clarification-and-reconnaissance-robot.aspx*

Con los datos antes mencionados se realizó la siguiente tabla comparativa:

Robot	Configuración	Modo de comunicación	Integración de sensores	Transmisión de audio y video
SUMMIT	Ackerman	Wi-Fi	Puertos libres para distintos protocolos de comunicación	Solo video
THROWBOT-XT	Diferencial	RF	No disponible	Audio y video
FIRSTLOOK	Dos orugas de deslizamiento	RF	Existen variaciones con sensores integrados	Solo video
OSCAR	Cuatro orugas de deslizamiento	RF	Módulos de expansión provistos por el fabricante	Video en HD

Tabla 1.1 Tabla comparativa de las caracteristicas de los robots de exploración más comunes.

2

Estructura general del robot móvil para exploración

En este capítulo se muestran: el modelado matemático del robot, la descripción de las herramientas de software empleadas en la programación de las tareas, una descripción de los elementos que comprenden el prototipo, así como cual es la forma de comunicación empleada entre el robot y el operador.

2.1 Modelo matemático del robot móvil tipo diferencial

Para la realización de este modelo es necesario realizar ciertas consideraciones, como:

- El robot se mueve sobre una superficie plana.
- Las ruedas del robot tienen las mismas dimensiones.
- No existen deslizamientos.

La posición y orientación de un robot móvil en el plano pueden describirse por sus coordenadas (x, y) con respecto a un sistema de referencia fijo (X_G, Y_G) y el ángulo θ que el robot forma con respecto al eje X_G, como se muestra en la Figura 2.1.

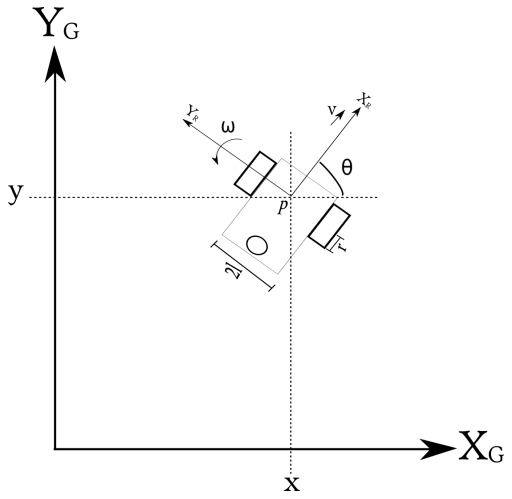


Figura 2.1 Localización del robot en el plano.

Teniendo esto en consideración, es posible encontrar las siguientes relaciones:

$$\dot{x} = v\cos(\theta) \tag{2.1a}$$

$$\dot{y} = v\sin(\theta) \tag{2.1b}$$

$$\dot{\theta} = \omega \tag{2.1c}$$

En donde x representa la posición a través del eje X_G, y representa la posición a lo largo del eje Y_G y θ representa la orientación del eje longitudinal del robot con respecto al eje X_G. De manera ideal la velocidad lineal v y la velocidad angular ω pueden ser consideradas como variables de control del sistema (2.1). Sin embargo, desde un punto de vista más realista, es necesario considerar que estas variables se encuentran en función de las velocidades angulares de las velocidades del robot. [5]
Si se consideran ω_d y ω_i, como las velocidades angulares de las ruedas derecha e izquierda respectivamente, entonces se puede mostrar de la siguiente manera que v y ω están relacionadas con ω_d y ω_i.

Si se sabe que ω_d y ω_i son las velocidades angulares de las ruedas, y tomamos en cuenta el radio de las ruedas r, la velocidad lineal de cada una queda expresada de la siguiente forma:

$$v_d = r\omega_d \tag{2.2a}$$

$$v_i = r\omega_i \tag{2.2b}$$

Para obtener la velocidad lineal, se analizará el movimiento de las ruedas independientemente. En un principio si la rueda derecha se desplaza mientras la izquierda se mantiene estática, se obtiene lo siguiente:

$$v_1 = \frac{v_d}{2} \tag{2.3}$$

Esto es debido a que el punto que considera la posición del robot, se encuentra a la mitad de la distancia de ambas ruedas (l). Por lo tanto, siguiendo el mismo principio; al aplicar una velocidad a la rueda izquierda mientras la derecha se mantiene estática, se obtiene esta nueva velocidad:

$$v_2 = \frac{v_i}{2} \tag{2.4}$$

Por lo tanto, para obtener la velocidad de las ruedas que se muevan juntas, se realiza una suma de velocidades y se obtiene la velocidad lineal del vehículo:

$$v = v_1 + v_2$$
$$v = \frac{v_d}{2} + \frac{v_i}{2}$$

$$v = \frac{v_d + v_i}{2} \tag{2.5}$$

$$v = \frac{r(\omega_d + \omega_i)}{2} \tag{2.6}$$

A continuación para obtener la velocidad angular se realiza la suma de las aportaciones individuales de cada una de las ruedas al vehículo. Si se considera que la rueda derecha gira con una velocidad angular ω_d y la rueda izquierda se mantiene estática, se formara una circunferencia de radio $2l$ y la velocidad angular del robot sera:

$$\dot{\theta_1} = \frac{r\omega_d}{2l} \tag{2.7}$$

Por lo tanto, para el análisis de la rueda izquierda se considera que se va a desplaza a una velocidad $-\omega_i$, el hecho de que esta velocidad sea negativa es para realizar el giro, ya que de otra forma el vehículo se desplazaría en línea recta, entonces la aportación de la velocidad izquierda es la siguiente:

$$\dot{\theta_2} = -\frac{r\omega_i}{2l} \tag{2.8}$$

De la misma forma que para la velocidad lineal, se suman las contribuciones de ambas velocidades para obtener la velocidad angular total:

$$\dot{\theta} = \dot{\theta_1} + \dot{\theta_2}$$

$$\dot{\theta} = \frac{r(\omega_d - \omega_i)}{2l} \tag{2.9}$$

Con lo anterior, se obtiene la relación buscada entre la representación del sistema y las variables de estado mediante la siguiente representación:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = T \begin{bmatrix} \omega_d \\ \omega_i \end{bmatrix}, \qquad T = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix}$$

Puesto que la transformación T es no singular, cualesquiera valores de las velocidades v y ω pueden ser obtenidos mediante la adecuada selección de ω_i y ω_d.

Por lo tanto, si sustituimos lo anterior en la Ec. (2.1), el sistema queda representado de la siguiente manera:

$$\dot{x} = \frac{r}{2}(\omega_d + \omega_i)\cos\theta \tag{2.10}$$
$$\dot{y} = \frac{r}{2}(\omega_d + \omega_i)\sin\theta$$
$$\dot{\theta} = \frac{r}{2l}(\omega_d - \omega_i)$$

2.1.1

Modelo Odométrico del robot

La técnica para determinar la posición del robot en todo momento, respecto a un marco de referencia conocido fue el método odométrico, el cual consiste en determinar la posición y orientación del vehículo en todo momento, esto es posible debido a que conociendo el modelo del robot, se integran las ecuaciones que representan al sistema, esto nos otorga la posición y orientación de el robot en todo momento, de tal forma que el modelo para la estimación queda de la siguiente forma [6]:

$$x = \int_0^t \frac{r\cos(\theta)}{2}(\omega_d + \omega_i)\,dt + x_0 \tag{2.11}$$
$$y = \int_0^t \frac{r\sin(\theta)}{2}(\omega_d + \omega_i)\,dt + y_0$$
$$\theta = \int_0^t \frac{r}{2l}(\omega_d - \omega_i)\,dt + \theta_0$$

En donde:

x_0, y_0 y θ_0 representan la posición incial del robot en el marco de referencia.

2.2 Herramientas de Software

Para este proyecto se emplearon dos herramientas para la programación de las diversas tareas del robot, en el caso de las tareas de adquisición de datos y movimiento del robot, se utilizó LabVIEW en su versión 2011 con diversos paquetes (los cuales se detallan en este capitulo) y para el procesamiento de vídeo se empleó Microsoft Visual C# 2010.

2.2.1

LabVIEW

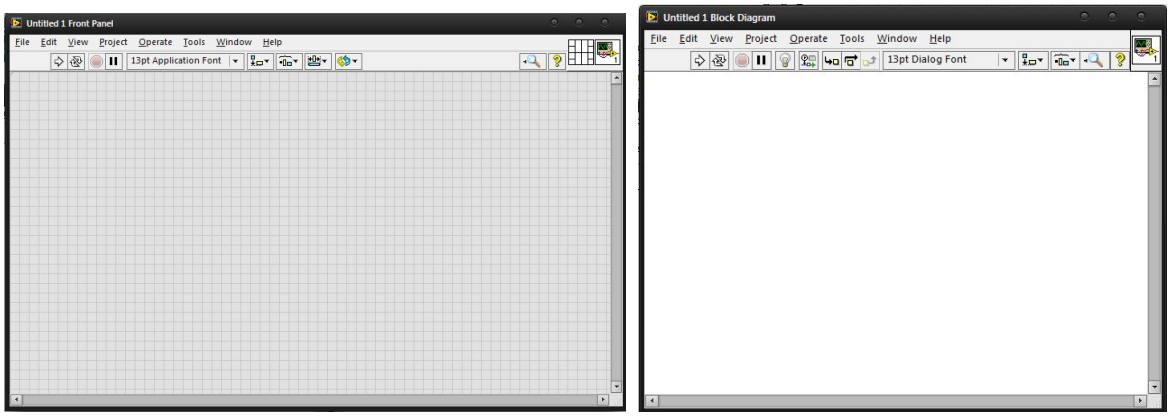
LabVIEW es un lenguaje gráfico de programación que emplea iconos en lugar de líneas de texto (por lo que es conocido como un lenguaje G), y a diferencia de lo establecido comúnmente en la programación basada en texto, donde las instrucciones determinan la ejecución del programa, LabVIEW emplea el flujo de datos y este es el que determina la ejecución del programa.[7]

En el caso de los programas que se realizan en LabVIEW estos se denominan instrumentos virtuales y tienen una extensión (*.vi), se denominan así debido a que la interfaz que presenta tiene mucha similitud con un instrumento físico de medición.

Un instrumento virtual está compuesto principalmente por una interfaz interactiva de usuario, un diagrama de flujo de datos que se puede entender como el código fuente y las conexiones de iconos que permiten la anidación con otros instrumentos virtuales.

2.2.1.1 Entorno de programación

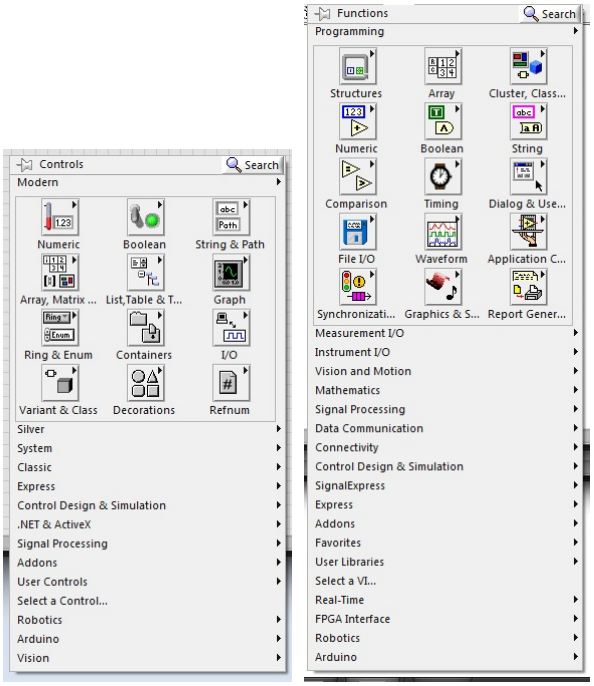
LabVIEW es un lenguaje que tiene su propio entorno de programación, y este se compone principalmente de dos ventanas; el *Panel Frontal* (Figura 2.2a), que es la interfaz interactiva del usuario, la cual contiene los indicadores, botones y etiquetas que permiten la interacción entre el usuario y el programa. Por otro lado, se encuentra la ventana que contiene el *Diagrama de Bloques* (Figura 2.2b), en donde se desarrolla la programación realizada mediante la conexión de los distintos iconos.



(a) Panel Frontal (b) Diagrama de Bloques

Figura 2.2 Entorno de programación de LabVIEW.

La programación de los elementos que componen al programa se lleva acabo mediante la selección de iconos de paletas, estas son distintas para el panel frontal (Figura 2.3a) y para el diagrama de bloques (Figura 2.3), en el primer caso las paletas están orientadas a la composición de la interfaz del usuario con esto me refiero a que en esta paleta se encuentran los controles e indicadores que llevan a cabo la inserción y presentación de datos. En el caso de las paletas que se localizan en el diagrama de bloques, estas están orientadas a la programación, esto quiere decir que algunas de sus funciones son la creación de estructuras, manipulación de los datos, definición de variables, comunicación entre aplicaciones y/o dispositivos, etc.



(a) Panel frontal (b) Diagrama de Bloques

Figura 2.3 Paletas de herramientas.

2.2.1.2 Complementos de LabVIEW

La empresa National Instruments, es la responsable del soporte de LabVIEW por lo que distribuye distintos complementos para el desarrollo de aplicaciones, ejemplos de estos son: el *toolkit* para la tarjeta de desarrollo Arduino, inclusión de código en otros lenguajes, comunicación mediante protocolos industriales, etc. Por lo que para la ejecución del proyecto es necesario agregar complementos para su realización, estos complementos están contenidos como versiones demostrativas con DaNI, a continuación se realiza una descripción de las paletas contenidas en estos:

LabVIEW Robotics
Este es el complemento principal que apoya en la programación del robot, ya que contiene una serie de clases y tipos de datos que se ajustan a la programación de tareas para proyectos de robótica, este complemento no solo genera paletas en LabVIEW, sino que genera una interfaz propia (Figura 2.4) que permite una fácil navegación y ubicación de las herramientas necesarias para la realización de este tipo de proyectos. Esta herramienta cuenta con un asistente para realización de proyectos que facilita el proceso de generación de archivos necesarios, además de contar con un simulador de distintos tipos de robots para probar el funcionamiento de algoritmos.

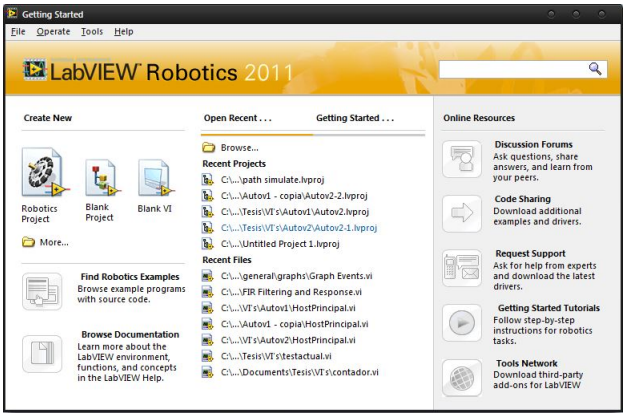
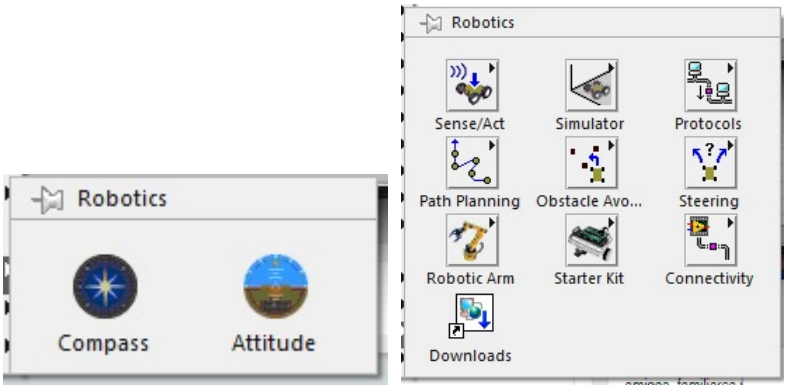


Figura 2.4 LabVIEW Robotics.

Las herramientas con las que cuentan las paletas de este complemento para el panel frontal (Figura 2.5a) son dos y estos son indicadores; el primero es una brújula que recibe un dato de tipo flotante y apunta en la dirección de los grados recibidos; el segundo es un indicador de Altura, el cual como argumento recibe los valores de los ángulos de alabeo (Roll) y elevación (Pitch). En el caso de la paleta que pertenece al Diagrama de Bloques existen muchas más herramientas, las cuales se agrupan en distintas categorías dependiendo su uso. Estas se explican a continuación;

- **Sensors and Actuators *(Sensores y Actuadores)***: Aquí se agrupan algunas funciones que ayudan al sensado, ya que proveen formas de interacción con distintos dispositivos comúnmente usados en la robótica, como lo son: GPS's, Sensores de luz, Brújulas electrónicas, Sensores térmicos, Láseres, etc.
- **Simulator *(Simulador)***: Contiene las funciones necesarias para llevar a cabo la simulación del robot, algunas de ellas son la creación de objetos y entornos, así como el control de flujo de datos.
- **Protocols *(Protocolos)***: Esta paleta tiene herramientas para la manipulación de datos de acuerdo con la norma NMEA 0183 (Véase Anexo E), así como ejemplos de como lograr la comunicación a través del FPGA que contiene la tarjeta usando los protocolos I2C y SPI.
- **Path Planning *(Planeación de Trayectorias)***: Agrupa instrumentos virtuales que se emplean para generar trayectorias a un punto deseado en un mapa que representa el ambiente del vehículo, la elección de la trayectoria puede ser basada en los algoritmos A* y AD*
- **Obstacle Avoidance *(Evación de obstaculos)***: Contiene VI's para la identificación de obstáculos enfrente del robot, los algoritmos empleados utilizan métodos estadísticos para la elección de la ruta.
- **Steering *(Direccionamiento)***: Contiene elementos para obtener información acerca de la trayectoria del vehículo, tomando en cuenta la estructura del robot, el tipo y numero de ruedas y la velocidad de los motores que utilizan las ruedas.
- **Robotic Arm *(Brazo Robótico)***: Esta paleta es empleada en el control y simulación de brazos robóticos, tiene herramientas para la realización de modelados de cinemática directa e inversa, así como la obtención de los modelos dinámicos.

- **Starter Kit** ***(Paquete de Inicio)***: Aquí se encuentran los VI's principales (lectura de *encoders*, aplicación de velocidad a los motores, lectura de sensor ultrasónico, etc.) para el control de DaNI en sus versiones 1.0 y 2.0.
- **Connectivity** ***(Conectividad)***: Permite buscar software de otras compañías (Skilligent y JTK) relacionadas con actividades entorno a la robótica.
- **Downloads** ***(Descargas)***: Contiene un hipervínculo que dirige al sitio de LabVIEW Robotics, en donde es posible encontrar material para descarga de ejemplos relacionados con robots.



(a) Panel Frontal (b) Diagrama de Bloques

Figura 2.5 Paletas del complemento LabVIEW Robotics

LabVIEW Real-Time

Este complemento tiene la función de servir de apoyo en la construcción de sistemas embebidos autónomos, los cuales se desarrollan como un instrumento virtual y posteriormente se descargan en el dispositivo deseado, el módulo soporta programación para NI CompactRIO, NI CompactDAQ Autónomo, PXI, sistemas de visión y computadoras personales destinadas a la ejecución de sistemas de adquisición y control en tiempo real, la función de este módulo en el proyecto consiste en realizar la descarga del programa en la tarjeta sb-RIO que contiene el robot DaNI.
La paleta de funciones (Figura 2.6) que genera el modulo Real-Time contiene las siguientes herramientas:

- **RT FIFO**: Es un conjunto de herramientas para el manejo de estructuras del tipo *'First Input, First Output'* (Colas).
- **RT Timing** ***(Sincronización en tiempo real)***: Estas herramientas sirven para controlar el tiempo de los ciclos de ejecución en los instrumentos virtuales.
- **RT Utilities** ***(Utilidades en tiempo real)***: Este conjunto comprende herramientas para la obtención de información del dispositivo objetivo en el que se ejecutará el programa; como lo es el nombre, los re-inicios y la protección para la ejecución del programa en el dispositivo.
- **Function Blocks** ***(Bloques de funciones)***: Aquí se contienen elementos de bloques para realizar programación siguiendo el estándar IEC 61131-3 en el formato de Diagrama de Bloques de Funciones, contiene elementos para la detección de flancos, temporizadores, acumuladores y la posibilidad de diseñar controladores PID.

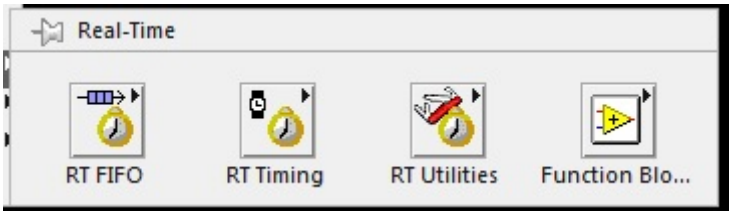


Figura 2.6 Paleta del módulo LabVIEW Real-Time

LabVIEW FPGA

Como su nombre lo indica este complemento tiene la función de realizar las labores de programación y comunicación en distintos FPGA's, en este proyecto se emplea para configurar los puertos de entrada y salida del FPGA con el que cuenta la tarjeta contenida en el robot. Los archivos generados se conocen como FPGA VI's.

La descripción de los iconos contenidos en esta paleta (Figura 2.7), es la siguiente:

- **Open FPGA VI Reference** ***(Abrir FPGA VI de referencia)***: Abre un archivo *pre-compilado*[1] para el FPGA en el cual se definen los puertos a utilizar así como la configuración de estos.
- **Read/Write Control** ***(Control de lectura/escritura)***: Lee o escribe valores obtenidos de la configuración del FPGA seleccionado con la herramienta anterior.
- **Invoke Method** ***(Invocación de metodo)***: Invoca métodos de interfaz del FPGA o una acción del dispositivo al que pertenece el FPGA. Emplea métodos para las siguientes funciones: descarga, parada, reinicio e inicio del FPGA VI en el FPGA objetivo.
- **Close FPGA VI Reference** ***(Cerrar FPGA VI de referencia)***: Cierra el archivo referencia del FPGA VI.
- **Dynamic FPGA Interface Cast** ***(Interfaz de reasignación dinámica del FPGA)***: Esta función realiza una conversión del tipo de elementos que contiene la referencia del FPGA VI.
- **FPGA Interface Dynamic Refnum**[2] ***(Interfaz dinámica Refnum del FPGA)***: Es un tipo de dato el cual hace referencia a una interfaz ya definida, se puede interpretar como un dato constante.
- **Scalling** ***(Escalamiento)***: En esta paleta se agrupan herramientas que modifican el tiempo y la tasa de muestreo, así como herramientas que sirven de complemento a funciones de análisis de señales y matemáticas.
- **Advanced** ***(Avanzado)***: Esta paleta contiene la herramienta para convertir una referencia especifica de un FPGA VI en una referencia genérica, lo cual permite la compatibilidad con distintos VI's que no están diseñados específicamente para FPGA VI's.
- **Peer to peer streaming** ***(Transmisión continua de punto a punto)***:En esta paleta se agrupan elementos para desarrollar comunicación entre el dispositivo donde se descargara la aplicación y algún otro, sin necesidad de pasar por el *host* en ningún momento.
- **Simulation** ***(Simulación)***: Emplea VI's de simulación que interactúan con otros simuladores.

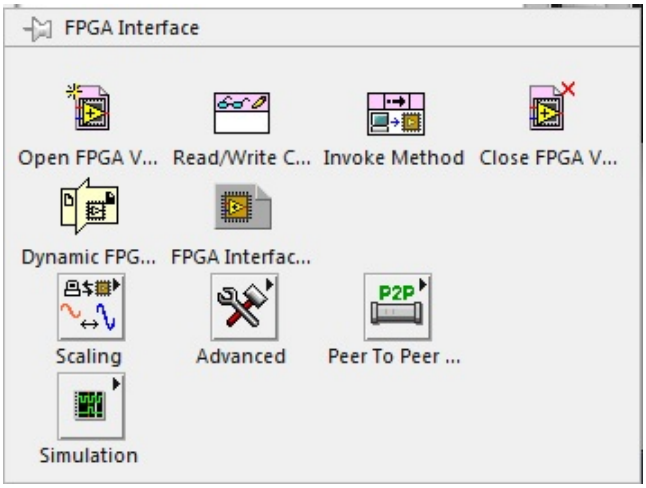


Figura 2.7 Paleta del módulo LabVIEW FPGA

LabVIEW Vision and Motion

Este complemento está desarrollado para el trabajo en sistemas de visión y el procesamiento de vídeo e imágenes, aunque debido a que es necesario contar con una licencia para el desarrollo de aplicaciones, no es posible la explotación de estas herramientas. Por lo que solo se emplea como un ejemplo en la presentación del vídeo obtenido.

[1] El proceso para la generación de este archivo puede consultarse en el Anexo C.

[2] Es un tipo de control utilizado en LabVIEW, se puede considerar como un *control constante*.

2.2.2

Microsoft Visual Studio

Visual Studio es la interfaz de desarrollo que Microsoft ofrece a los programadores para el desarrollo de distintas aplicaciones y páginas web, es capaz de soportar varios tipos de lenguajes como: C++, C# (Sharp), Visual Basic, Java, Python, entre otros. Dadas las facilidades que ofrece se pueden crear aplicaciones que se comuniquen entre páginas web, dispositivos móviles, estaciones de trabajo, sistemas embebidos, etc. En este caso el lenguaje empleado para llevar a cabo el procesamiento de vídeo fue C#.

2.2.2.1 Lenguaje C#

C# es un lenguaje desarrollado por Microsoft para la plataforma .NET como un lenguaje que facilitara la migración a esta nueva plataforma. Tiene sus raíces en los lenguajes C, C++ y Java; contiene las mejores características de cada uno y agrega nuevas propias. C# es un lenguaje orientado a objetos y contiene una poderosa biblioteca de clases, que consta de componentes pre-construidos que permiten el desarrollo de aplicaciones de una manera más rápida. Este lenguaje es apropiado para las tareas de desarrollo de aplicaciones demandantes, en especial para aquellas basadas en la Web.
El lenguaje de programación C# original se estandarizó a través de Ecma International en diciembre del 2002, como *Estándar ECMA-334: Especificación del lenguaje C# (Disponible en: www.ecma-international.org).*
Microsoft ha propuesto varias extensiones del lenguaje que se han adoptado como parte del estándar Ecma revisado, esta nueva versión es conocida como **C# 2.0**.[8]

2.2.2.2 Aforge.NET Framework

Gracias a la diversidad que ofrece la plataforma .NET, existen distintas bibliotecas de clases que están diseñadas para tareas especificas; en este proyecto se emplearon las de Aforge. NET. Este es un proyecto que provee bibliotecas para C# diseñadas por desarrolladores e investigadores en Visión por computadora e inteligencia artificial, procesamiento de imágenes, redes neuronales, algoritmos genéticos, lógica difusa, aprendizaje automático, robótica, etc.

Ejemplos de las bibliotecas que contiene, son las siguientes:

Aforge.Imaging : Contiene filtros y rutinas para el procesamiento de imágenes.
Aforge.Vision : Biblioteca con funciones para visión por computadora.
Aforge.Video : Bibliotecas con funciones para el procesamiento de vídeo.
Aforge.Neuro : Bibliotecas para el computo de redes neuronales.
Aforge.Genetic : Biblioteca de programación evolutiva.
Aforge.Robotics : Biblioteca que provee soporte a algunos paquetes de robótica.
etc...

2.3 Plataforma DaNI

DaNi es el nombre que recibe el robot (Figura 2.8) contenido en el paquete de inicio para *NI Labview Robotics*, esta plataforma esta diseñada para la enseñanza de robótica, la realización de prototipos y la investigación [10].

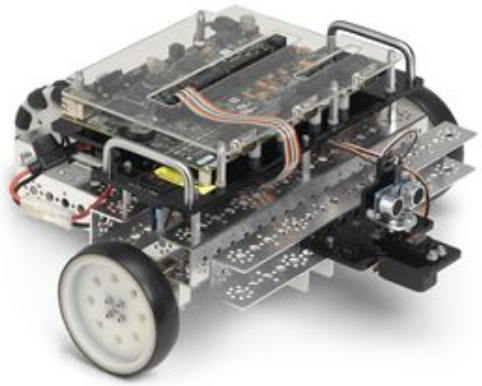


Figura 2.8 [**Robot DaNI**] Recuperado de: *http://www.ni.com/white-paper/11564/en/*

El robot está constituido por una estructura metálica que le otorgan una configuración diferencial, dos motores de DC que cuentan con codificadores ópticos de cuadratura (encoders), un controlador para el movimiento de los motores, una batería para la alimentación del robot, un sensor ultrasónico, un servomotor para el posicionamiento de este y una computadora NI sbRIO-9632.

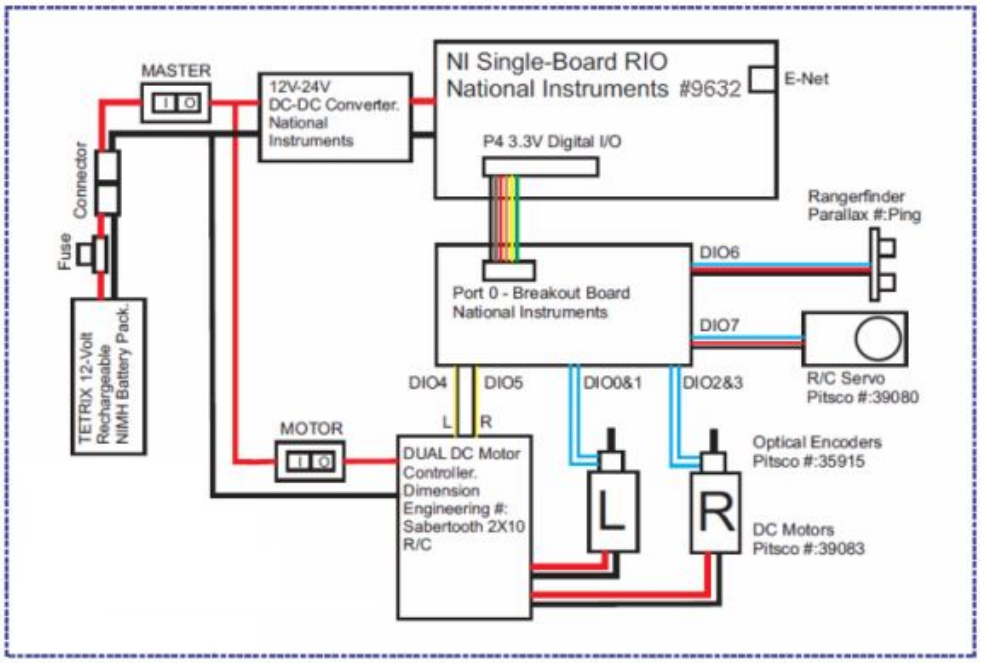


Figura 2.9 [**Diagrama de bloques de los componentes de DaNI**] Recuperado de: [11]

En las paginas siguientes se realiza una descripción detallada de cada uno de los elementos de manera individual.

2.3.1

Tarjeta NI sbRIO-9632

La tarjeta NI sbRIO-9632 (Figura 2.10) es un dispositivo de control y adquisición de datos que contiene un procesador en tiempo real, un FPGA y puertos configurables de entrada y salida (analógicos y digitales), esta tarjeta se alimenta con 24 VDC [12].

Figura 2.10 [**Tarjeta NI sbRIO-9632**] Recuperado de: *http://sine.ni.com/nips/cds/view/p/lang/es/nid/205899*

La lista de elementos que contiene la tarjeta, es la siguiente:

- Un *FPGA Xilinx Spartan* de 2M.
- Un procesador *Freescale* de 400 MHz.
- 110 lineas digitales de 3.3V (tolerable a niveles TTL) de entrada y salida.
- 32 entradas analógicas.
- 4 salidas analógicas.
- 1 puerto serial para comunicación RS 232.
- 1 puerto Ethernet 10/100BASE-T

2.3.2

Motores y Ruedas

El robot cuenta con dos motores de la marca PITSCO EDUCATION modelo TETRIX® MAX DC Gear (Fig. 2.11a), los cuales requieren una alimentación de 12 VDC y alcanzan velocidades de $512\,RPM$; y un torque de $3.9\,kg \cdot cm$ [13]. Cuenta con tres ruedas, dos son del tipo estándar actuadas con un radio $r = 0.0508[m]$ y la distancia entre ambas es, $2l = 0.3302[m]$, la tercera rueda corresponde a una rueda sueca pasiva con 10 barriles de rodamiento a 90° y un diámetro de $4\,[in]$.

(a) Motor TETRIX® MAX DC Gear (b) Rueda estandar (c) Rueda sueca

Figura 2.11 [**Motor y ruedas**] Recuperado de: *http://www.pitsco.com/Engineering*

Como se muestra en la Figura (2.9), para el control de velocidad de los motores se utiliza el controlador *Sabertooth dual 10A motor driver for R/C* (Figura 2.12) de la marca Dimension Engineering, el cual soporta una corriente de 10 A como nominal y picos de 15 A[14], el controlador es dual por lo que solo es necesario uno para ambos motores, la forma de variar la velocidad es a través de una modulación por ancho de pulso (PWM), la cual se realiza mediante las lineas digitales 4 y 5 del puerto 0 que se encuentran en la tarjeta sbRIO-9632.

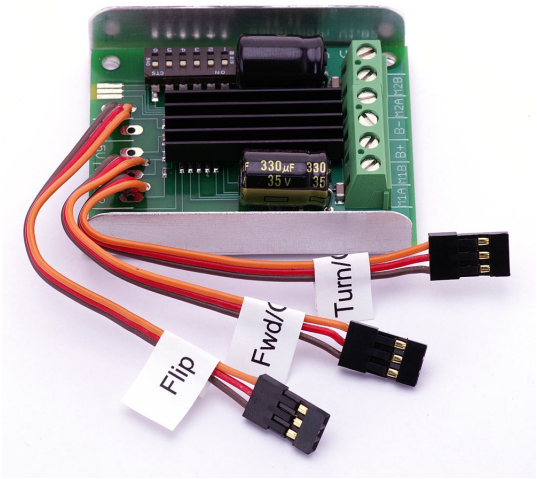


Figura 2.12 [**Sabertooth dual 10A motor driver for R/C**] Recuperado de: *https://www.dimensionengineering.com/products/sabertooth2x10rc*

2.3.3 Codificadores ópticos de cuadratura

Para obtener la velocidad de las ruedas se utilizan codificadores opticos incrementales de cuadratura (”encoders”) en este caso se utiliza el modelo W35915 de la marca PITSCO[15], estos *encoders* tienen una resolución de 400 pulsos/revolución. La ubicación de estos en el robot, se aprecia en la Figura 2.13.

Para la lectura de los pulsos, se emplean las líneas digitales 0,1,2 y 3 del puerto 0 de la tarjeta sbRIO-9632.

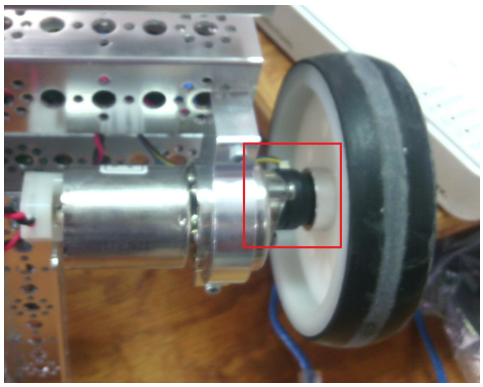
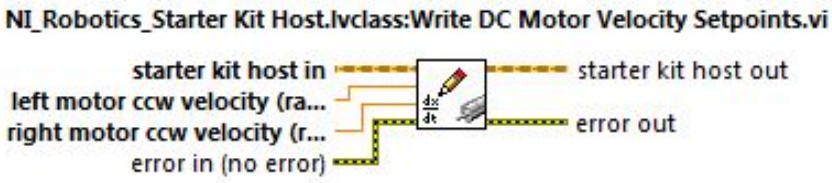


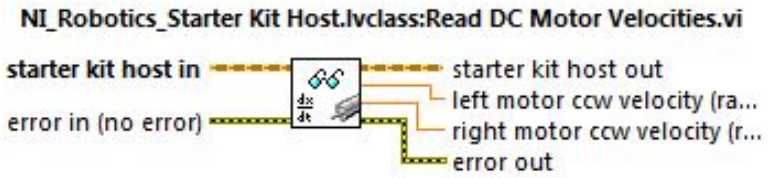
Figura 2.13 Codificadores ópticos.

2.3.3.1 Lectura y escritura de velocidades de las ruedas

El proceso con el cual se realiza la comunicación con la tarjeta con los motores es mediante la clase de LabVIEW *Write DC Motor Velocity Setpoints* (Figura 2.14a), la cual se encuentra en la paleta *Robotics*, esta clase permite generar la señales enviadas al controlador de los motores, el cual fija la velocidad asignada como argumento en la clase este parámetro se encuentra en [rad/s].
De forma similar para obtener la lectura de los *encoders* se emplea la clase *Read DC Motor Velocities* que se encuentra en la paleta *Robotics*, ésta retorna los valores de las velocidades izquierda y derecha respectivamente en [rad/s].



(a) Write DC Motor Velocity Setpoints



(b) Read DC Motor Velocities

Figura 2.14 Clases para lectura y escritura de velocidades

2.3.4

Sensor ultrasonico

Para la detección de objetos, el robot cuenta con un sensor ultrasónico modelo PING))) de la marca PARALLAX (Figura 2.15), el cual mediante pulsos ultrasónicos puede medir distancias entre el sensor y el objeto con un alcance de entre $2[cm]$ y $3[m]$[16], se alimenta con 5 V y como se aprecia en la Figura 2.9 los datos del sensor se envían a través de la línea digital 6 del puerto 0.

(a) **Sensor** Recuperado de: *https://www.parallax.com/product/28015*

(b) **Servomotor** Recuperado de: *http://www.tetrixrobotics.com/Servos/180_Standard-Scale_HS-485HB_Servo_Motor*

Figura 2.15 Sensor ultrasónico PING)))

Además, para poder orientar el sensor, se cuenta con la ayuda de un servomotor modelo HS-485HB de la marca HITEC (Figura 2.15b), el cual tiene una apertura de $180°$ y un torque de $6[kg \cdot cm]$ [17], debido a la posición del sensor en el robot, no se emplean los 180 grados para la realización del barrido, solo se emplean 130 grados, los cuales quedan distribuidos como se muestra en la Figura 2.16, el punto medio del robot se considera como $0°$ y la asignación de los grados se realiza en sentido anti-horario de $-65°$ a $65°$.

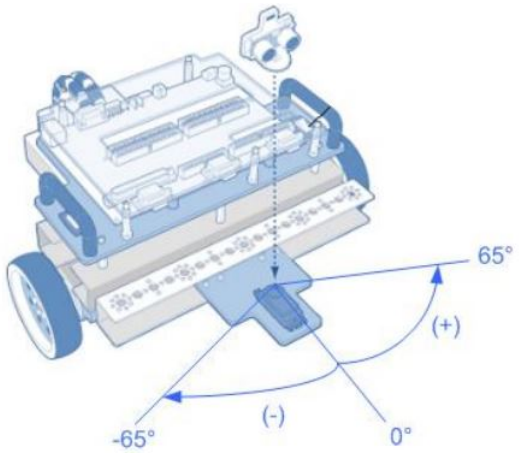


Figura 2.16 [**Rango de barrido del sensor ultrasónico**] Recuperado de: [18]

2.3.4.1 Interacción con el robot

Para la obtención de datos del sensor ultrasónico se emplea la clase *Read PING))) Sensor Distance*, el empleo de esta clase facilita mucho el uso de este componente, ya que internamente tiene el algoritmo para transformar el ancho de pulso recibido por este en la distancia a la cual se encuentra el objeto en el que rebota la señal, como parámetros recibe cual será la distancia o el dato a mostrar en caso de que no se detecte ninguna señal o se exceda el tiempo pre-asignado para la espera de la señal (*"timeout"*); y retorna un valor flotante el cual representa la distancia a la que se encuentra el objeto y un dato de tipo booleano que indica si se ha excedido o no el *'timeout'*.

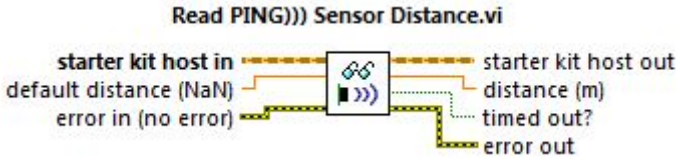


Figura 2.17 Clase: Read PING))) Sensor Distance.

Mientras que para el posicionamiento del sensor, existen varias clases que ayudan con este propósito todas estas se encuentran en la paleta *Robotics*, la primera de ellas (de izquierda a derecha) es la clase *Write Sensor Servo Angle*, tiene la función de recibir como argumento la posición que tendrá el servo motor con respecto a la asignación de grados que se muestran en la figura 2.16, las dos clases siguientes tienen la función de asignar el *'offset'* que servirá como referencia al servomotor, la clase *Write Sensor Servo Offset* fija el punto a ser considerado como el nuevo *'cero grados'* y la clase *Read Sensor Servo Offset* retorna el valor de la posición dentro del rango de barrido del servomotor que es tomada como cero.

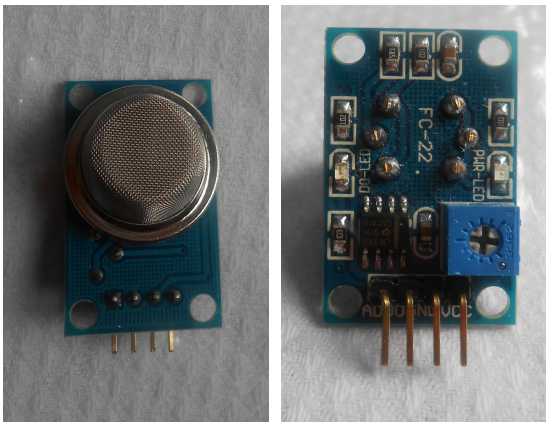
Figura 2.18 Clases para la lectura y escritura del servomotor.

2.4 Sensores agregados y acondicionamiento

Debido a que la intención del proyecto es la realización de un prototipo de robot para exploración, se incorporaron elementos de sensado para el cumplimiento de este propósito, la descripción, el funcionamiento e interacción de estos con el robot, se muestra a continuación.

2.4.1 Sensor de calidad del aire

Para obtener una aproximación de la calidad del aire que se encuentra en el ambiente a explorar por el robot, se utilizó el sensor MQ-135, este consiste en un microtubo cerámico de Al_2O_3(Oxido de Aluminio), el cual es sensible a diversos contaminantes, esta sensibilidad ocasiona una variación en la resistencia del tubo, por lo cual a través de una resistencia de carga es posible obtener una lectura en voltaje [19].



(a) Frontal (b) Trasera

Figura 2.19 Sensor MQ-135

Para el acondicionamiento de este sensor, solo es necesario implementar una resistencia de carga, y debido a las condiciones del modulo (Figura 2.19b) que se obtuvo, este ya cuenta con una resistencia, la cual es de $1[K\Omega]$.

2.4.2 Detector de vibración

El transductor que se empleo para la obtención de las vibraciones que se presenten en el ambiente en que se desenvuelva el robot, consiste en una película piezoeléctrica (Figura 2.20) el cual por las características que presenta es suficiente para este propósito.

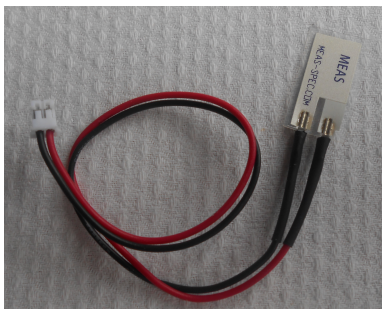


Figura 2.20 Transductor de vibración.

Este transductor está contenido en el paquete para la detección de vibración provisto por la compañía Grove, pero debido a las características que ofrece éste; de proporcionar una señal digital se optó por realizar una amplificación de la señal otorgada por el transductor. Este proceso se describe con mayor detalle en el capítulo siguiente.

2.4.3

Sensor de temperatura

El sensor que se empleó para la medición de temperatura del ambiente en el que se desenvolverá el robot fue el LM35 de la compañía *National Semiconductors* en su modelo de encapsulado TO-92 (Figura 2.21). Se eligió este sensor debido a su fácil calibración e implementación ya que otorga una señal de voltaje proporcional a la temperatura que se esta midiendo.

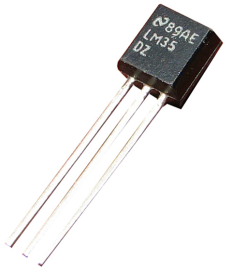


Figura 2.21 [**Sensor LM35**] Recuperado de: *http://www.trastejant.es/tutoriales/electronica/sensordetemperatura_lm35.html*

2.4.4

Detector de flama

El modulo empleado es el SEN05082P (Figura 2.22), éste se compone de un fototransistor NPN YG1006 el cual es capaz de detectar ondas de luz con longitudes entre 760-1100[nm], cuando detecta luz la señal del transistor pasa a un comparador que mediante el ajuste de un potenciómetro otorga una señal digital según el voltaje emitido por el transistor. Genera una señal alta en caso de que la intensidad de la flama supere un umbral y una baja en caso de que no supere éste.

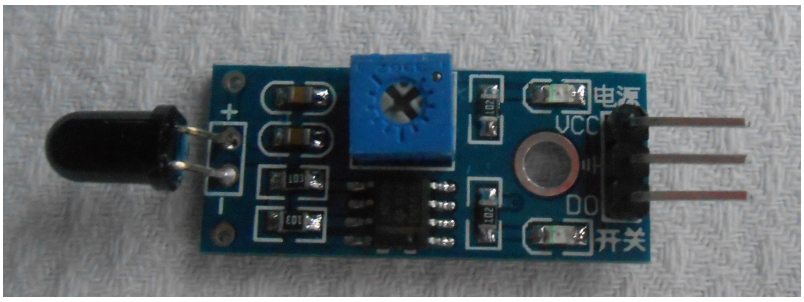


Figura 2.22 Modulo detector de flama.

2.4.5

Modulo GPS

El módulo EVA2035-H (Figura 2.23) de *Maestro GPS Receiver* fue el seleccionado para la recepción de las coordenadas terrestres, al ser un módulo de desarrollo es posible la obtención de datos a través de dos formas; mediante un cable USB conectado a un puerto serial de alguna computadora y a través de los bloques con los que cuenta, éstos envían datos mediante los protocolos UART y SPI.

Dado que el sistema GPS es empleado en diversas aplicaciones, se consideró realizar una descripción de este sistema en el apartado siguiente.

2.4.5.1 Sistema GPS

GPS(*Global Positioning System*) es un sistema de satélites que orbitan alrededor de la tierra y tiene como objetivo la determinación de las coordenadas espaciales de puntos respecto de un sistema de referencia mundial. Este sistema puede ser descompuesto en tres segmentos básicos, de los cuales, los dos primeros son de responsabilidad militar:

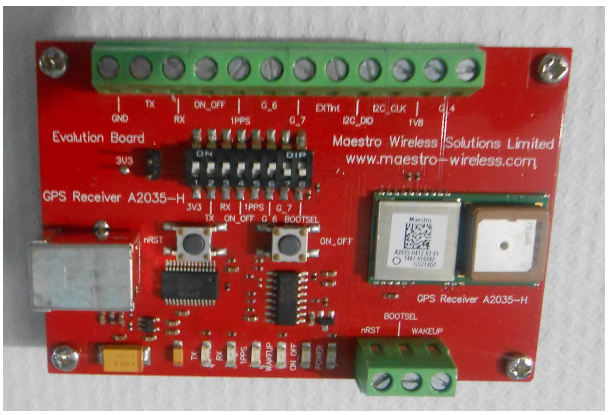


Figura 2.23 Modulo GPS EVA2035-H.

- **Segmento espaciado:** Este segmento consiste en una constelación de satélites (Figura 2.24) transmitiendo señales de radio a los usuarios. Los Estados Unidos de América, son los responsables de mantener en operación al menos 24 satélites, el 95 % del tiempo. Para asegurar este cometido, la Fuerza Aérea de este país ha enviado 31 satélites operacionales en los últimos años.

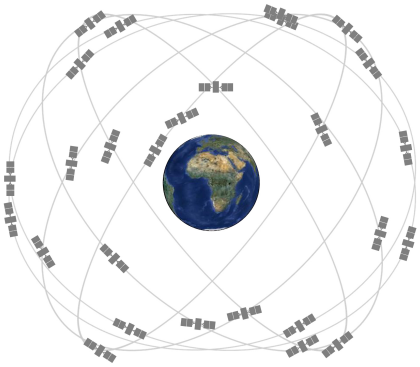


Figura 2.24 [**Constelación de los 24 satelites, de acuerdo al Standard Positioning Service (SPS)**] Recuperado de: *http://www.gps.gov/systems/gps/space/*

- **Segmento de control:** Consiste en una red global de instalaciones en tierra (Figura 2.25) que siguen la ruta de los satélites de GPS, monitorizan sus transmisiones, analizan su desempeño; y envían comandos y datos a los satélites de la constelación.

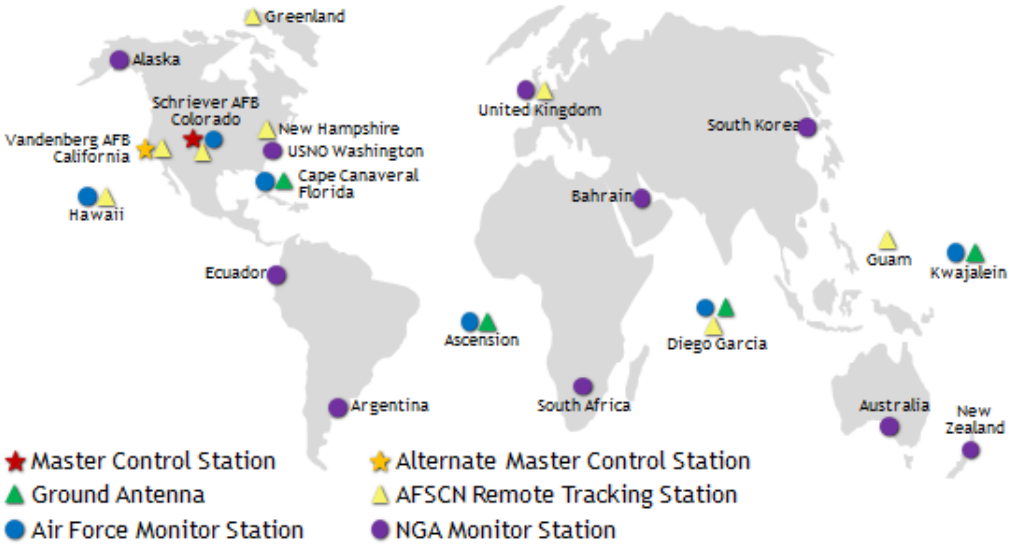


Figura 2.25 [**Mapa de las estaciones terrestres**] Recuperado de: *http://www.gps.gov/systems/gps/control/*

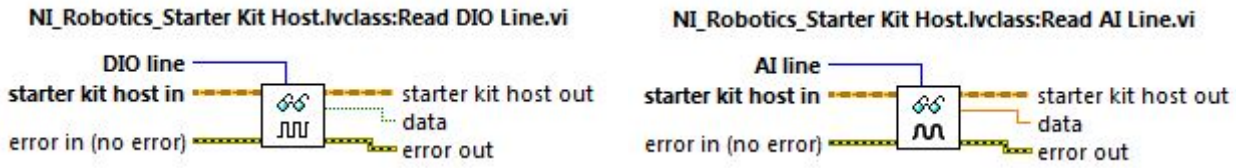
- **Segmento usuario:** Está formado por las antenas y receptores pasivos situados en la tierra. Los receptores calculan distancias y proporcionan una estimación de posición y de tiempo a partir de los mensajes que provienen de cada satélite visible.

El sistema GPS, tiene por objetivo calcular la posición de un punto cualquiera en el espacio, mediante las coordenadas (x, y, z), partiendo del cálculo de las distancias del punto a un mínimo de tres satélites cuya localización es conocida.
El concepto básico de posicionamiento por GPS es el *Posicionamiento por intervalos*, para el caso de un sistema tridimensional, este se basa en el siguiente concepto, si se conoce la distancia a un punto conocido, el punto conocido se puede situar en el centro de una esfera, y la distancia será el radio de esta esfera.
La distancia entre el receptor (Segmento usuario), y un satélite, se mide multiplicando el tiempo que la señal tarda en ser transmitida-recibida por la velocidad de la propagación de la señal. Para medir dicha velocidad, es necesario que los relojes de los satélites y de los receptores estén sincronizados, pues deben generar el mismo código. La desviación en los relojes de los dispositivos receptores añade una incógnita más que hace necesario un mínimo de cuatro satélites para estimar correctamente las posiciones.
Para aplicar esta técnica en el GPS, se obtienen la distancia del receptor a tres satélites, con lo cual se obtienen tres distancias. Con estas mediciones se forman tres esferas, cuyas intersecciones se encuentran en dos puntos en el espacio, de los cuales solo uno es valido ya que el otro se encuentra fuera del planeta.

2.4.6

Interacción entre sensores agregados y el robot

La lectura de los sensores agregados a excepción del GPS, se realiza mediante el acceso a los puertos de lectura digitales y analógicos con los que cuenta la tarjeta, el modulo de LabVIEW Robotics ofrece clases para este propósito, con la particularidad de que solo se pueden emplear puertos ya definidos.
En caso de que se quisieran utilizar mas puertos digitales o analógicos, o en su defecto cambiar el modo de lectura analógica es necesario realizar una nuevo programa empleando el compilador para FPGA que viene con el modulo, este proceso se detalla en el Anexo C.



(a) Linea Digital (b) Linea Analógica

Figura 2.26 Clases para la lectura de puertos

Mientras que para la obtención de los datos que envía el modulo GPS, se emplea el puerto serial que sigue el protocolo RS-232. Se empleó el paquete VISA *(Virtual Instrument Software Arquitecture)*; VISA provee la interfaz de programación entre el hardware y la plataforma de desarrollo (en este caso LabView). Éste es un estándar para configurar, programar y depurar sistemas de instrumentación que comprenden interfaces GPIB, VXI, PXI, serial(RS-232/RS-485), Ethernet/LXI y/o interfaces USB.

2.4.7

Cámara y micrófono

Para la obtención de vídeo y audio en el robot, se decidió emplear un teléfono celular Huawei Y300 con camara de 5 MP y sistema operativo Android, el cual mediante la aplicación **IP Webcam** (Figura 2.27), que se encuentra disponible en la tienda de aplicaciones (*Google Play Store*).
Esta aplicación resulta muy útil ya que durante su ejecución el celular se convierte en una cámara IP la cual transmite audio y vídeo a distancia en *streaming*, ademas de que dadas las características con las que cuenta actualmente un teléfono celular, este es capaz de almacenar audio y vídeo, poniendo a un dispositivo con estas características en competencia con cámaras IP del mercado a un costo muy bajo.

Figura 2.27 Aplicación IP Webcam

2.5 Sistema de Comunicaciones

La comunicación de los tres elementos principales (computadora del operador, cámara (teléfono celular) y robot) que conforman al robot prototipo se realizo mediante una red WLAN con topología de estrella.

Una red WLAN es como se le denomina a una red inalámbrica de área local. Ésta es una expansión de las redes LAN, las cuales se denominan de área local debido a su tamaño ya que comúnmente se usan en regiones pequeñas y para la interconexión de equipos. El protocolo que se emplea en la mayoría de los casos es el Ethernet.

En redes con topología estrella (Figura 2.28) cada estación esta conectada directamente a un nodo central común, generalmente a través de enlaces punto a punto.
Existen dos alternativas para el funcionamiento del nodo central. Una es en modo difusión, en la que la transmisión de una trama por parte de una estación se retransmite sobre todos los enlaces de salida del nodo central, la otra consiste en que el nodo central funcione como dispositivo de conmutación de tramas. Una trama entrante se almacena temporalmente en el nodo y se retransmite sobre un enlace de salida hacia la estación del destino.[24]

Para este proyecto se configuro el nodo central como dispositivo de conmutación, y la conexión entre el nodo central y el robot se realiza a través de 'par trenzado' (UTP-5), mientras que la conexión del teléfono celular y la computadora del operador con el nodo central es mediante Wi-Fi (protocolo IEEE 802.11).

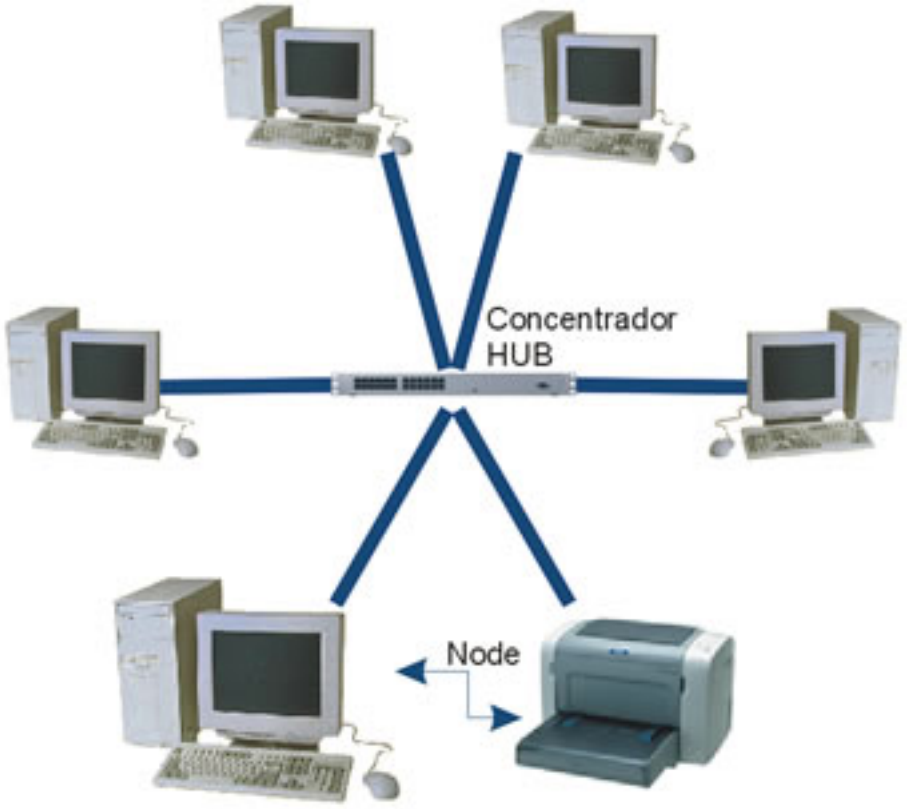


Figura 2.28 [**Topología en Estrella**] Recuperado de: *http://www.ipcolombia.com/cap_redes_lan1.htm*

3

Desarrollo del prototipo

En este capítulo se detallan las actividades que se llevaron a cabo para la realización del prototipo; se explica cual es la forma en que se realizan las distintas actividades que realiza el robot, se describe el principio de operación de los sensores y el método empleado para la obtención de datos del robot, así como el funcionamiento de los distintos instrumentos virtuales para la operación del prototipo.

3.1 Descripción general

Las tareas que realiza el *Sistema de Exploración y Monitoreo*[1] pueden ser agrupadas en cuatro categorías principales, las cuales se representan por lo bloques mostrados en el diagrama de la Figura 3.1. Para un mayor detalle sobre las distintas tareas que realiza el sistema, en la Figura 3.2 se presenta el diagrama de bloques del sistema de una forma mas detallada.

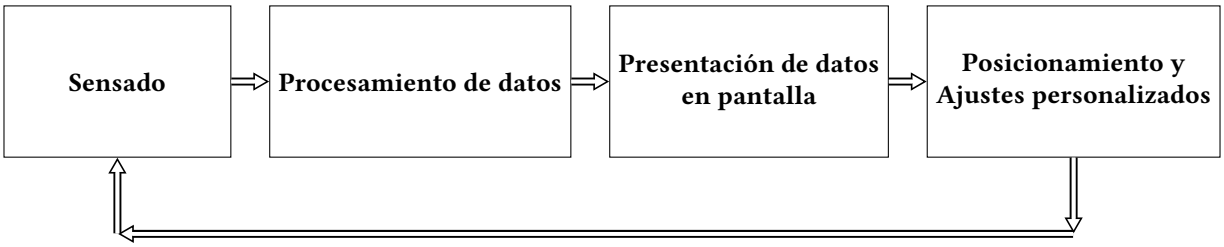


Figura 3.1 Diagrama de bloques del Sistema de Exploración y Monitoreo

La descripción de cada uno de los bloques es la siguiente:

Sensado: Dentro de este bloque se encuentran las siguientes tareas; **Lectura de sensores** (Véase 3.4.3), **Recepción de cadenas de datos del GPS** (Véase 3.4.9), adquisición de **Señales de audio y video** (Véase 3.4.11) y **Obtención de variables del robot**. Este bloque tiene el propósito de recabar información acerca del entorno en el que se desplaza el robot así como datos del mismo.

Procesamiento de datos: Las tareas que se agrupan en este bloque son; el **Analisis de Datos**, la resolución del **Modelo Odométrico** (Véase 3.4.2) y el **Procesamiento de video** (Véase 3.5). Estas tareas se agrupan en este bloque dado que su propósito es la adecuación de los datos obtenidos por el robot en datos inteligibles para el operador.

Presentación de datos: Aquí se agrupan las tareas encargadas de proveer al operador los datos en una interfaz, esta etapa se encuentra compuesta por la tarea denominada como: **Interfaz de usuario** (Véase 3.2).

Posicionamiento y Ajustes personalizados: Por último, las tareas contenidas en este bloque son las encargadas de asignar la posición del robot y otras relacionadas con el almacenamiento de los datos tomados para su posterior análisis. Estas tareas son: **Lectura de Joypad**, **Asignación de velocidades de las ruedas** (Véase 3.3) y **Personalización de ajustes**.

Cada una de estas actividades se describen en las secciones siguientes.

[1] Con este nombre se define al conjunto de robot y la computadora para su operación.

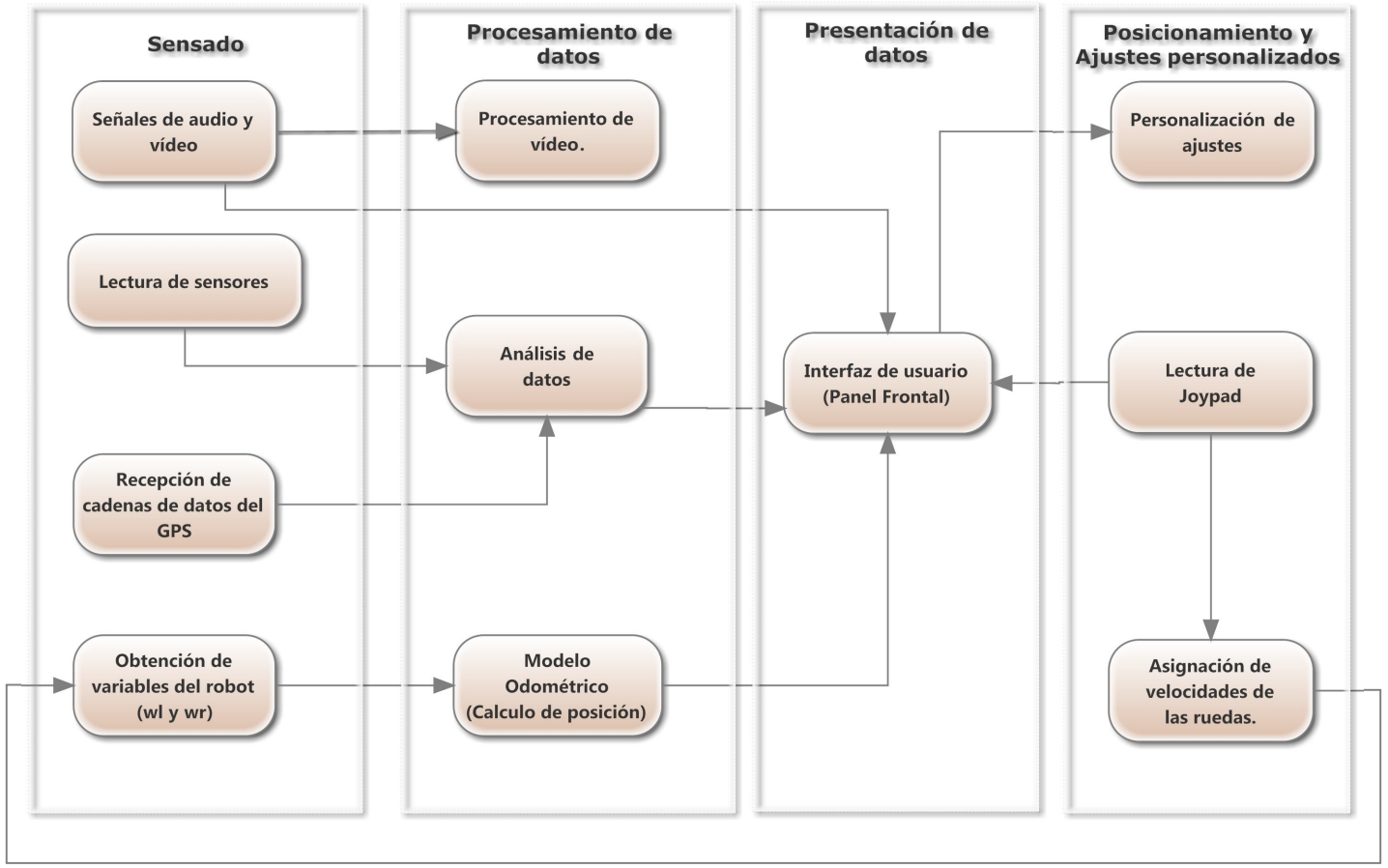


Figura 3.2 Diagrama de bloques detallado del Sistema de Exploración y Monitoreo

3.2 Panel frontal

Uno de los elementos más importantes para el desarrollo de aplicaciones de este tipo es la interfaz que emplea el operador para el control y la visualización de los datos recabados, por lo que en esta sección se explica el funcionamiento del panel frontal en el que se presentan los datos y se implementan algunas instrucciones para el robot.

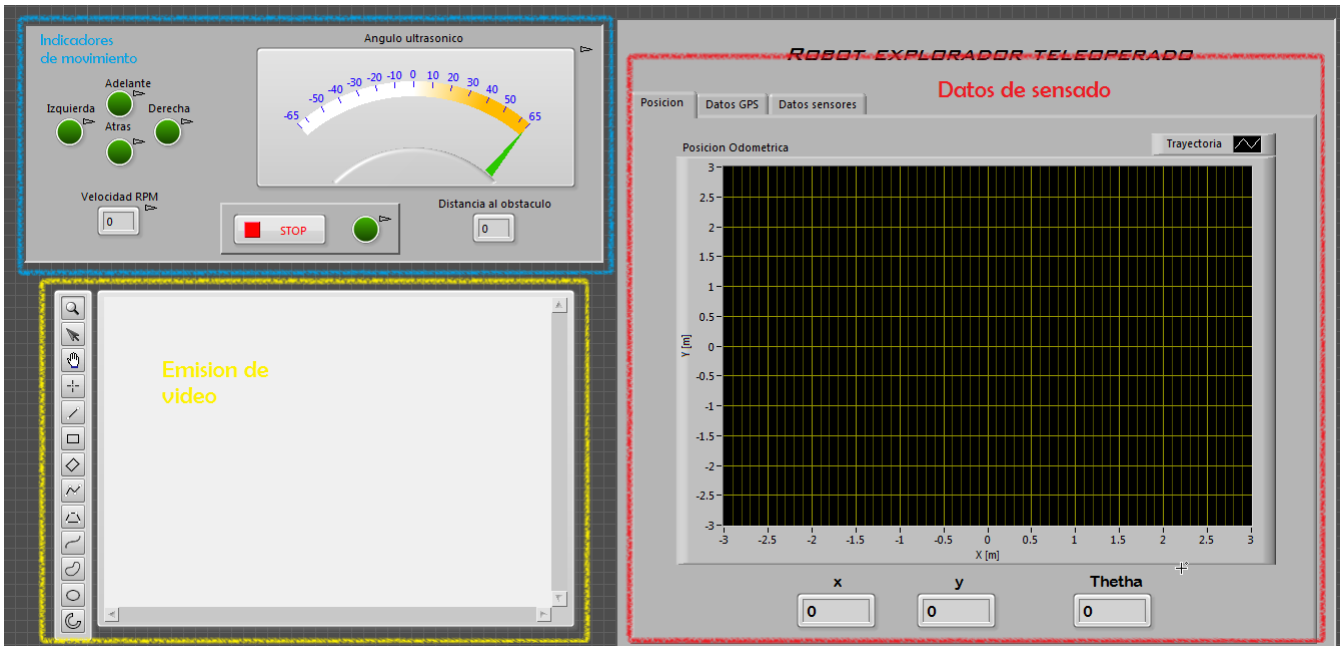


Figura 3.3 Panel frontal para modo teleoperado

En la Figura 3.3 se presenta la interfaz de usuario principal para la operación del robot, ésta se puede dividir en tres secciones principales:

- **Indicadores de movimiento** (Parte superior izquierda): En esta sección se presentan cuatro indicadores tipo led que cambian de estado en el momento en que el operador ejecuta las instrucciones de desplazamiento y orientación del robot que llevan ese nombre en el Joypad. A lado de estas se muestran indicadores que presentan información del sensor ultrasónico, como lo son el ángulo en que se posiciona el sensor (Véase sección 2.3.4) y la distancia a la que se encuentra el objeto en donde rebota la señal ultrasónica. En la parte inferior del indicador de posicionamiento del sensor ultrasónico se encuentra un botón de paro que detiene la ejecución del programa y el movimiento del robot.

- **Presentación de video** (Parte inferior izquierda): En esta sección se dispone de un indicador para la visualización de las señales de video recibidas.

- **Datos de sensado** (Parte derecha): En esta sección se agrupan los datos recabados por los sensores en tres categorías distintas:
  - ***Posición:*** Esta ventana se definió como parte de la pantalla principal, tiene la tarea de mostrar la posición y orientación del vehículo (estados), además del recorrido realizado por el robot (Véase sección 3.4.2).
  - ***Datos GPS:*** En esta ventana se muestran los datos correspondientes a las lecturas del receptor GPS, su aspecto se puede apreciar en la Figura 3.22.
  - ***Datos sensores:*** Aquí se muestran los datos recibidos por los sensores en el robot, así como las alarmas y configuración que indican el umbral de los sensores; también se le agregó un botón para el almacenamiento de datos. En la Figura 3.4 se aprecia la distribución de estos elementos.

Figura 3.4 Ventana para la visualización de datos de sensado

3.3 Control de instrucciones

La manipulación del robot se implementó con un control de videojuegos conocido como *'Joypad Dual Shock'* que se muestra en la Figura 3.5. Dadas las características que tienen estos dispositivos, no existen restricciones para intercambiarlo con otro del mismo tipo. El Joypad se conecta a cualquier puerto serial de la computadora huésped y LabVIEW permite la obtención de datos de estos dispositivos a través del índice de una lista que genera la misma computadora, la cual numera los dispositivos de juego.

(a) Vista frontal (b) Vista lateral

Figura 3.5 Joypad empleado en el proyecto

Las funciones y botones asociados del Joypad se muestran a continuación, mientras que en la Figura 3.6 se presenta la identificación de los botones:

1.- *Abre programa para procesamiento de vídeo.*
2.- *Abre un navegador con los datos de audio obtenidos.*
3.- *Almacena datos de los sensores.*
4.- *Visualiza posición del robot en Google Maps.*
5.- *Aumenta la velocidad de las ruedas.*
6.- *Aumenta los grados de posición del sensor ultrasónico.*
7.- *Disminuye la velocidad de las ruedas.*
8.- *Disminuye los grados de posición del sensor ultrasónico.*
9.- *No se utiliza.*
10.- *Detiene ejecución del programa.*

- *Joystick Derecho (JD): No se utiliza.*
- *Joystick Izquierdo (JI): No se utiliza.*
- *Botones de dirección:*

  Arriba: *Robot hacia adelante.*
  Abajo: *Robot hacia atrás.*
  Izquierda: *Robot gira en sentido antihorario.*
  Derecha: *Robot gira en sentido horario.*

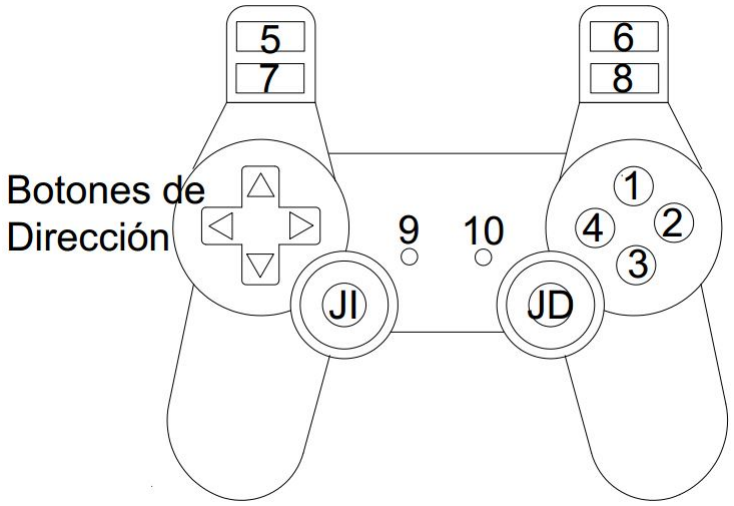


Figura 3.6 Identificacion de botones del Joypad

3.3.1

Elementos del VI

La elaboración del instrumento virtual (Figura 3.7) correspondiente a la lectura de instrucciones del Joypad se basó en las herramientas contenidas en la paleta *Input Device Control*. Éstas se encuentran en *Connectivity/Input Device Control*; las herramientas que aquí se encuentran sirven para la obtención de datos de los siguientes dispositivos: teclado, dispositivos de juego y mouse(ratón).
Como se puede observar el funcionamiento de este instrumento es simple ya que obtiene los datos de los botones y estos se conectan con variables de un VI externo; los datos que se procesan dentro de éste son los límites para la velocidad de las ruedas, ya que no permiten una rapidez negativa; de manera similar también se genera una protección para el servomotor limitando los valores de los grados posibles que se pueden asignar, los cuales se encuentran entre -65° y 65°.

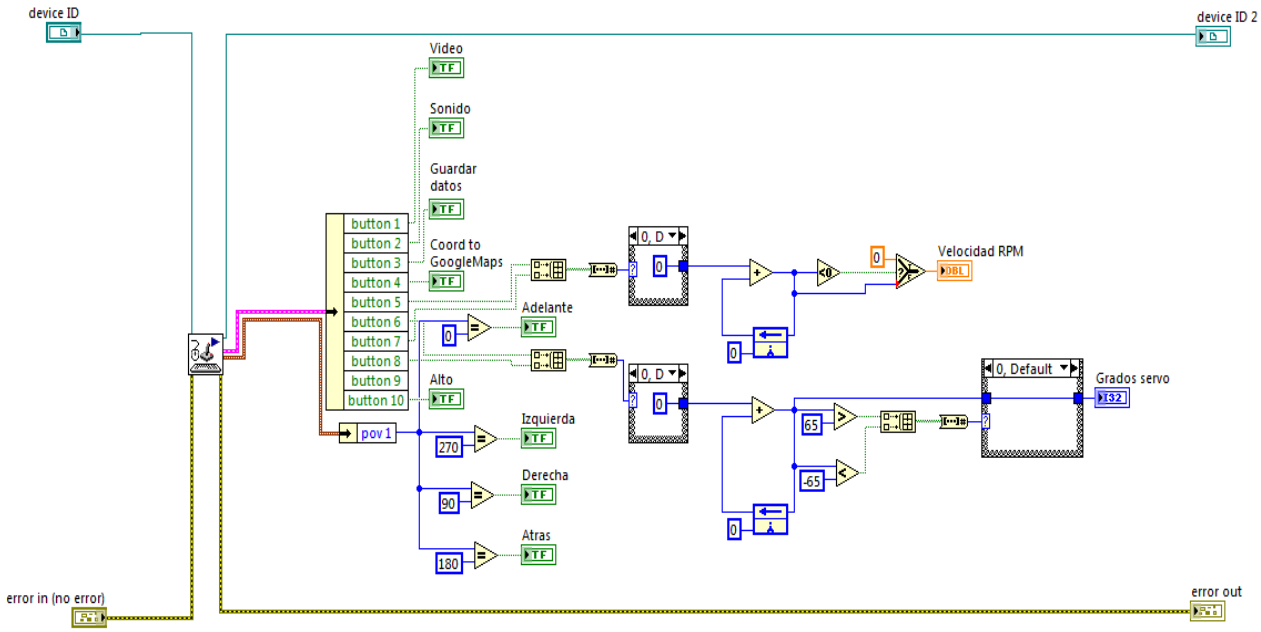


Figura 3.7 Elementos del VI para obtener las instrucciones del Joypad

3.4 Programación de movilidad y datos de sensores

Uno de los objetivos de este robot es la recolección de datos del entorno en el que se encuentra, por lo que el proceso de maniobrabilidad y la forma en que se toman e interpretan los datos son de sumo interés, por consiguiente los métodos empleados se describen a continuación.

3.4.1 Asignación y Lectura de velocidad en las ruedas

Cuando se obtienen los datos para el movimiento del robot que envía el Joypad, éstos se almacenan en *variables compartidas*[2], las cuales se emplean en la asignación de velocidades para cada una de las ruedas. Ésta se lleva a cabo mediante una estructura del tipo *Case*, la cual recibe como argumento un dato del tipo entero que se obtiene mediante la conversión del arreglo de datos booleanos que se forma a partir de las variables que indican el movimiento a realizar por el robot. Los casos posibles son los siguientes:

0. Velocidad cero para ambas ruedas.
1. Velocidad definida por *VRPM*[3] para ambas ruedas.
2. Velocidad en sentido contrario a la definida por *VRPM*.
3. Velocidad cero para ambas ruedas.
4. Velocidad llanta derecha igual a 30 RPM y velocidad llanta izquierda igual a -30 RPM.
5. Velocidad cero para ambas ruedas.
6. Velocidad cero para ambas ruedas.
7. Velocidad cero para ambas ruedas.
8. Velocidad llanta derecha igual a -30 RPM y velocidad llanta izquierda igual a 30 RPM.

Las instrucciones anteriormente descritas, son ejecutadas durante el tiempo que el botón del Joypad se mantenga presionado.
En caso de que ningún botón sea presionado, el caso asignado por defecto es el ”caso cero”, el cual asigna velocidad cero a ambas ruedas. En la Figura 3.8 se muestra el diagrama de bloques correspondiente a la realización de esta tarea.

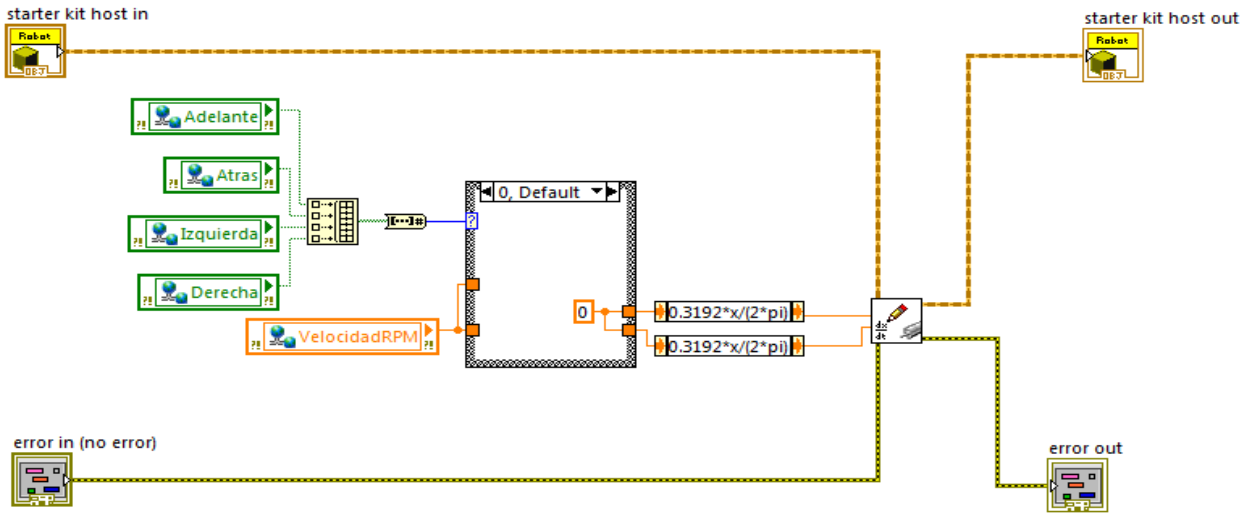


Figura 3.8 Diagrama de bloques para asignación de velocidades

Cuando se decide cuál será la velocidad a asignar, se realiza un escalamiento para transformar la velocidad asignada de revoluciones por minuto (RPM) a radianes por segundo (Rad/s), que es la unidad definida en la clase encargada de asignar la velocidad a los controladores de cada una de las ruedas. El instrumento virtual encargado de las anteriores tareas fue encapsulado en un SubVI con el nombre *Setspeed*. El subVI encargado de las tareas de lectura y escritura de

[2]Es un tipo de variable, la cual puede ser empleada por distintos VI's contenidos en el mismo proyecto aunque se encuentre en dispositivos distintos.
[3]Es la abreviación empleada para la variable compartida VelocidadRPM.

los datos de velocidad, se muestra en la Figura 3.9.

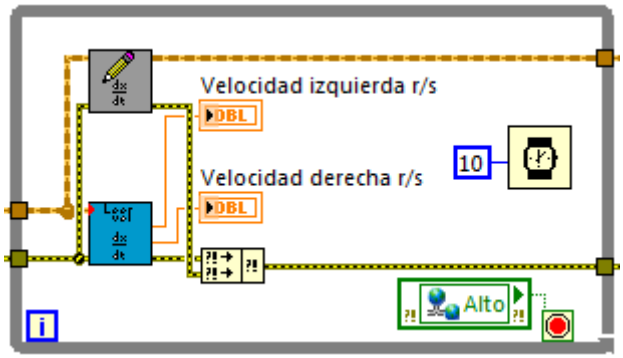


Figura 3.9 Segmento del VI encargado de escritura y lectura de velocidad de las ruedas

3.4.2 Localización del robot

El método que se utilizó para la localización del robot fue el modelo odométrico que emplea la solución de las ecuaciones (2.11). En la Figura 3.10 se observa la forma en que se implementan las ecuaciones en un instrumento virtual. La estructura usada para obtener la solución es un *Control & Simulation Loop* que se encuentra disponible en la paleta *Control Design & Simulation\Simulation*, esta estructura es utilizada en la simulación e implementación de lazos de control.

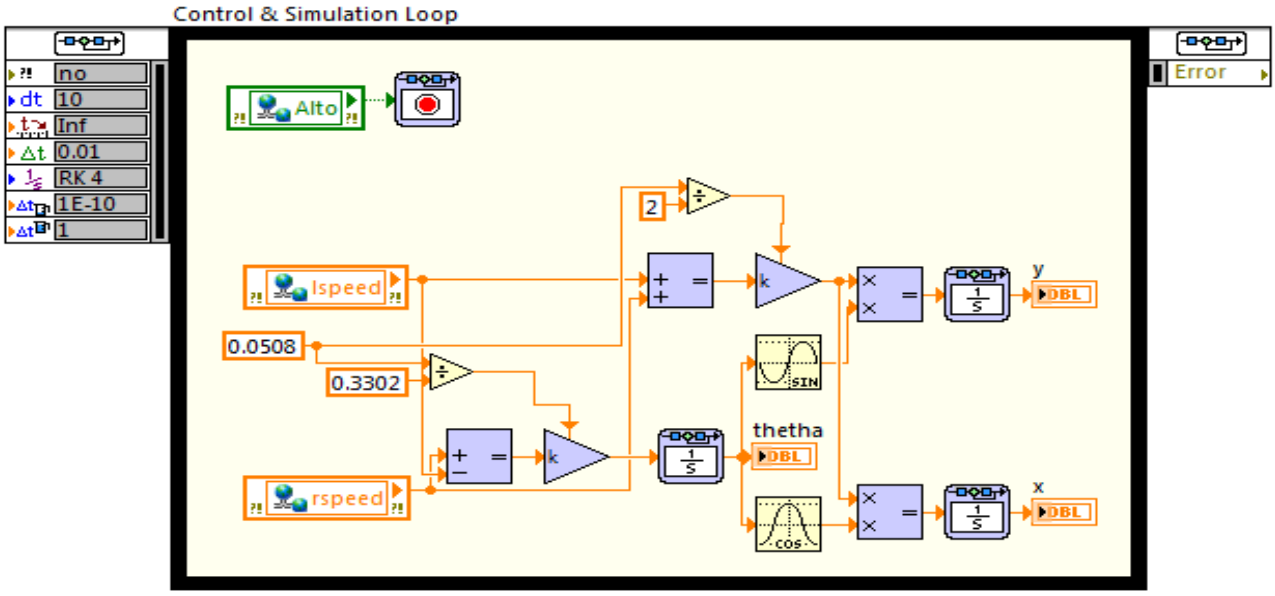


Figura 3.10 Diagrama de bloques para la estimación odométrica

Para el funcionamiento de esta estructura es necesario definir algunos parámetros para su ejecución, los cuales se pueden apreciar en la Figura 3.11.

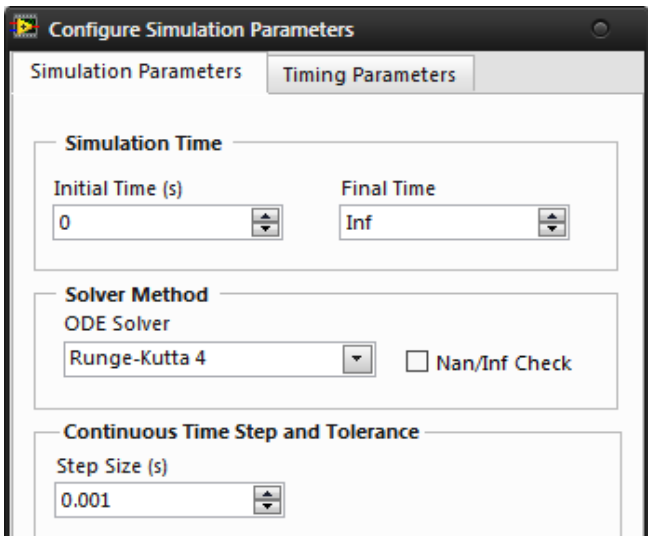
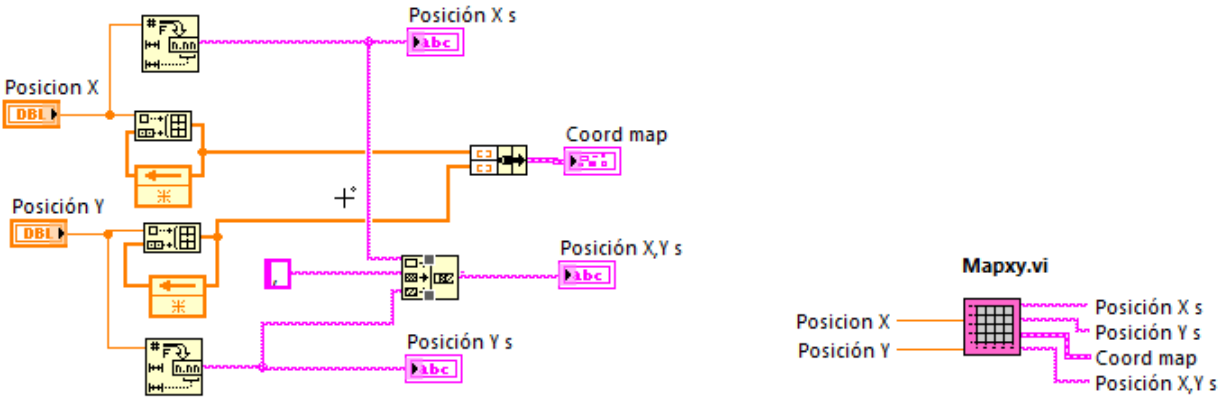


Figura 3.11 Parametros para la estimación odométrica en LabView

En el diagrama de bloques que se muestra en la Figura 3.12 se observa cual es proceso de manipulación que sufren los datos obtenidos de la estimación odométrica, éste consta en la generación de un arreglo que se convierte en un dato de tipo *cluster* que posteriormente se usa para graficar la trayectoria que describe el movimiento del robot.



(a) Diagrama de bloques.

(b) Icono del subVI.

Figura 3.12 VI para la manipulación de datos sobre la posición

3.4.3 Lectura de puertos de la tarjeta sbRIO-9632

La lectura de las señales emitidas por los sensores de calidad del aire, vibración, temperatura y el detector de flama, se lleva a cabo mediante las clases que se encuentran en la paleta *Direct Input and Output*, las cuales están disponibles en: *Robotics\Starter Kit\2.0*. Estas clases permiten el acceso a líneas analógicas y digitales ya definidas; en el caso de las señales digitales se pueden emplear cualquiera de las líneas del puerto tres y la salida *FPGA LED*[4]; y para las señales analógicas se emplean los canales del puerto de entradas analógicas en modo diferencial. El uso de los canales de esta manera ocasiona una reducción en los canales de 32 a 16 [12].
En la Figura 3.13 se muestra el bloque que lleva a cabo la tarea de obtención de lecturas y su posterior almacenamiento en variables compartidas.

[4]Es un led que se encuentra en la parte posterior del robot y es empleado para enviar señales predefinidas por el usuario sobre la ejecución del programa.

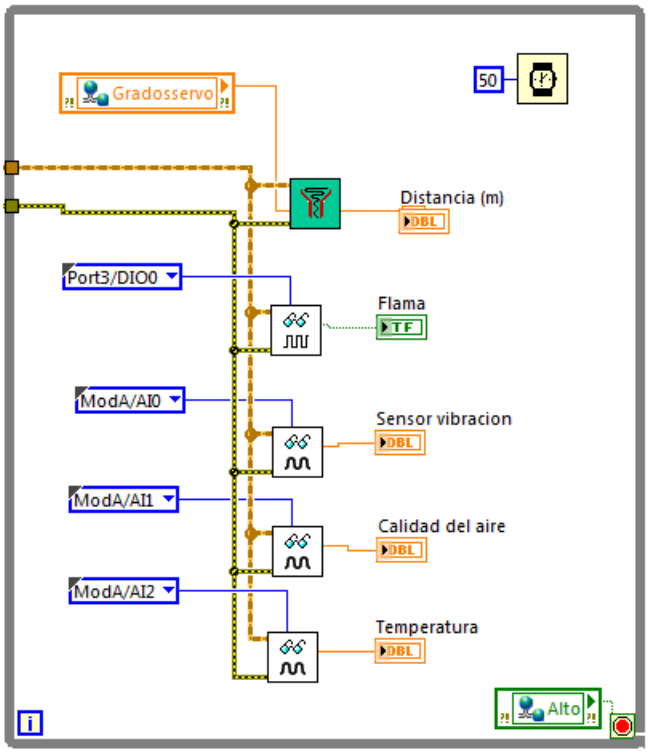


Figura 3.13 Segmento de VI para lectura de puertos

3.4.4

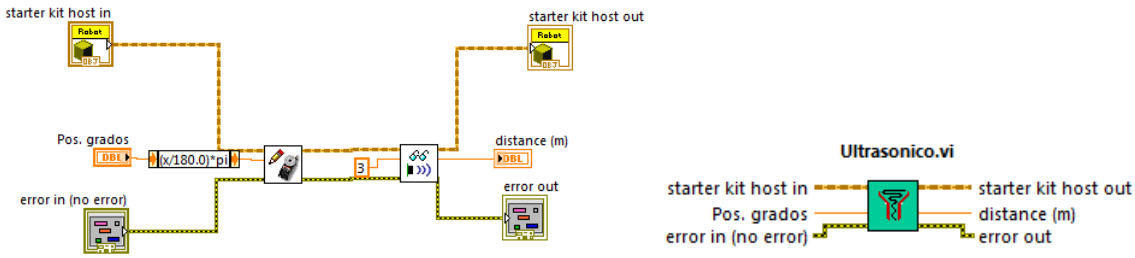
Detector de flama

Los datos emitidos por el módulo detector de flama consisten en señales digitales que indican si alguna flama ha sido detectada, la forma en que se obtiene esta señal es mediante un comparador de voltaje que contiene este módulo, el cual coteja el voltaje emitido por el fototransistor con otro de referencia que es ajustado mediante el potenciómetro del detector. En caso de que la señal emitida sea verdadera, se generará el evento para el almacenamiento de datos (Véase sección 3.4.10).

3.4.5

Sensor ultrasónico

Se desarrolló el VI que se muestra en la Figura 3.14 empleando las clases descritas en el capítulo anterior para la lectura y posicionamiento del sensor ultrasónico. Este instrumento virtual recibe como argumentos la posición en grados con la que se orientará el sensor ultrasónico y retorna la distancia a la que se encuentra el objetivo en que rebota la señal ultrasónica. En caso de que se supere el tiempo de espera de la señal de retorno, el valor emitido por este instrumento será de 3 [m], que es el máximo valor que puede leer el sensor.



(a) Diagrama de bloques.

(b) Panel Frontal.

Figura 3.14 VI encargado del posicionamiento y obtención de distancias del sensor ultrasónico

Sensor de calidad del aire

De la hoja de datos provista por el fabricante se obtuvo el circuito sugerido para la implementación del sensor, el cual se muestra en la Figura 3.15. Dado que este sensor funciona variando su resistencia lo que se obtiene con este circuito es un divisor de voltaje, del cual se obtienen las siguientes relaciones:

$$i = \frac{V_S}{R_S} = \frac{V_c - V_{Out}}{R_s} = \frac{V_{Out}}{R_L}$$

$$R_s = R_L \left(\frac{V_c - V_{Out}}{V_{Out}} \right) \tag{3.1}$$

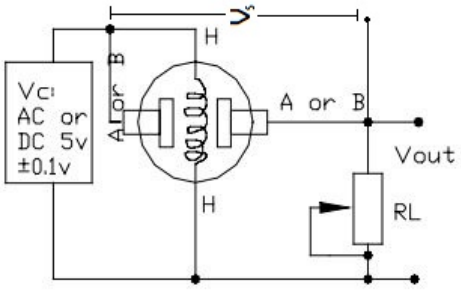


Figura 3.15 [**Circuito para medición de calidad del aire**] Recuperado de: [19]

Con ayuda de la gráfica mostrada en la Figura 3.16, en la que se muestra la relación existente entre las variaciones de resistencia y la concentración de partículas (ppm) para distintos agentes a los que el transductor es sensible, se realizó la calibración del sensor. De esta gráfica se desprende una tercera variable que es R_O, la cual es posible obtener tomando en cuenta la gráfica mencionada y una concentración conocida de partículas de CO_2.

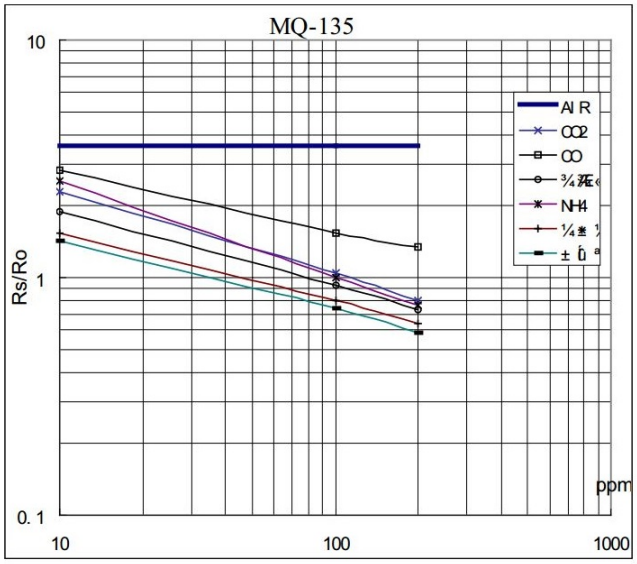


Figura 3.16 [**Gráfica ppm vs RS/RO**] Recuperado de: [19]

En este caso, que no se cuenta con un instrumento calibrado para conocer dicha concentración, se tomó como patrón los datos provistos por la página «*http://co2now.org/*», la cual provee la concentración de CO_2 en la atmósfera. En el momento de consultar la página, la concentración de CO_2 fue de 401.9 ppm. A partir de este dato, se trató de obtener el valor de la resistencia R_O mediante distintos métodos de ajuste, los que obtuvieron una correlación más alta fueron las regresiones potencial y exponencial (Figura 3.17). Dado que la correlación mas alta fue la regresión potencial, se optó emplear ésta curva (3.2):

$$y = ax^b \qquad ppm = 116.9\left(\frac{R_S}{R_O}\right)^{-2.909} \tag{3.2}$$

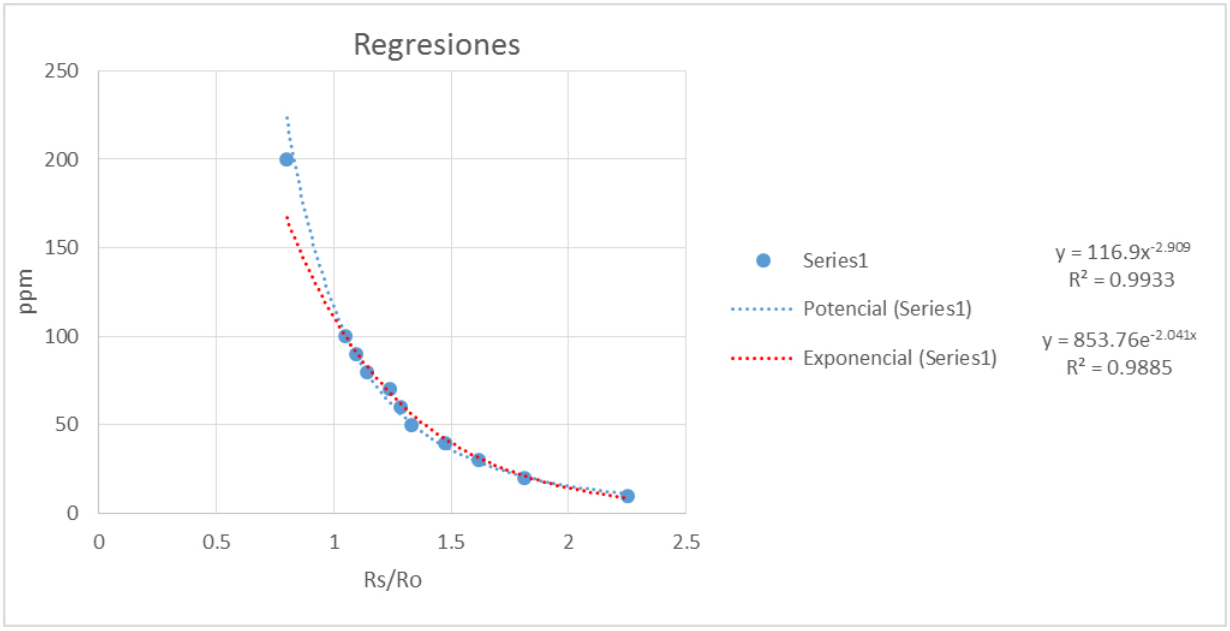
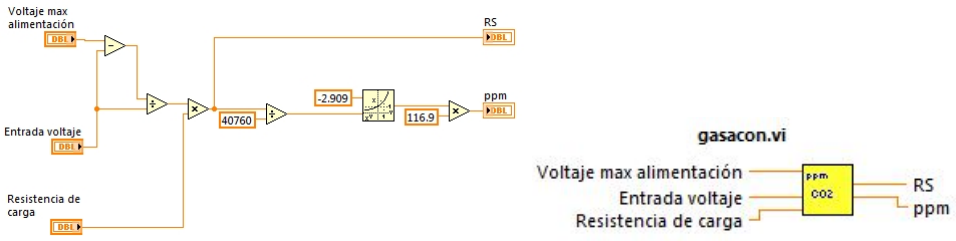


Figura 3.17 Curvas de ajuste RS/RO vs ppm

Empleando esta curva (Ec. (3.2)), se realizó un despeje para obtener el valor R_0 en función de parámetros conocidos:

$$R_O = R_S\sqrt[b]{\frac{a}{ppm}} = R_S e^{\frac{\ln\frac{a}{ppm}}{b}} \tag{3.3}$$

Promediando varias lecturas se obtuvo el valor de la resistencia R_O empleando la Ec. (3.3)a. Para el caso de este sensor la resistencia tiene un valor de R_O=40760.22[Ω] $\approx$ 40[$k\Omega$].
Conociendo estos elementos se realizó el instrumento virtual que se muestra en la Figura 3.18, cuya función principal es realizar el escalamiento para convertir el valor de voltaje en las partes por millón de CO_2 presentes en el ambiente. En la Figura 3.18b se aprecia que este VI recibe como argumentos el voltaje con que es alimentado el sensor, la lectura de voltaje emitida por éste y la resistencia de carga que se utiliza, mientras que los valores que retorna son la resistencia del sensor R_S y las ppm calculadas de CO_2.



(a) Diagrama de bloques. (b) Ícono.

Figura 3.18 VI para el escalamiento del sensor MQ135

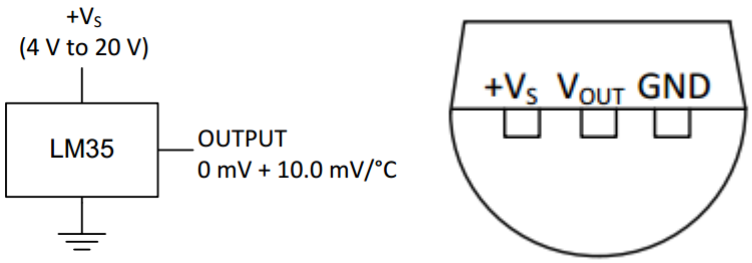
3.4.7 Sensor de vibración

Teniendo en cuenta que para realizar la caracterización de este sensor es necesario contar con equipo especializado, se optó por realizar un escalamiento de la señal para poder lograr su detección. Este proceso se llevó a cabo mediante un amplificador operacional en configuración inversora, los detalles y el circuito empleado se muestran en la sección 3.6

3.4.8

Sensor de temperatura

Como se mencionó anteriormente, la implementación del sensor LM35 es bastante sencilla, ya que sólo es necesario aplicar un voltaje de alimentación entre 4 y 20 VDC para poder obtener una señal de voltaje proporcional a la temperatura. De acuerdo a la hoja de datos del sensor, la relación entre temperatura y voltaje es de 10 $mV/°C$ (dato que se corroboró comparándolo con un termómetro ya calibrado). Debido al rango de temperaturas en las que se encuentra la zona en que el robot se desplazará, se optó por seguir la configuración básica sugerida por el fabricante (Figura 3.19a) con la cual se obtiene un rango de medición entre 2 °C y 150 °C.



(a) Circuito sugerido (b) Vista desde el fondo del sensor

Figura 3.19 Datos de conexión para el sensor LM35

3.4.9

Módulo GPS

En el capítulo anterior se mencionó que para la obtención de datos del receptor GPS, se empleó el paquete VISA. Con las herramientas que contiene se realizó el segmento de programa que se muestra en la Figura 3.20, donde a diferencia del esquema básico que se propone, a éste se le agregó la función de limpiar el *buffer* de almacenamiento al inicio del ciclo de recepción y durante el ingreso de las cadenas.
Para poder realizar la recepción de datos, se ajustaron los parámetros de la configuración serial siguiendo lo mencionado por la norma NMEA 0183 (Véase Anexo E).

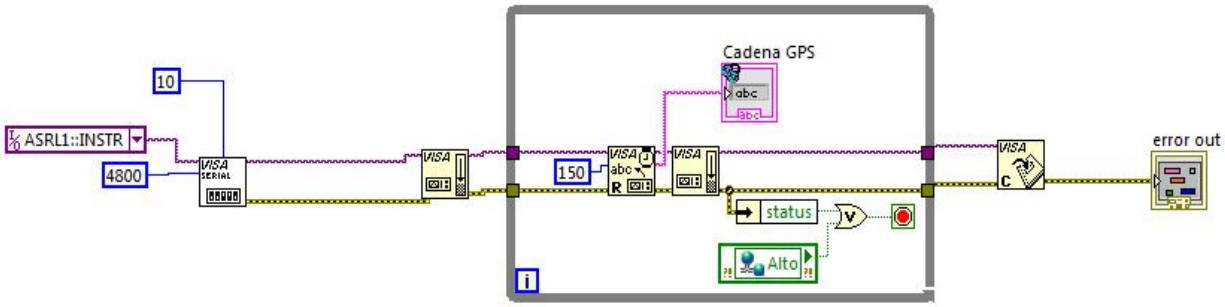
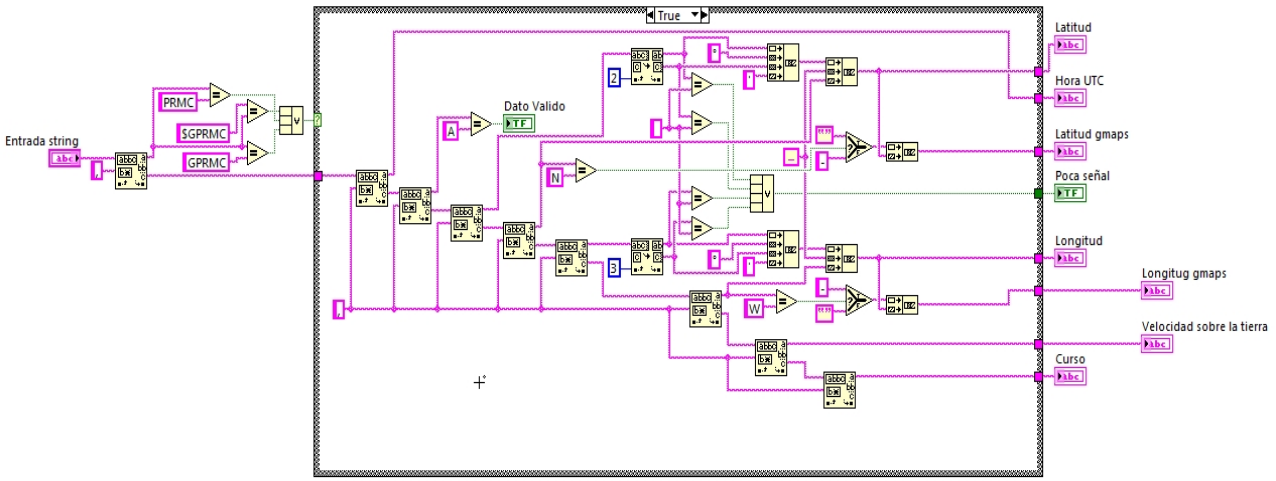


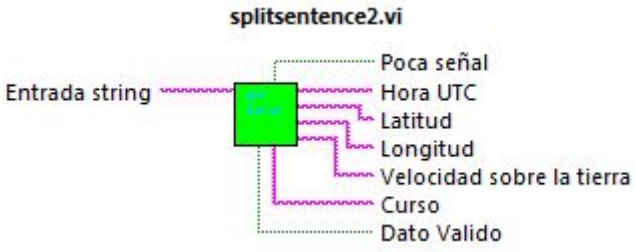
Figura 3.20 Elementos empleados del paquete VISA

Cuando los datos son recibidos por la computadora huésped, la cadena recibida es filtrada con base en los identificadores de oración. En el caso particular de este proyecto, sólo se tomó en cuenta la oración **RMC**, la cual contiene los datos sobre la posición y desplazamiento del receptor. Los datos que se obtienen a partir de esta oración son:

- **Hora UTC**: (Tiempo Coordinado Universal) Es la hora a la que se fijan todos los relojes. Este horario es el que corresponde a la zona por la cruza el meridiano de Greenwich.
- **Latitud**: Es la distancia angular medida en paralelo entre el ecuador a un punto sobre la superficie terrestre.
- **Longitud**: Es la distancia angular medida en paralelo entre el meridiano de Greenwich a un punto sobre la superficie terrestre.
- **Velocidad sobre la tierra**: Velocidad a la que se desplaza el receptor, es medida en nudos.
- **Curso**: Es el azimut que sigue el receptor, se mide en grados sexagesimales.



(a) Diagrama de bloques



(b) Ícono del subVI

Figura 3.21 Instrumento virtual para la interpretación de datos del GPS

- **Dato válido**: Es un dato booleano que indica si los datos de la oración son válidos.

En la Figura 3.22 se muestran los indicadores de los distintos datos obtenidos gracias al receptor; estos datos tienen la posibilidad de almacenarse en un archivo de texto gracias a un botón que se muestra en el panel frontal.

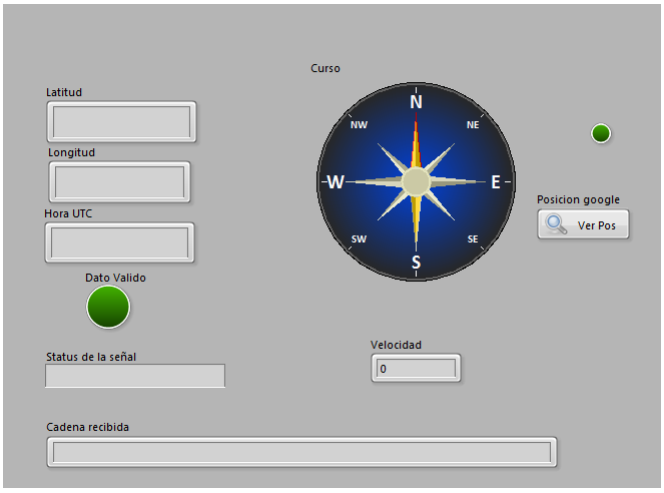


Figura 3.22 Panel frontal de datos del GPS

Como una herramienta para poder visualizar de una forma más específica la ubicación del robot, se creó un subVI encargado de mostrar la posición con base en las coordenadas recibidas por el receptor empleando la aplicación Google Maps. Este subVI es ejecutado cuando se presiona el botón que se encuentra en la parte derecha de la imagen anterior (Botón *Ver Pos*) o presionando el *botón 4* del Joypad (Véase sección 3.3).

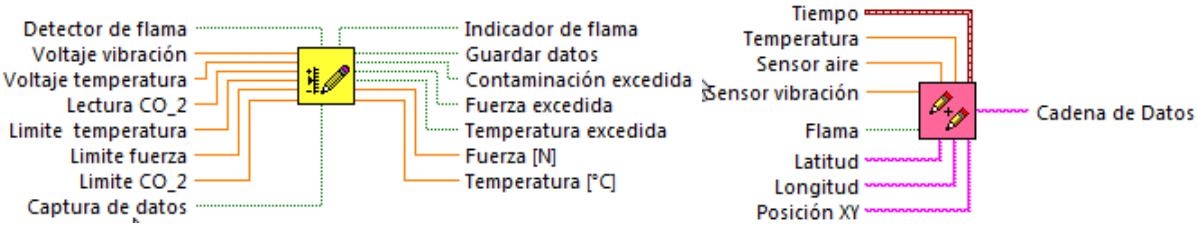
3.4.10

Almacenamiento de Datos

Como una herramienta de apoyo en la tarea de recolección de datos, se implementó una tarea para el almacenamiento de datos obtenidos por los sensores con los que cuenta el robot. Esto se llevó a cabo empleando clases de la paleta *Programming\File I/O*, en ésta se encuentran herramientas para la manipulación de archivos (apertura, cierre y modificación), así que se implementó una tarea que almacena los datos que captura el robot en el momento que se supere el límite del valor predefinido a la lectura de alguno de los sensores o cuando el operador así lo decida.
Se desarrollaron dos VI's para la implementación de esta tarea; el primero de ellos (Figura 3.23a) se emplea para generar las condiciones en las que sucede el evento para la escritura de los datos. El segundo consiste en generar la cadena a escribir en el archivo, esta organiza el orden y formato en que se escribirán los datos, éstos llevan el siguiente orden:

Variable	Fecha y hora	Concentración de CO_2	Magnitud de la vibración	Temperatura	Flama detectada?	Latitud	Longitud	Posición XY 1 (Odométrico)
Tipo de dato	Timestamp	Flotante [ppm]	Flotante [N]	Flotante [°C]	Booleano	Cadena	Cadena	Cadena [m,m]

En la primera fila se muestra lo que representan los datos de cada columna y en la segunda se describe el tipo de dato que se almacena, así como sus unidades.



(a) Icono del VI writetofile.vi (b) Icono del VI concatstring.vi

Figura 3.23 VI's de apoyo para la escritura de archivos

3.4.11

Recepción de datos de video

Como se mencionó en la sección 2.4.7 se optó por implementar como cámara IP, un teléfono celular con sistema operativo Android mediante la ejecución de una aplicación diseñada para este propósito. Cuando el teléfono celular envía los datos, es posible visualizarlos accediendo al servidor usando un navegador web e ingresando a la dirección IP asignada al dispositivo.
Para este proyecto resulta ineficiente acceder a los datos de esta manera, por lo que se halló la solución mediante un adaptador sugerido por el desarrollador de la aplicación, el cual se puede encontrar en la pagina «*http://ip-webcam.appspot.com/*». El adaptador **IP Camera Adapter 2.0** (Figura 3.24), es una aplicación desarrollada para sistemas operativos Windows cuya función es la de simular que la cámara IP esta conectada a la computadora huésped.

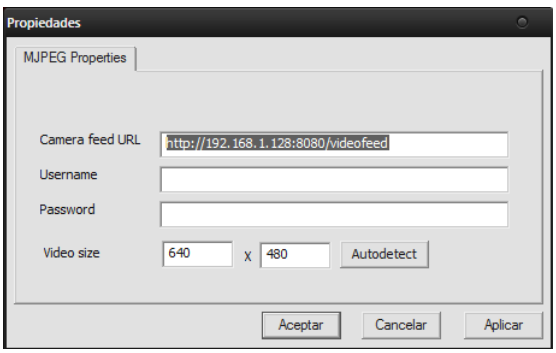


Figura 3.24 IP Camera Adapter 2.0

Para configurar el adaptador es necesario conocer la dirección IP del servidor (en este caso el teléfono celular) y el puerto definido para el envío de datos. Al teléfono se le asignó la dirección 192.168.1.128 (Véase Anexo B) y el puerto para

el envío de datos es el 8080. En caso de haber seleccionado opciones de seguridad dentro de la aplicación, es necesario ingresar el nombre de usuario y la contraseña.

En la sección 3.2 se mencionó que existe un indicador en el panel frontal encargado de la presentación del video. Para su configuración se emplearon elementos de la paleta *Vision and Motion*. En la Figura 3.25 se muestra la estructura básica para la visualización del video, en donde el único dato necesario es conocer el índice que identifica a la cámara en la lista de dispositivos de video detectados por LabVIEW, en este caso le corresponde el índice *'cam1'*.

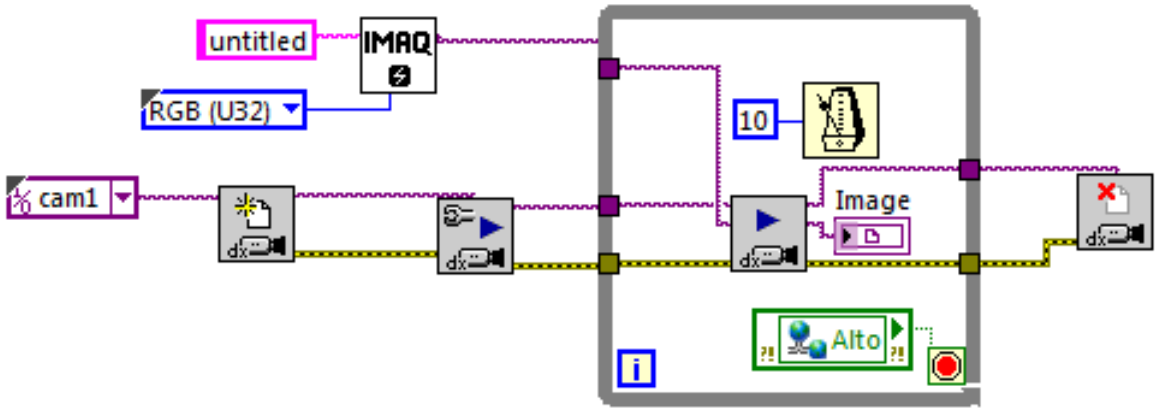


Figura 3.25 Diagrama de bloques para la presentación de vídeo

En el caso de la recepción de audio, se empleó el navegador web con el que cuenta la computadora huésped, por lo que al ingresar a la dirección *http://192.168.1.128:8080/audio.wav* se puede reproducir el audio que es enviado por la aplicación. Para realizar esta tarea se desarrollo un programa en C# que es llamado mediante el Joypad (Véase sección 3.3) y accede a la anterior URL.

3.4.12 Ejecución de aplicaciones externas

La aplicación descrita anteriormente y el programa para el procesamiento de video se realizaron en la plataforma Visual Studio, lo que generó un archivo ejecutable que realiza la tarea descrita. Para su ejecución es necesaria la llamada desde la plataforma para el control del robot, la cual esta realizada en LabVIEW, esto se solucionó con herramientas para la interacción con el sistema de la computadora. En este caso se empleo la clase *System Exec*, que esta contenida en la paleta *Connectivity*. Esta herramienta tiene la funcionalidad de ejecutar la instrucción que recibe como argumento en la consola de comandos de Windows.

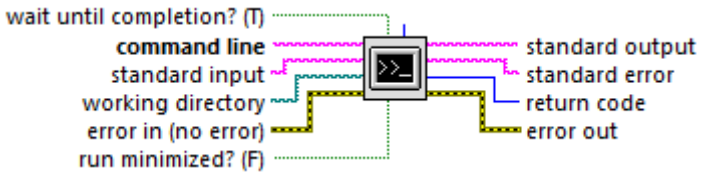


Figura 3.26 Ícono de la herramienta *System Exec.vi*

3.5 Procesamiento de video

Se realizó una aplicación para el procesamiento de las señales de video para servir de apoyo al operador en la obtención de imágenes de interés. Esta aplicación se desarrolló utilizando el lenguaje C# y clases pertenecientes al proyecto Aforge.NET. En esta sección se explica el funcionamiento del programa.

3.5.1 Pantalla principal

En la Figura 3.27 se aprecia la pantalla principal de esta aplicación, desde la cual se selecciona la fuente de video a analizar, el procesamiento que llevará a cabo el video, parámetros para el procesamiento y la señal de video procesada.

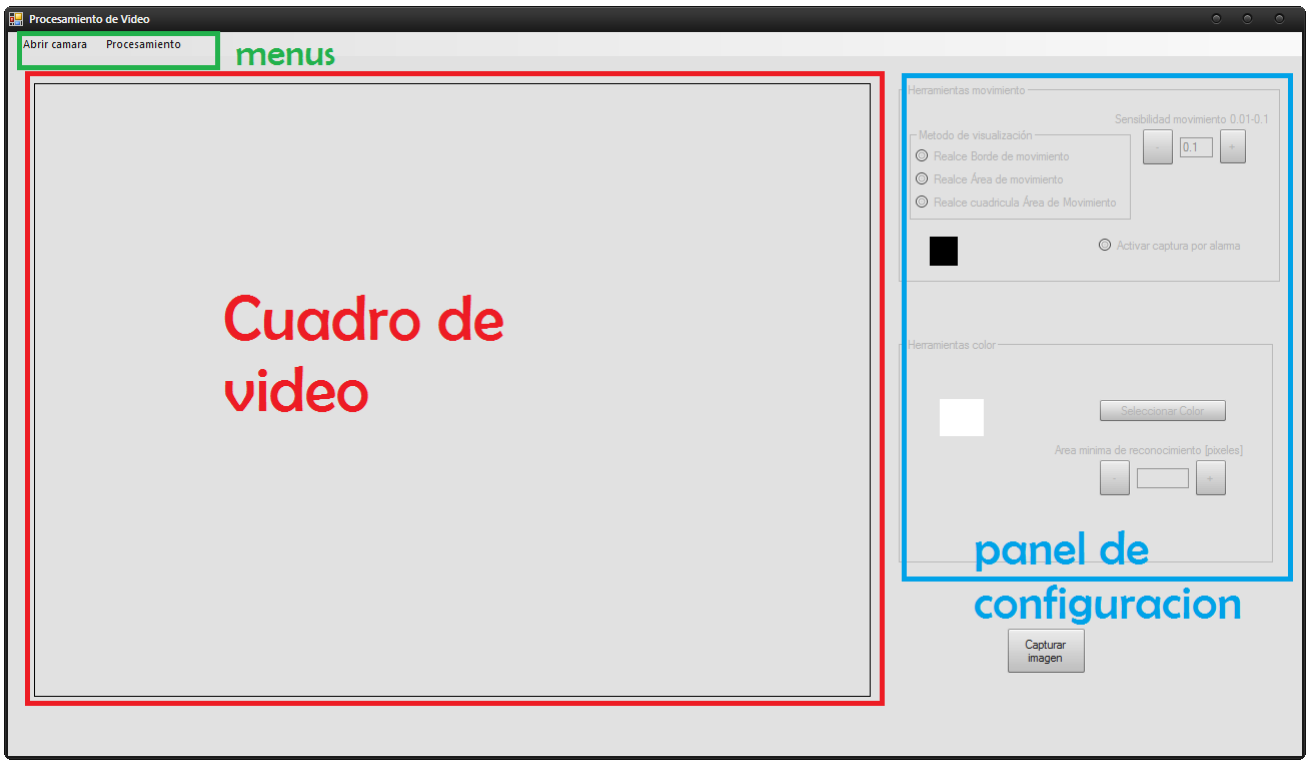


Figura 3.27 Ventana principal del programa para el procesamiento de video

La descripción de los elementos que contiene y su forma de operación, es la siguiente:

Cuadro de video: En esta sección de la ventana se muestra la señal de video procesada. El valor asignado por defecto al seleccionar la fuente de video es mostrar la señal de video recibida.

Panel de configuración: En esta sección se definen los parámetros para emplear los algoritmos de procesamiento, su funcionamiento se detalla en la sección 3.5.4. Al inicio de la aplicación estos elementos se encuentran deshabilitados y se habilitan en el momento de la selección del método de procesamiento.

Menús: A partir de estos menús desplegables es posible seleccionar la fuente de video y el procesamiento que se llevará a cabo (Véase sección 3.5.2).

Botón *"Capturar imagen"* : Cuando se presiona este botón, existen dos posibilidades: en caso de que exista una imagen disponible en el cuadro de video se despliega un nuevo formulario (Véase sección 3.5.5), en caso contrario aparece una ventana emergente que alerta la inexistencia de alguna imagen a capturar.

3.5.2 Menús superiores

Los menús mostrados en la Figura 3.28 agrupan las instrucciones principales para el funcionamiento del programa.

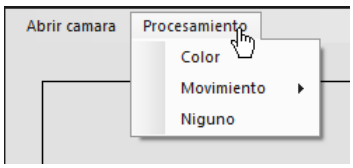


Figura 3.28 Menús de la venta principal

3.5.2.1 Abrir cámara

Cuando se selecciona la opción *Abrir cámara*, se despliega una ventana emergente desde la cual se selecciona la fuente de video. En esta ventana se enlistan los dispositivos de captura con los que cuenta el sistema y es posible seleccionar la resolución en la que se transmitirá el video.

3.5.2.2 Procesamiento

En este menú se agrupan las opciones de procesamiento, éstos son; la detección de movimiento, la detección de algún color y ninguno.

3.5.3 Algoritmos para procesamiento de video

En esta sección se describe el funcionamiento de los algoritmos y los métodos implementados para el procesamiento de las señales de video y la visualización de éste.

3.5.3.1 Detección de movimiento

Los métodos empleados en esta tarea se encuentran contenidos en la biblioteca *AForge.Vision.Motion*, algunas clases se emplean para la detección de movimiento (*Simple Background Modeling Detector* y *Two Frames Difference Detector*), mientras que otras se emplean para observar dicha detección (*Motion Area Highlighting, Motion Border Highlighting* y *Grid Motion Area Processing*). Su funcionamiento es el siguiente:

Simple Background Modeling Detector (Modelado sencillo de fondo): Esta clase implementa un algoritmo de detección de movimiento cuya función es la de calcular la diferencia entre el cuadro actual de video y el cuadro de fondo modelado. La diferencia entre estos cuadros es segmentada y se calcula la cantidad de pixeles contenida en esta segmentación.
Como primera aproximación del cuadro de fondo, se toma el primer cuadro que es recibido durante la transmisión. A lo largo de la ejecución de este algoritmo, el cuadro de fondo es constantemente actualizado, lo que origina un decremento en la diferencia entre el fondo modelado y el cuadro actual.

Two Frames Difference Detector (Diferencia entre dos cuadros): Esta clase implementa el algoritmo más simple para la detección de movimiento, ésto se realiza mediante la comparación de dos cuadros, el actual y el anterior. La diferencia entre estos cuadros es segmentada y se calcula la cantidad de pixeles contenida en esta segmentación.

Grid Motion Area Processing (Procesamiento del Area de la cuadricula de movimiento): Esta clase genera una cuadrícula sobre la imagen a analizar e ilumina aquellos segmentos en los que se detecta movimiento (Figura 3.29a).

Motion Area Highlighting (Resaltar de área de movimiento): El propósito de esta clase es resaltar el área de movimiento, en este caso resalta el área generada por la diferencia entre cuadros (Figura 3.29b).

Motion Border Highlighting (Resaltar borde de movimiento): Tiene el mismo funcionamiento que la clase anterior, pero en lugar de resaltar el área, resalta el borde del segmento (Figura 3.29c).

(a) Grid Motion Area Processing (Procesamiento del Area de la cuadricula de movimiento)

(b) Motion Area Highlighting (Resaltar de área de movimiento)

(c) Motion Border Highlighting (Resaltar borde de movimiento)

Figura 3.29 [Resultados para los distintos algoritmos de visualización para la de detección de movimiento] Recuperado de: *http://www.aforgenet.com/framework/features/motion_detection_2.0.html*

3.5.3.2 Detección de color

En el caso de la detección de color, se emplearon clases contenidas en las bibliotecas: *Aforge.Imaging y AForge.Imaging.Filters*. El funcionamiento de este detector se basa en realizar un filtrado por color de la imagen recibida, a continuación se detectan las áreas de color y sobre la imagen original se crean nuevos objetos (Rectángulos) para delimitar el área donde se ha detectado el color seleccionado. La descripción de las clases empleadas, es la siguiente:

Euclidean Color Filtering Esta clase esta diseñada para filtrar los pixeles de una imagen cuyo color se encuentra dentro/fuera de una esfera RGB con un centro y radio especificado, lo que mantiene a los pixeles dentro/fuera de la esfera especificada y rellena a los demás con algún color específico. Este filtro acepta imágenes en 24 y 32 bpp[5] (bits por pixel) para su procesamiento. Los resultados de la aplicación de este filtro se pueden observar en la Figura 3.30.

(a) Antes del filtro (b) Despues del filtro

Figura 3.30 [Aplicación de la clase *Euclidean Color Filtering*] Recuperado de: *http://www.aforgenet.com/framework/docs/*

[5]**Bits por pixel**: Es la cantidad de información sobre el color de un pixel, en el caso de 24 bits, se dividen en 8 segmentos que corresponden a cada una de las tonalidades principales en cuanto intensidad (RGB). Mientras que en el caso de 32, 24 bits son los descritos anteriormente y se agregan otros ocho para el canal alfa, el cual define el grado de opacidad del pixel.

Blobcounter Esta clase permite el conteo y extracción de objetos aislados en imágenes empleando el *"Algoritmo de etiquetado de componentes conectados"*[6]. Un ejemplo de la utilidad de esta clase se observa en la Figura 3.31.

Figura 3.31 [**Uso de la clase *Blobcounter***] Recuperado de: *http://www.aforgenet.com/framework/samples/image_processing.html*

3.5.4 Panel de configuración

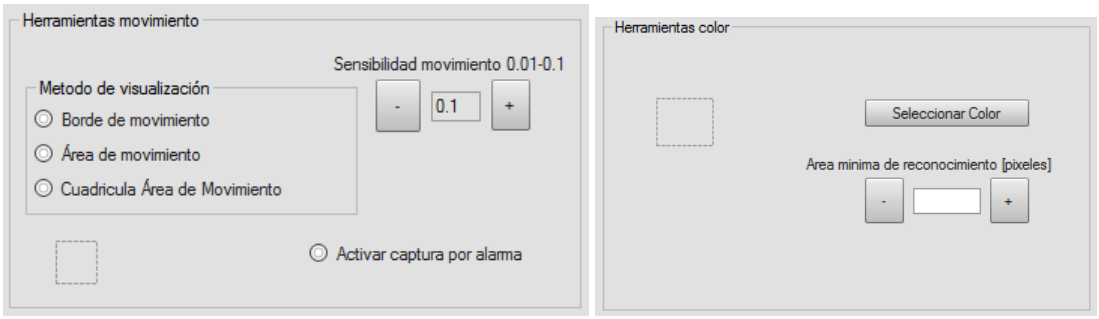
El panel de configuración se encuentra dividido principalmente en dos secciones; cada una de éstas se enfoca en la configuración de las tareas de detección.
En la Figura 3.32a se encuentra el panel de configuración para la detección de movimiento, este bloque puede dividirse en las siguientes dos áreas:

Método de visualización: Cuando se pulsa alguna de las opciones, se define el método de visualización para la detección de movimiento, cada uno de estos botones implementa un método distinto, la relación entre botón y clase es la siguiente:

- *'Borde de movimiento'* →Motion Border Highlighting.
- *'Área de movimiento'* →Motion Area Highlighting.
- *'Cuadricula Área de movimiento'* →Grid Motion Area Processing.

Sensibilidad de movimiento: En este bloque se agrupan los componentes relacionados con el ajuste de sensibilidad de la alarma que detecta el movimiento, los elementos de esta sección son los siguientes:

- *Ajuste de sensibilidad:* Se compone de dos botones que incrementan o disminuyen la sensibilidad con la que se detona la alarma; cuando el valor mostrado es alto, se necesita una mayor *"cantidad de movimiento"* para detonar la alarma. Los valores que puede adquirir esta propiedad se encuentran entre 0.1 y 0.01.
- *Activar captura por movimiento*: En caso de que la alarma se active y esta opción se encuentre habilitada, se generará un evento para la captura de la imagen actual (Véase sección 3.5.5). Después de que este evento suceda, esta opción retorna al valor inicial de desactivado.
- *Indicador de alarma*: Este cuadro muestra oscilaciones cada vez que se detona la alarma de movimiento.



(a) Movimiento (b) Color

Figura 3.32 Bloques del panel de configuración

[6]Este algoritmo convierte todos los pixeles con valores menores o iguales a un color definido como fondo, y los pixeles con valores mayores los define como pixeles de objetos

En la Figura (3.32b), se muestra el bloque correspondiente a las opciones de configuración para la detección de color. Este se encuentra constituido por dos elementos principales:

- *Selección de color*: Cuando este botón es presionado, se despliega una paleta de colores desde la que se elige el color a buscar en el vídeo, por defecto el color seleccionado para la búsqueda es blanco.
- *Área mínima de reconocimiento*: Este elemento engloba los botones para el incremento y decremento de la cantidad mínima de pixeles conjuntos en los que se debe detectar el color para ser reconocido. El valor a seleccionar se encuentra entre 100 y 10000.

3.5.5 Captura de la imagen actual

El evento para la captura de una imagen de interés se realiza mediante un formulario emergente, que es invocado cuando se genera el evento correspondiente (el botón *Capturar imagen* ha sido presionado o la alarma de movimiento se activa). En esta ventana aparece la imagen capturada y un botón que ofrece la posibilidad de almacenar dicha captura en algún directorio de la computadora huésped en los formatos *.jpg, *.png y *.bmp.

3.6 Hardware y conexiones

En esta sección se presentan los diagramas esquemáticos empleados en la realización de las placas de circuito impreso. En la Figura 3.33 se aprecia que para el acondicionamiento de las señales enviadas por el GPS se utilizó el circuito integrado MAX3232, el cual tiene la función de realizar una nivelación de tensión para hacer posible la compatibilidad con el puerto DB-9 (Estandar RS-232) contenido en la tarjeta sbRIO-9632. Para la alimentación de este circuito se agregó un regulador de voltaje a 3.3V.

En la Figura 3.35 se muestra el diagrama esquemático empleado para la conexión de los sensores. Debido a que estos ya contienen los elementos necesarios para su correcto funcionamiento, en esta placa se incluyeron conectores para soporte y conexión entre éstos y los puertos de la tarjeta. El único elemento agregado para el funcionamiento de los sensores consiste en un amplificador operacional LM358 en configuración inversora para el transductor piezoeléctrico.

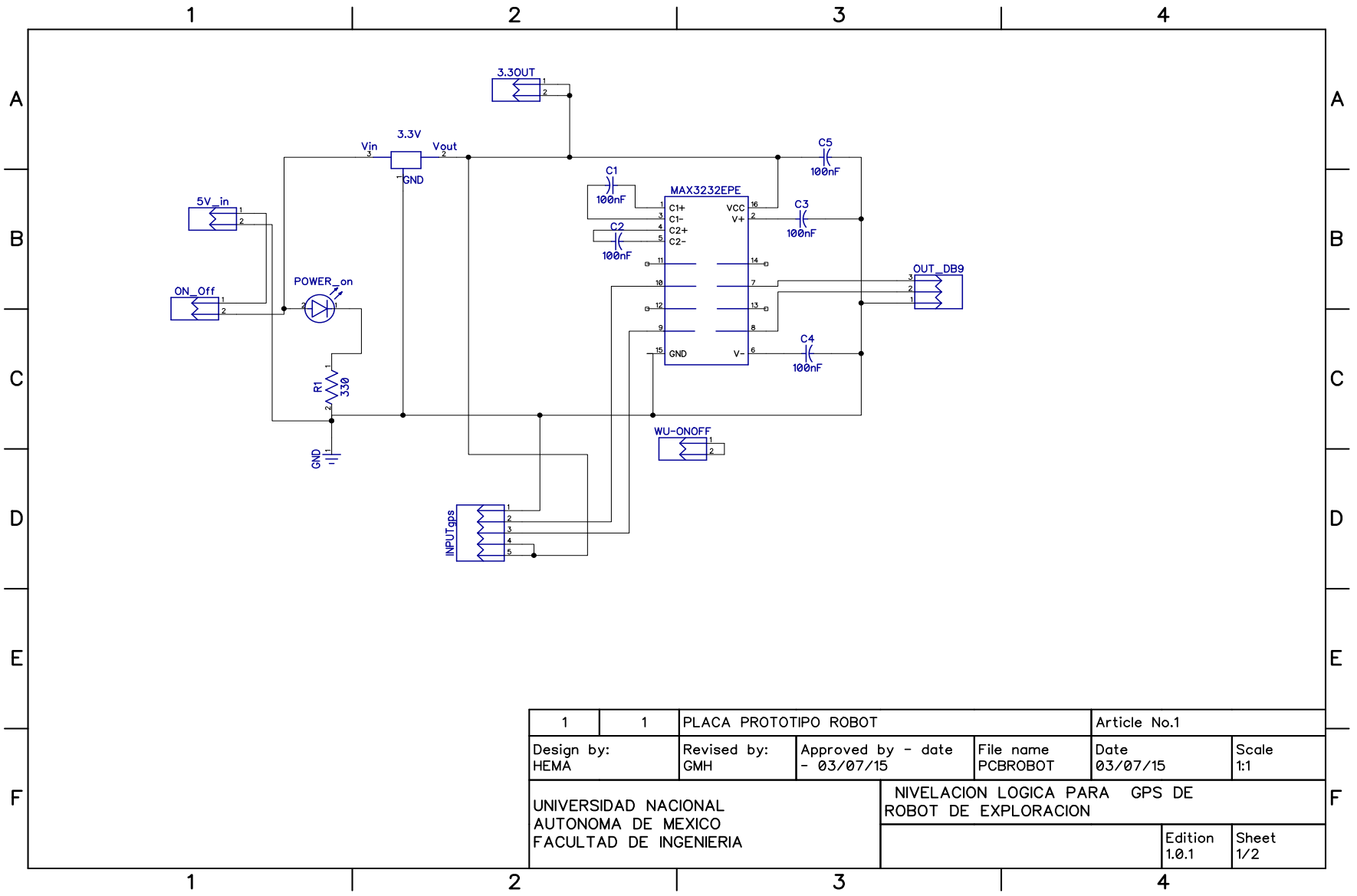
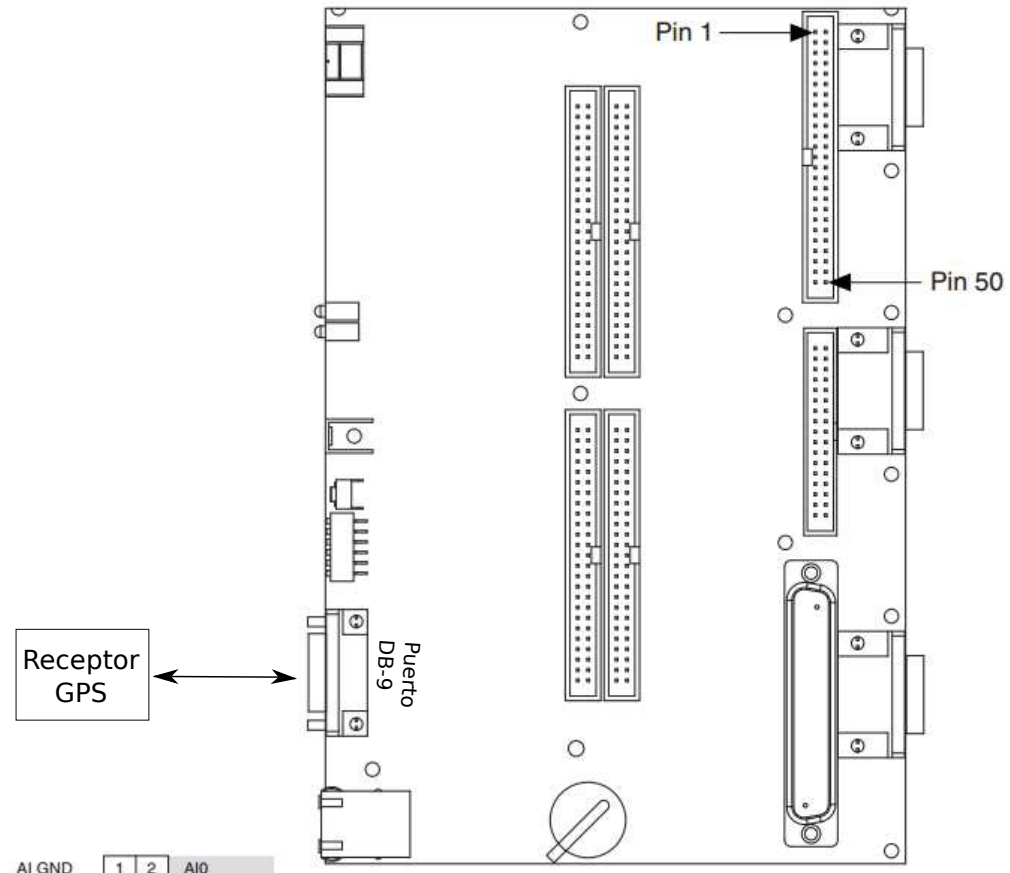


Figura 3.33 Circuito de acondicionamiento para el GPS.

En la figura que se muestra a continuación se aprecia cuales son los puertos de la tarjeta empleados en la conexión de los sensores.



AI GND	1	2	AI0
AI8	3	4	AI9
AI1	5	6	AI GND
AI2	7	8	AI10
AI11	9	10	AI3
AI GND	11	12	AI4
AI12	13	14	AI13
AI5	15	16	AI GND
AI6	17	18	AI14
AI15	19	20	AI7
AI GND	21	22	AI16
AI24	23	24	AI25
AI17	25	26	AI GND
AI18	27	28	AI26
AI27	29	30	AI19
AI GND	31	32	AI20
AI28	33	34	AI29
AI21	35	36	AI GND
AI22	37	38	AI30
AI31	39	40	AI23
AI SENSE	41	42	AO GND
AO3	43	44	AO GND
AO2	45	46	AO GND
AO1	47	48	AO GND
AO0	49	50	AI GND

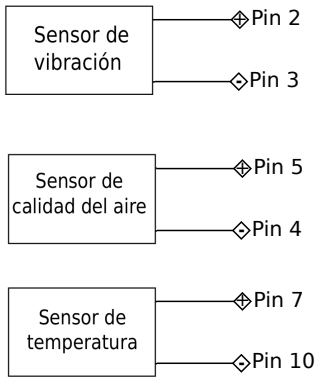


Figura 3.34 Diagrama de conexiones entre sensores y tarjeta.

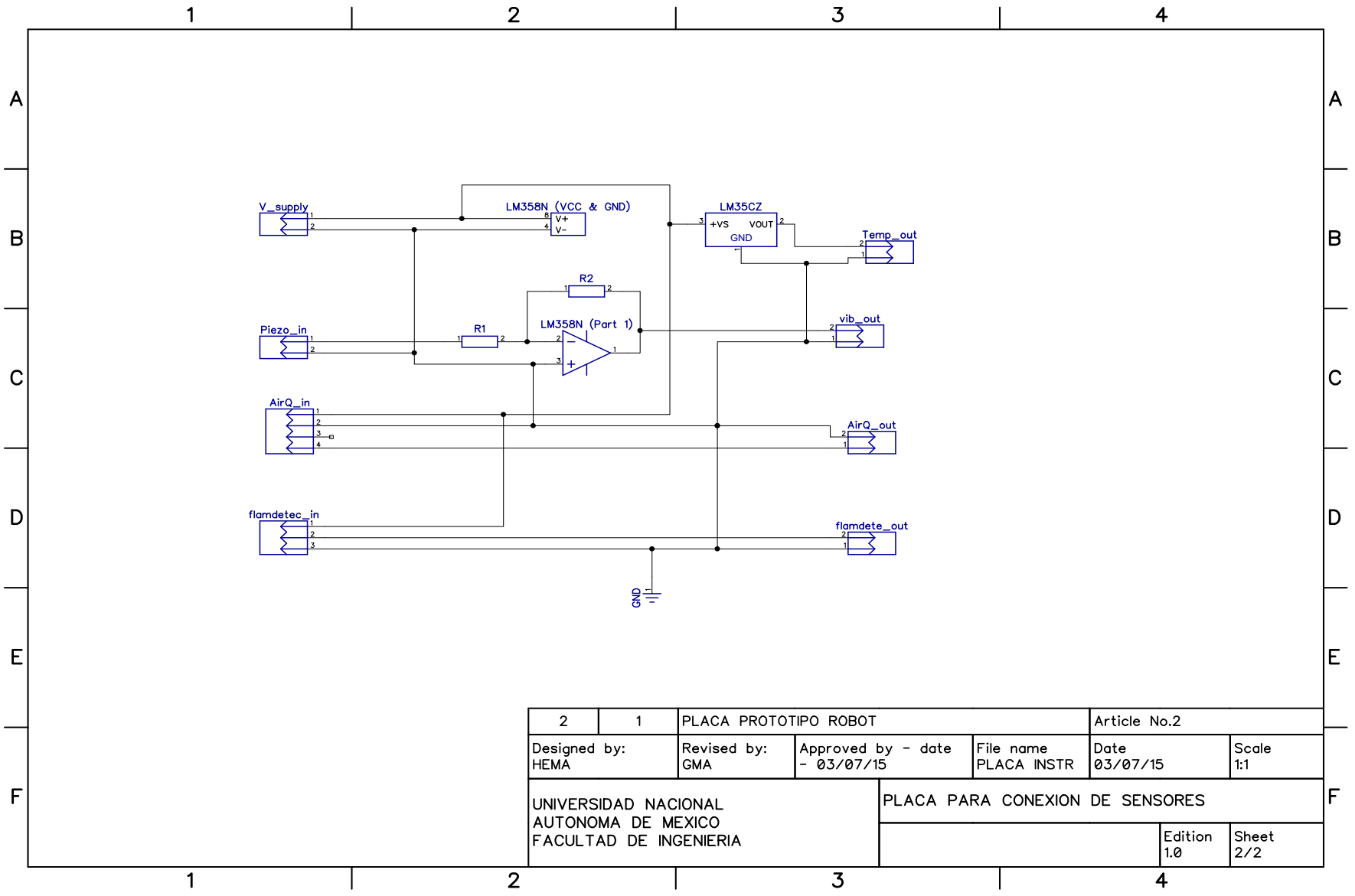


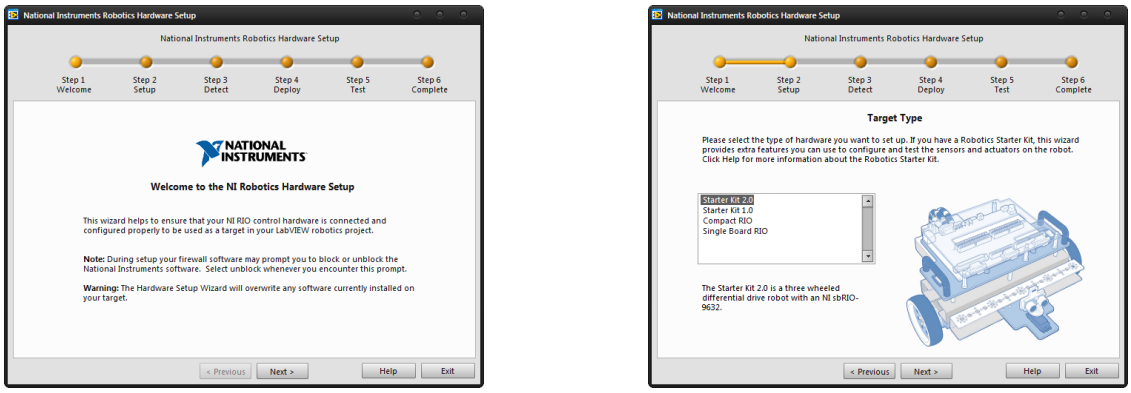
Figura 3.35 Circuito para montar los distintos sensores.

4 Pruebas y resultados

En este capítulo se explican las distintas actividades que se llevaron a cabo para probar el correcto funcionamiento del prototipo. En un principio las pruebas se realizaron con los sensores individuales y posteriormente se verificó su funcionamiento con el software realizado para este propósito.

4.1 Funcionalidad del robot

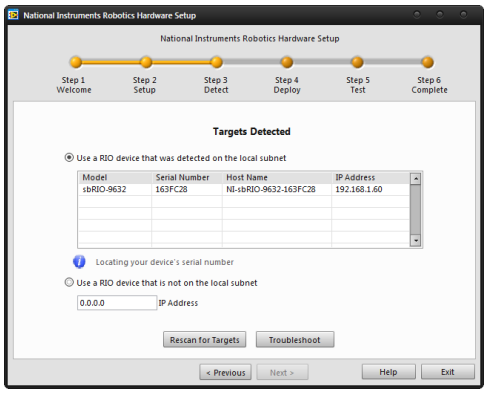
Como un primer acercamiento al robot, se verificó su funcionamiento empleando el programa de demostración que se incluye en el paquete *NI LabVIEW Robotics*, éste consiste en realizar una evasión de obstáculos empleando un método estadístico para la asignación de la trayectoria.
Cuando el robot se utiliza por primera vez, se sugiere utilizar el asistente para la configuración de éste. El proceso de descarga de programas y configuración del robot se describe a continuación.
En la Figura 4.1a se presenta cual es la ventana que se despliega al momento de ingresar al asistente. El siguiente paso consiste en seleccionar la plataforma con la cual se desea trabajar. Este proceso se muestra en la Figura 4.1b y se aprecia que en este caso se selecciona la opción *Starter Kit 2.0* la cual pertenece al robot DaNI.



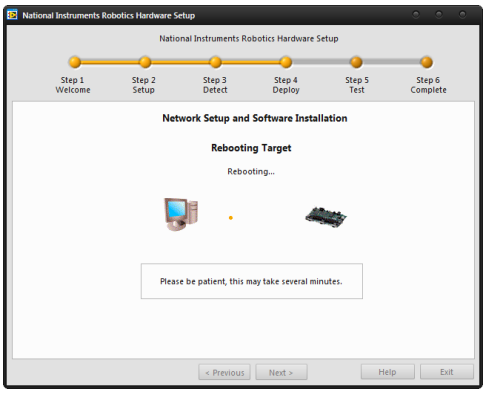
(a) (b)

Figura 4.1 Proceso de ejecución del programa de demostración 1/4

Cuando se selecciona el robot destino, el programa busca dentro de la red local todos los objetivos disponibles (tarjetas sbRIO-9632), éstos son mostrados en una lista como se ve en la Figura 4.2a. En caso de que el sistema no lo detecte es posible encontrar el dispositivo mediante la dirección IP asignada a la plataforma deseada (Véase Anexo B). Posteriormente el asistente proseguirá a la descarga y generación de los archivos para la configuración de DaNI por primera vez como se muestra en la Figura 4.2b.



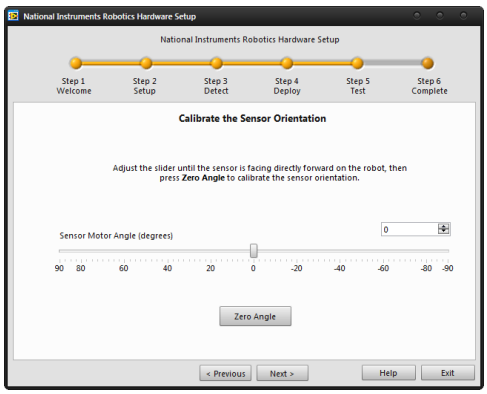
(a)



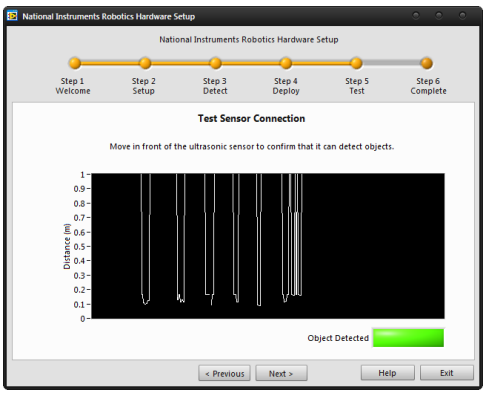
(b)

Figura 4.2 Proceso de ejecución del programa de demostración 2/4

Después de la descarga de estos archivos, el asistente presenta herramientas para la configuración de los sensores y actuadores con los que cuenta el robot. En la Figura 4.3a se muestra la pantalla para la configuración del servomotor con el que se asigna la orientación del sensor ultrasónico. Esta consiste en seleccionar cual será la orientación a partir de la que se considerará el ángulo como ”cero”. El siguiente paso es la comprobación del funcionamiento del sensor ultrasónico, en la Figura 4.3b se aprecia la respuesta del sensor a algunas perturbaciones que se le aplicaron, en ésta se observa la distancia a la que los objetos fueron detectados.



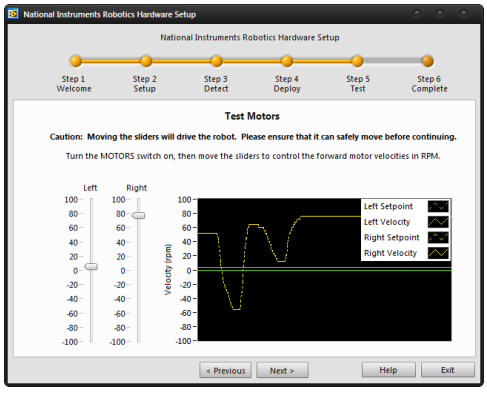
(a)



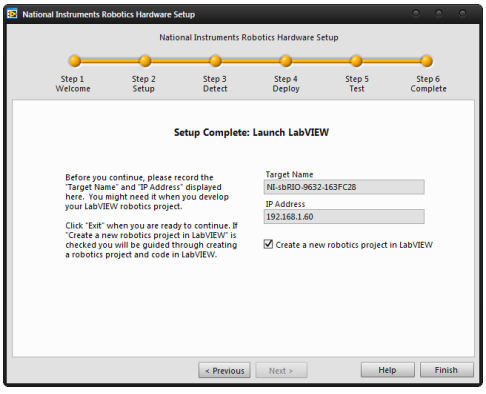
(b)

Figura 4.3 Proceso de ejecución del programa de demostración 3/4

El siguiente paso es la verificación del accionamiento de los motores y su respectiva respuesta mediante la lectura de los *”encoders”*. En la Figura 4.4a se muestra el comportamiento de estas señales. En la siguiente imagen (Figura 4.4b) se muestra la última ventana del asistente, la cual confirma que el proceso de calibración del robot ha sido exitoso.



(a)



(b)

Figura 4.4 Proceso de ejecución del programa de demostración 4/4

Al terminar con la configuración de DaNI el asistente genera un proyecto, el cual contiene el programa muestra para el robot. En la Figura 4.5 se presenta la organización de los archivos contenidos en éste.

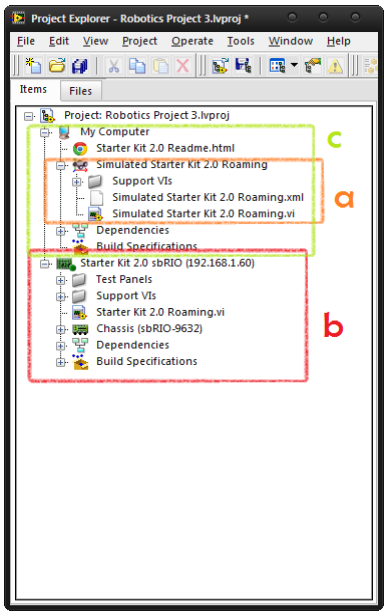


Figura 4.5 Proyecto generado

La descripción de los bloques que se muestran en la figura anterior es:

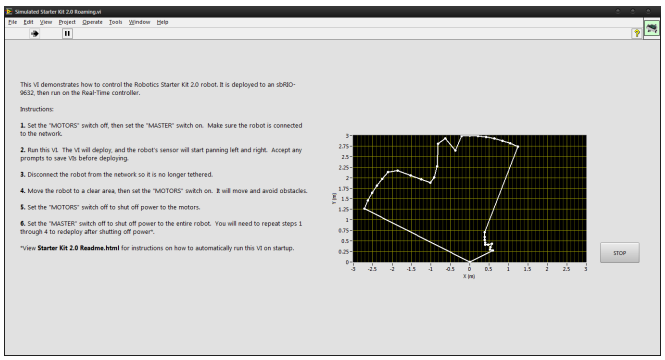
a Este bloque es un ejemplo de la herramienta que contiene el paquete *Robotics*, el cual es un simulador de entornos. En éste es posible probar distintos algoritmos antes de implementarlos en el robot real, cuenta con distintos ambientes y cinco tipos de robots.

b Este bloque muestra los elementos que se desean ejecutar en la tarjeta sbRIO-9632, la cual es la computadora que contiene el robot. Estos programas se realizan en la computadora huésped para posteriormente descargarlos en la tarjeta en el momento de la ejecución.

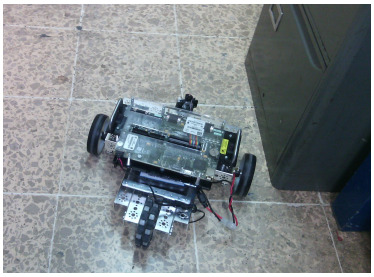
c Este bloque agrupa los elementos que se ejecutarán en la computadora huésped.

La diferencia de agrupar los distintos VI's en diferentes bloques, consiste en que al crearse en la respectiva plataforma destino se acceden a distintas paletas que son propias de ésta. Por ejemplo, en el caso de la tarjeta que contiene DaNI, cuando se crean programas en esta sección, se muestran paletas para la configuración del FPGA; mientras que si se trabaja en la computadora huésped, se muestran paletas para la manipulación de imágenes y audio (por mencionar un ejemplo).

Durante la ejecución del programa se tomaron las imágenes mostradas en la Figura 4.6. La imagen que se encuentra de lado izquierdo pertenece al panel frontal en el cual se muestra el barrido que realiza el sensor ultrasónico y en la parte derecha se encuentra la respuesta del robot.



(a)



(b)

Figura 4.6 Prueba del programa muestra de DaNI

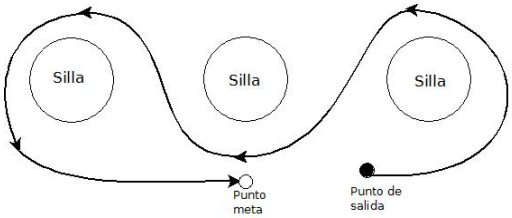
4.2 Prueba de modelo odométrico

Teniendo en cuenta los resultados obtenidos anteriormente en [18] con el mismo equipo, se probó la respuesta de la estimación ante un comportamiento teleoperado.
Se realizaron distintas pruebas para poder estimar la efectividad del modelo, las cuales se describen a continuación: La primera prueba consistió en seguir una trayectoria rectangular que es la que se muestra en la Figura 4.7a.

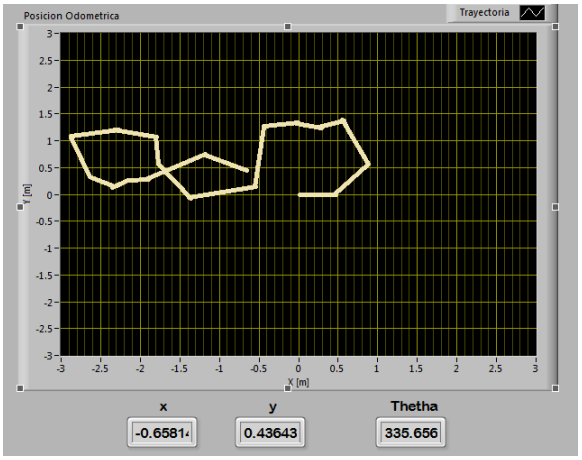
Figura 4.7 Prueba 1: Trayectoria deseada

Durante esta primera prueba se variaron los parámetros de la resolución de las ecuaciones del sistema, el tiempo de paso (*Step size*) se cambió de $0.001\ [s]$ a $0.1\ [s]$ y el periodo de resolución de la ecuación diferencial se ajustó a $100[ms]$. Con estas variaciones, se logró una mejor aproximación de la trayectoria.

La segunda y tercera pruebas consistieron en alcanzar un punto deseado mediante trayectorias "más libres". La segunda se basó en esquivar obstáculos (en esta caso sillas) siguiendo la trayectoria que se muestra en la Figura 4.8a, de esta prueba se obtuvo el resultado mostrado en la Figura 4.8b.
Por último, en el tercer recorrido se probó resolviendo un laberinto, el objetivo de esta prueba era cerciorarse del funcionamiento de la solución del método ante curvas más cerradas. El resultado de esta prueba se observa en la Figura 4.9.

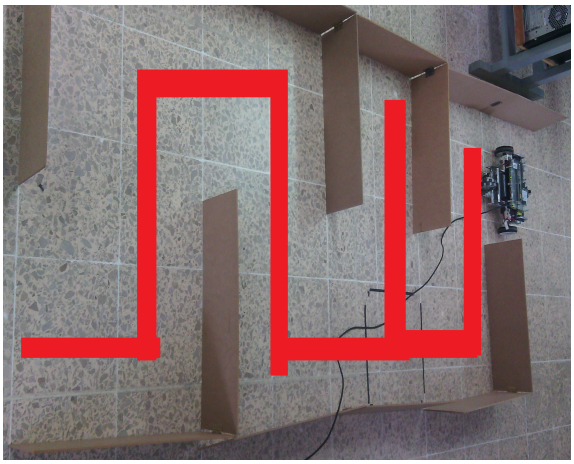


(a) Trayectoria deseada

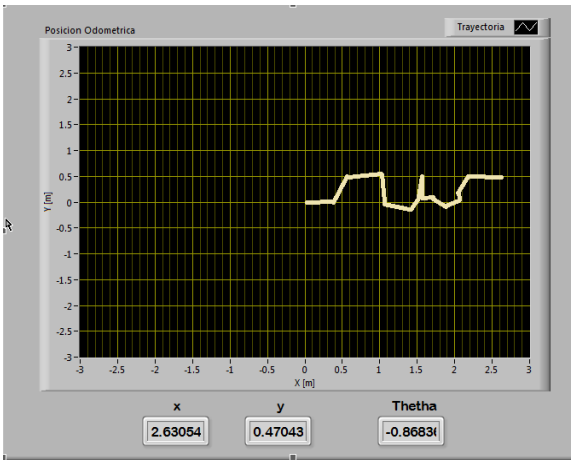


(b) Visualización del método odométrico

Figura 4.8 Prueba 2



(a)



(b)

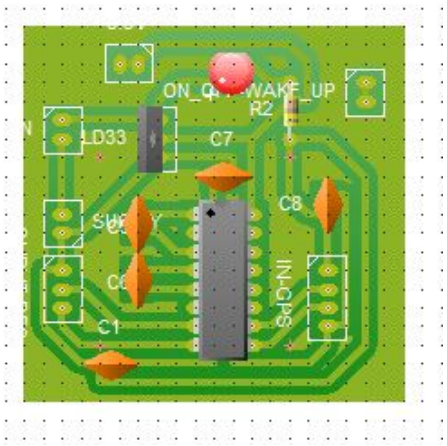
Figura 4.9 Prueba 3

4.3 Prueba de sensores

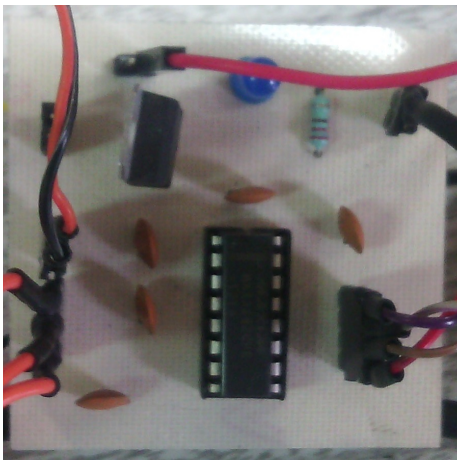
Los sensores fueron probados individualmente en distintas condiciones para tener más información acerca de su funcionamiento y las lecturas que alcanzaban en distintos ambientes.

4.3.1 Módulo GPS

El receptor GPS empleado envía datos siguiendo el protocolo UART mediante un cable USB, por lo que para realizar la interacción entre el módulo y el robot se implementó el circuito que se muestra en el diagrama de la Figura 3.33. Las distintas vistas del circuito con el que se logró el acondicionamiento se presentan en la Figura 4.10.



(a) Modelo virtual



(b) Modelo fisico

Figura 4.10 Circuito de acondicionamiento serial

La prueba del funcionamiento del circuito se llevó a cabo usando el programa NI Measurement and Automation Explorer (MAX), este programa es un complemento de la paquetería de National Instruments, en donde se pueden realizar pruebas del funcionamiento de los dispositivos sin necesidad de realizar una aplicación dedicada, por lo que se configuraron los parámetros dados por el fabricante del receptor y se comprobó la comunicación entre el receptor y DaNI. Esta prueba se muestra en la Figura 4.11.

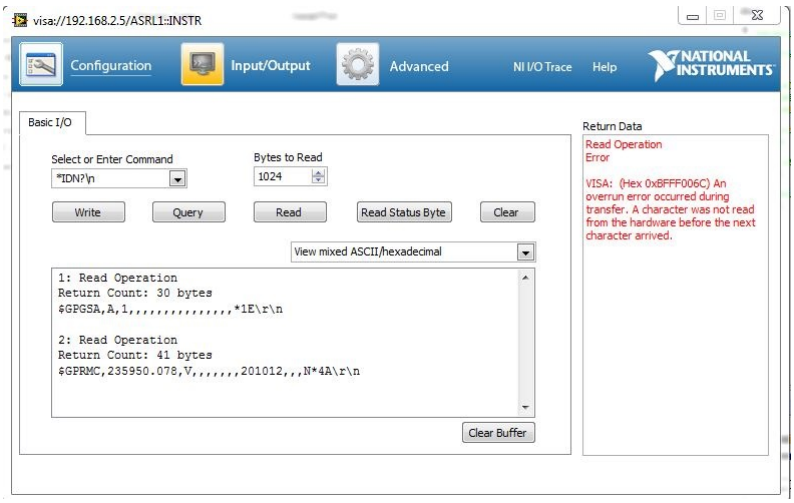
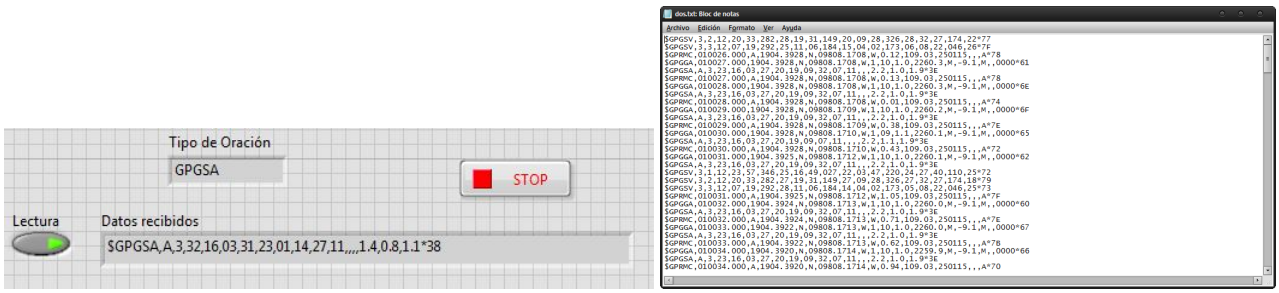


Figura 4.11 Prueba de recepción de datos mediante *NI MAX*

Al asegurar que existe una correcta recepción de datos en el robot, se probó una aplicación diseñada para este propósito que almacena los datos recibidos en un documento de texto plano. Los resultados obtenidos se muestran a continuación:

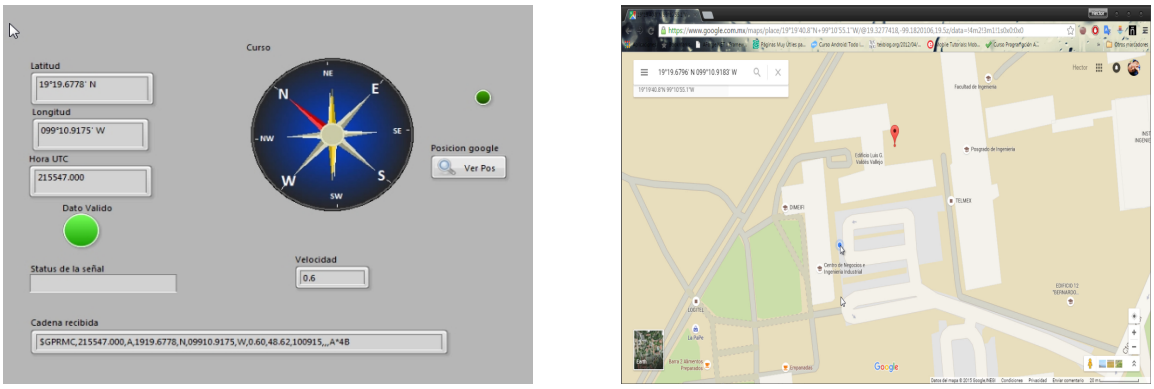


(a) Lectura de cadena en el panel frontal

(b) Archivo de texto con cadenas recibidas

Figura 4.12 Prueba de recepción de cadenas en LabVIEW

Como última prueba para este módulo se verificó el funcionamiento de los VI's cuya tarea es la separación de elementos de las cadenas y la ubicación de las coordenadas provistas por el receptor empleando el sitio de Google Maps. En el caso del primer instrumento, los resultados de la separación de cadenas en datos inteligibles para el usuario se observan en la Figura 4.13, mientras que en la Figura 4.13b se encuentra cuál es la imagen que aparece al ejecutar el instrumento encargado de vincular las coordenadas con el sitio web.



(a) Datos presentados en panel frontal

(b) Posición de las coordenas en Google Maps

Figura 4.13 Visualización datos modulo GPS

4.3.2

Detector de flama

En el caso del detector de flama, que proporciona un valor booleano, se ajustó el valor del potenciómetro con el que cuenta el detector hasta que la señal pasara de *falso* a *verdadero* a una distancia de aproximadamente un metro con una flama como la mostrada en la Figura 4.14. Estos resultados se obtuvieron en una prueba que se realizó en un espacio cerrado con iluminación artificial, mientras que en lugares con menor iluminación se detectó un cambio de señal a una distancia de entre 70 y 60 cm.

Figura 4.14 Prueba con el sensor de flama

4.3.3

Sensor calidad del aire

Para probar la respuesta del sensor de calidad del aire, se realizó el procedimiento descrito en la sección 3.4.6 y se tomaron lecturas de emisiones con distintos orígenes, las cuales se muestran en la Tabla 4.1. Se puede observar que el voltaje aumenta ante una mayor concentración de contaminantes.

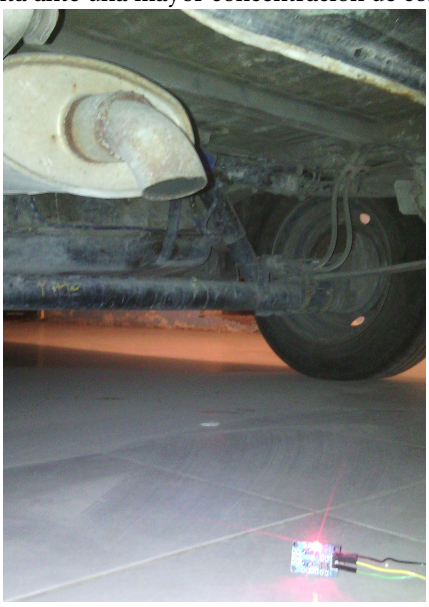


Figura 4.15 Lectura de contaminantes de un tubo de escape

Fuente de emisión	Lectura de voltaje [V]	Concentración de CO_2 [ppm]
Aire libre	0.19	436.19
Humo de cigarro (en el aire)	0.28	1584.47
Humo de cigarro (directo)	1.03	110157.7
Escape de auto (ralentí)	2.18	***2675516***
Escape de auto (acelerado)	3.45	***57920644***

Tabla 4.1 Prueba ante distintas fuentes de contaminación

4.3.4 Detector de vibración

Las pruebas que se realizaron para probar el funcionamiento consistieron en generar vibraciones y analizar el resultado de las lecturas analógicas. Dado que no se cuenta con un instrumento calibrado o un patrón de referencia para realizar la comparación, solo se comprobó la respuesta del sensor a distintas vibraciones, en la Figura 4.16 se muestra la respuesta del sensor ante la caída de una masa de 1 kg en un área próxima al robot, las lecturas mostradas corresponden a caídas de diferentes alturas.

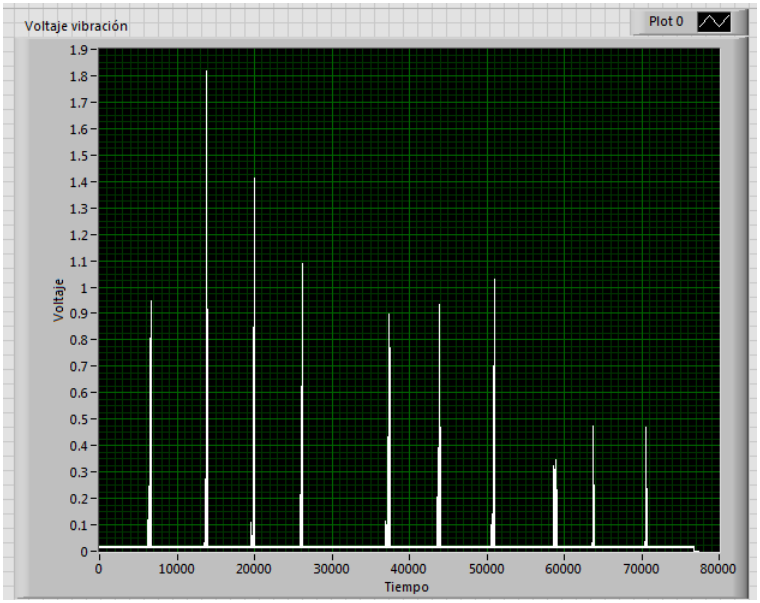


Figura 4.16 Prueba de lecturas del detector de vibración

4.4 Funcionamiento de procesamiento de video

El programa se probó con las distintas cámaras que se encuentran conectadas a la computadora huésped. Al ejecutar la aplicación, el primer paso consiste en seleccionar la fuente de vídeo a emplear mediante la ventana emergente que se presenta en la Figura 4.17. En este caso, las primeras dos opciones corresponden a cámaras web presentes en el equipo, mientras la cámara "**MJPEG Camera**" corresponde al nombre del adaptador para el teléfono celular.

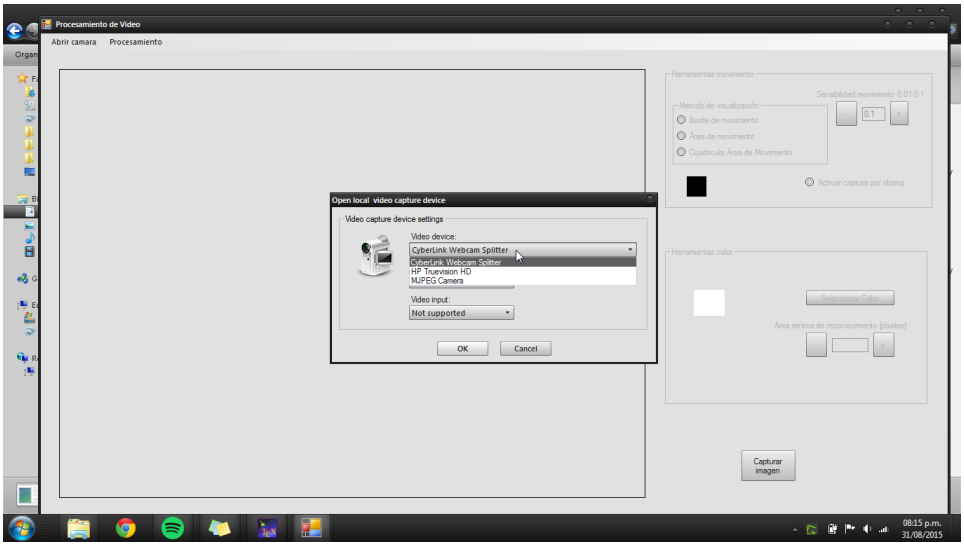


Figura 4.17 Selección de fuente de video.

La respuesta del programa a una entrada de vídeo sin ningún procesamiento se presenta en la Figura 4.18, en esta prueba se variaron las distintas resoluciones y la que presentó mejores resultados fue la de 640x480.

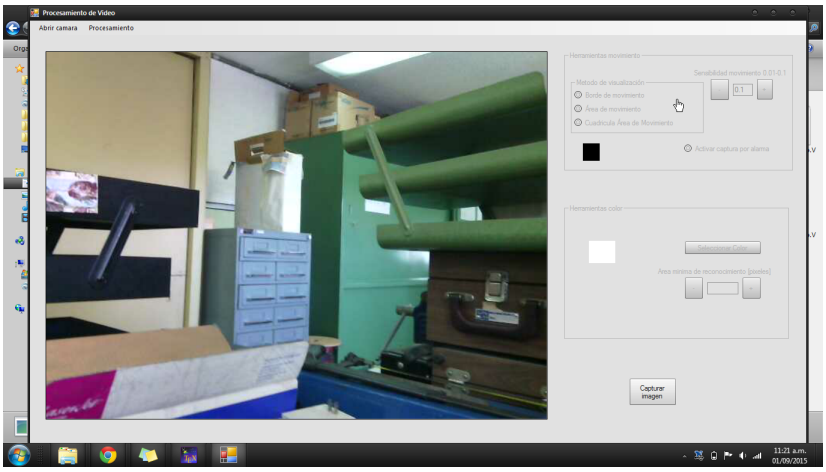
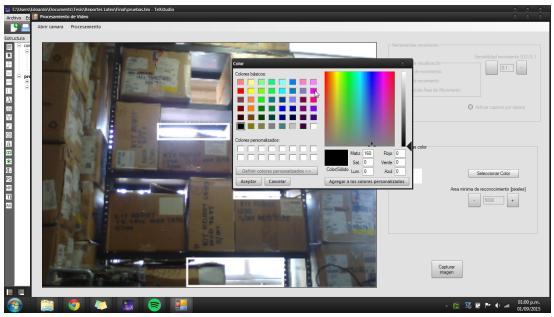


Figura 4.18 Ejemplo de captura de video.

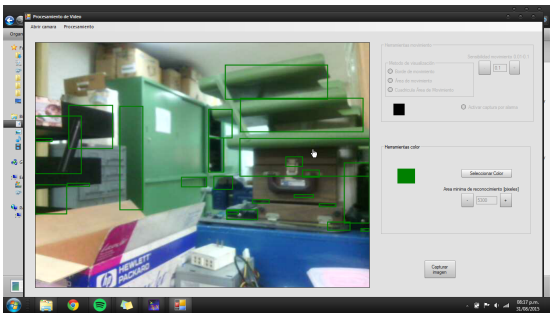
Posterior a la comprobación de la recepción de vídeo, se efectuaron pruebas correspondientes a las distintas funciones para el procesamiento de vídeo.

4.4.1 Detección de color

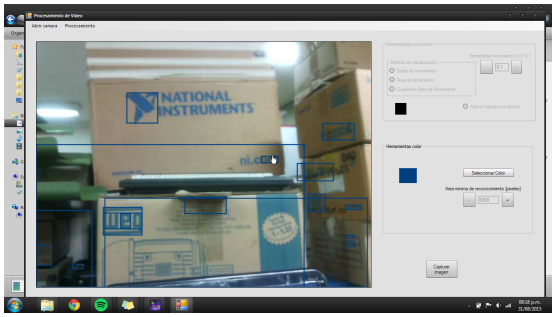
En este apartado se muestra la respuesta a la detección de color. En la Figura 4.19 se muestran las capturas de pantalla que se efectuaron durante la ejecución del programa. Para cambiar el color que se ha de detectar, se presiona el botón *Seleccionar Color* y como respuesta el programa mostrará una ventana emergente desde la cual es posible seleccionar el color a detectar (Figura 4.19a). En caso de que el color deseado para la búsqueda no se encuentre en la paleta básica de esta ventana, es posible ampliar el catálogo de colores añadiendo ”colores personalizados”.



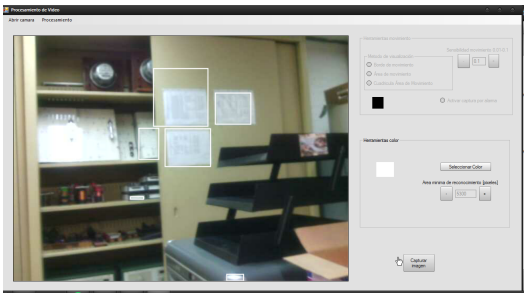
(a) Selección de color



(b) Color verde



(c) Color azul



(d) Color blanco

Figura 4.19 Prueba detección de color

Las imágenes contenidas en las Figuras (4.19b, 4.19c, 4.19d) presentan la respuesta a la detección de color, en cada una se modificó el color y el área mínima de color necesaria para ser enmarcado en la imagen.

4.4.2

Detección de movimiento

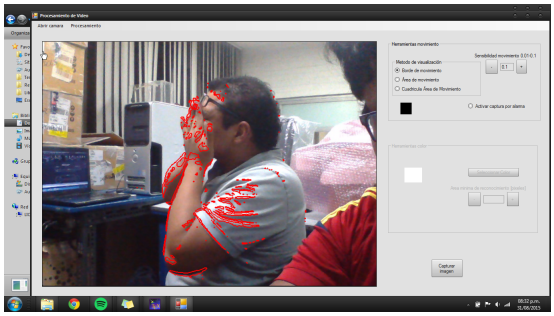
Posteriormente se verificó el funcionamiento de la detección de movimiento mediante los dos tipos de algoritmos implementados en el programa y se observó que ambos tienen distintos tipos de respuesta. Las diferencias encontradas son las siguientes:

Two Frames Difference Detector: Al ser éste el algoritmo más sencillo, presenta una rápida respuesta y cumple con el propósito de indicar el área de movimiento.

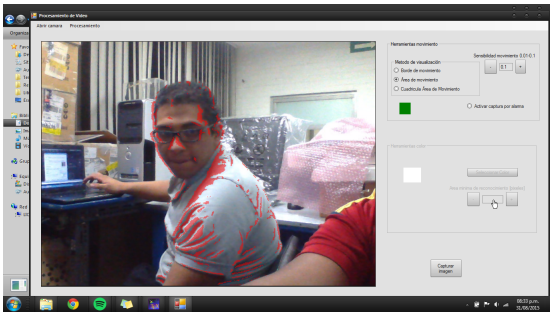
Simple Background Modeling Detector: Este algoritmo tiene un buen funcionamiento, pero debido al proceso de modelado que se emplea para distinguir entre el fondo y el objeto que se mueve, tiene un tiempo de repuesta mayor al primero.

Por otra parte, el primero es mejor cuando se emplea la función de capturar el movimiento en base a la alarma que indica que se ha superado un umbral de nivel de movimiento, esto es debido a que sólo se considera la diferencia entre pixeles para calcular el nivel. En otros casos, donde se requiere una mayor sensibilidad al nivel de movimiento o donde los objetos que se mueven son ”relativamente” pequeños se tuvo una mejor respuesta gracias al proceso de modelado que se lleva a cabo.

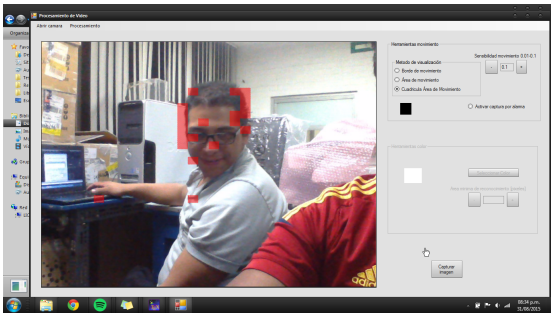
Como última prueba se realizó la verificación de los distintos métodos de visualización de movimiento que se incluyeron en el programa, la respuesta de cada uno de ellos puede ser visible en las capturas de pantalla mostradas en la Figura 4.20.



(a) Motion Border Highlighting



(b) Motion Area Highlighting



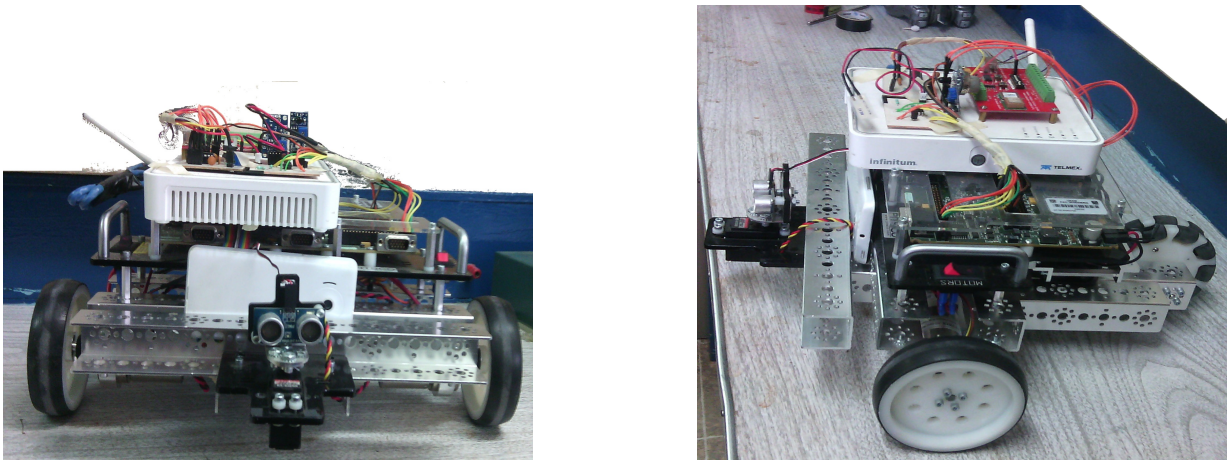
(c) Grid Motion Area Processing

Figura 4.20 Distintos métodos de visualización de movimiento (Véase sección 3.5)

4.5 Operación del robot prototipo

Como última prueba se verificó el funcionamiento de todos los sensores incorporados en el robot. Estas pruebas se realizaron de dos maneras distintas, en las cuales se varió el funcionamiento del robot entre una forma completamente inalámbrica y mediante la conexión con cable Ethernet entre el robot y la computadora huésped.

En estas pruebas se corroboró el funcionamiento de los instrumentos conjuntos en el robot. La prueba se realizó en un ambiente cerrado con el que cuenta el departamento. A partir de esto se realizó un video en el que se detalla el proceso de descarga del programa realizado para este propósito, el funcionamiento de los sensores y las pruebas realizadas al programa para el procesamiento de video. Este puede ser consultado en la siguiente dirección https://youtu.be/GBLYiOX9iXw.



(a) Vista frontal

(b) Vista lateral

Figura 4.21 Imagen del robot con los sensores.

4.6 Otras pruebas

De manera paralela se desarrolló una ley de control para el seguimiento de trayectorias que puede ser consultada en el Anexo A, el motivo de realizar esta aplicación consistía en otorgarle cierto grado de autonomía al robot. Sin embargo, aunque los resultados del seguimiento de trayectorias han sido buenos se optó por no incluir esta rutina en el proyecto principal dado que es necesario definir diferentes trayectorias de exploración, por lo que se decidió enfocarse en el funcionamiento teleoperado, en el cuál el usuario define la trayectoria a seguir.

5

Conclusiones

La idea de desarrollar este proyecto surge ante la posibilidad ofrecida por el departamento de Control de trabajar en un proyecto de robótica móvil dado al auge que ha tenido la investigación e implementación de los robots en distintos campos, por lo que en el desarrollo de esta tarea se utilizó el robot DaNI que es un plataforma para el aprendizaje y enseñanza de la robótica desarrollado por *National Instruments* con el que cuenta el departamento.
Al evaluar cuales podrían ser las tareas del robot, se llegó a la conclusión de realizar un prototipo de robot móvil para tareas de exploración en zonas de difícil acceso, esto como respuesta a la necesidad de adquirir conocimiento en entornos de difícil o imposible acceso para un ser humano.
Este prototipo se realizó con base en la investigación de los distintos tipos de robots para exploración que actualmente se encuentran en el mercado, de los cuales se tomaron sus principales capacidades para poder englobarlas en este prototipo. Esto resultó en un robot teleoperado con las siguientes capacidades:

- Permite obtener su posición mediante dos técnicas (modelo odométrico y posicionamiento satelital).
- Proporciona datos sobre el medio en que se encuentra.
  - Video y audio.
  - Sensado de dióxido de carbono.
  - Detección de flama.
  - Detección de vibraciones.
- Operado mediante un control de videojuegos tipo Joypad.
- Posé una interfaz de usuario para la apreciación y almacenamiento de datos.
- Cuenta con una herramienta de apoyo al operador para el procesamiento de datos de video.

De lo anterior, como primera conclusión es que se cumplió cabalmente con el objetivo propuesto al inicio de este trabajo. La segunda es que este trabajo deja experiencias en cuanto a las posibles mejores a desarrollar en el prototipo, las cuales se explican a continuación:

- El modelo odométrico tiene un buen funcionamiento pero este se encuentra limitado a cortas distancias, ya que el error inherente a este modelo complica su localización en distancias largas debido a la acumulación del error.
- Para un mayor detalle sobre los datos adquiridos, son necesarios sensores con mayores capacidades en lo que a rango y resolución se refiere, así como instrumentos calibrados y patrones de referencia para una adecuada calibración.
- La transferencia de datos entre la red informática se encuentra limitada a las características ofrecidas por el *router* empleado en el proyecto, por lo cual utilizando un instrumento con mejores capacidades de transmisión es posible aumentar la cantidad de datos transmitidos (especialmente vídeo y audio).
- El software empleado como apoyo en la detección de color y movimiento puede ser utilizado en distintas aplicaciones como en sistemas de identificación y vigilancia.
- La configuración del teléfono celular como cámara IP otorga una gran herramienta a bajo costo, comparable con las cámaras dedicadas a este proceso.
- En el caso de emplear un seguimiento de trayectorias la mejor opción consiste en acortar la distancia entre puntos dados al robot para un mejor funcionamiento.

Como resultado de este proyecto se generó un prototipo con características similares a los robots comerciales presentes en el mercado (Véase Tabla 1.1) y deja un trabajo experimental importante para la posterior implementación de un robot que pueda ser usado en zonas de riesgo, este robot puede ser empleado como una herramienta didáctica para el estudio y la enseñanza de la robótica móvil. Con los resultados obtenidos se considera que el objetivo planteado fue cumplido en su totalidad.

A nivel personal, el desarrollo de este proyecto resultó sumamente interesante, ya que para su ejecución se emplearon los conocimientos adquiridos durante mi formación profesional, lo cual confirma que la Facultad provee una formación íntegra para sus estudiantes. A su vez, me permitió entender la complejidad de los sistemas móviles de robótica y me brindó la oportunidad de presentar los avances de el trabajo desarrollado en un congreso.

Trabajo a futuro

Como se mencionó en el Prólogo, la última etapa del proyecto consiste en el robustecimiento del sistema; durante esta etapa se prevé que el robot cumpla con las características necesarias para ser sometido a un ambiente de riesgo. Para lo cuál, es necesario realizar cambios en el robot como lo son: la implementación de sensores e instrumentos con mayores capacidades, el añadir etapas de potencia dedicadas exclusivamente a la alimentación de sensores y ruteador para un comportamiento completamente inalámbrico, además de realizar un cambio por un *router* con una mayor velocidad de transferencia (ya que en el actual la transferencia mediante Wi-Fi esta limitada a 56 Kbps) y menor consumo energético (la alimentación del actual es 22V y un consumo de 0.6 A). De la misma manera para probar la funcionalidad del robot en un ambiente de riesgo, es necesario un cambio de rodamientos para lograr un mejor agarre en la superficie.

Además con el trabajo realizado se prevé el desarrollo de leyes de control con miras a obtener una mejor respuesta, así como la posibilidad de convertir el robot en autónomo al especificarle una tarea en concreto (tomando como base las pruebas documentadas en el Anexo A) y la posible integración de un brazo mecánico. Otra posibilidad es la de la creación de una red VPN para el manejo del robot desde distancias más lejanas. Con estos cambios las características del robot dejarían de ser de un prototipo para pasar a ser un robot de exploración en zonas de riesgo.

ANEXOS

A

Control de posicionamiento de un robot móvil

Dentro de las tareas que se llevaron a cabo, se encuentra el desarrollo de un controlador para el posicionamiento del robot. La realización de este se explica a continuación:
Teniendo al robot en un marco de referencia como el que se muestra en la Figura A.1, en donde se conoce la posición actual del robot $P_R = [x_r \quad y_r \quad \theta_r]$ es posible establecer un punto al que se desea llegar y definiremos como punto meta $P_D = [x_d \quad y_d]^T$, dado que el caso de un robot diferencial, es un sistema no holónomo es imposible realizar un control que modifique la orientación al mismo tiempo que modifica la posición, por lo que aprovechando estas restricciones se desarrollaron las siguientes leyes de control.

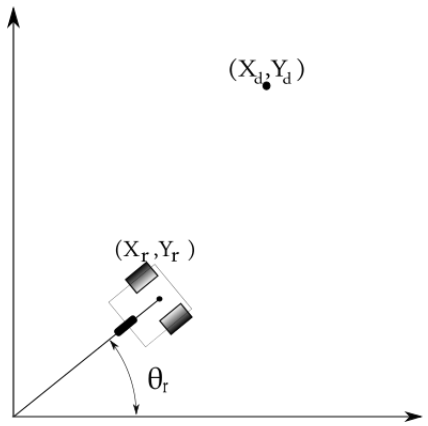


Figura A.1 Ubicación de un robot en el plano.

A.1 Control de orientación

El primer control consiste en lograr la orientación del robot, por lo que el objetivo es llevar al robot desde una orientación θ_r (orientación actual) a una θ_d (orientación deseada). Dado que la orientación (en el caso de este proyecto), va a estar dada por un punto asignado, se calcula cuál es el error presente entre la posición actual y la posición deseada:

$$P_D - P_R = \begin{bmatrix} x_d - x_r \\ y_d - y_r \end{bmatrix} = \begin{bmatrix} e_x \\ e_y \end{bmatrix} \tag{A.1}$$

Tomando esto en cuenta, se traza un vector de apoyo que definiremos como u (Figura A.2), el cual va en la dirección deseada y se puede expresar de la siguiente forma:

$$\begin{aligned} \vec{u} &= (x_d - x_r, y_d - y_r) \\ &= (e_x, e_y) \end{aligned} \tag{A.2}$$

Por lo anterior, para obtener la θ_d se hace uso de la función $atan2(y, x)$, la cual ayuda a determinar el angulo en función del valor de las coordenadas del vector u.

$$\theta_d = atan2(e_y, e_x) \tag{A.3}$$

Para poder efectuar la acción de control es necesario obtener el error de ángulo (e_θ), que queda definido de la siguiente manera:

$$\begin{aligned} e_\theta &= \theta_d - \theta_r \\ &= atan2(e_y, e_x) - \theta_r \end{aligned} \tag{A.4}$$

Por lo que la ley de control que se propone es la siguiente acción proporcional:

$$\begin{aligned} \omega_i &= -K_a e_a \qquad \text{Donde: } K_a \in \mathbb{R}^+ \\ \omega_d &= K_a e_a \end{aligned} \tag{A.5}$$

A.2 Control de desplazamiento

El control de desplazamiento del robot, se llevó a cabo teniendo en cuenta las restricciones holonómicas, esto quiere decir que sólo se puede desplazar en linea recta, por lo que con base al error de posición (e_p) se desarrolló la ley de control, este error se define de la siguiente manera:

$$\begin{aligned} e_p &= \sqrt{(x_d - x_r)^2 + (y_d - y_r)^2} \\ &= \sqrt{e_x^2 + e_y^2} \end{aligned} \tag{A.6}$$

Y la acción de control que se propone es:

$$\begin{aligned} \omega_i &= K_p e_p \qquad \text{Donde: } K_p \in \mathbb{R}^+ \\ \omega_d &= K_p e_p \end{aligned} \tag{A.7}$$

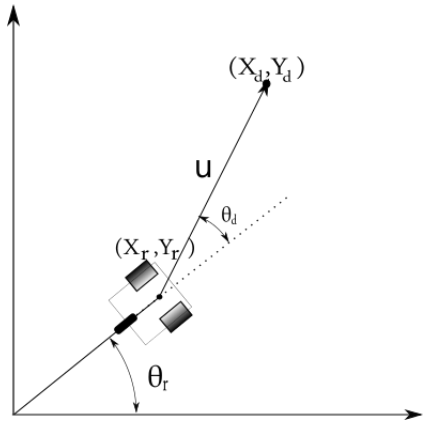


Figura A.2 Plano de orientación del vehículo.

Por lo anterior, las leyes de control definidas para el posicionamiento del robot quedan de la siguiente forma:

$$\omega_i = \begin{cases} -K_a e_a & \text{si} \quad e_a <= tol_{ang} \\ K_p e_p & \text{si} \quad e_p <= tol_{pos} \end{cases} \tag{A.8}$$

$$\omega_d = \begin{cases} K_a e_a & \text{si} \quad e_a <= tol_{ang} \\ K_p e_p & \text{si} \quad e_p <= tol_{pos} \end{cases} \tag{A.9}$$

Teniendo esto en cuenta, se propone un control conmutado, el cual consiste primero en orientar el vehículo y a continuación seguir en linea recta hasta llegar al punto deseado, además se agregan dos nuevos términos tol_{ang} (tolerancia en el error de ángulo) y tol_{pos} (tolerancia en el error de posición) los cuales se proponen para asegurar que el robot no se mantendrá en un ciclo infinito.

A.3 Implementación de control en LabVIEW

En esta sección se aborda la manera en que se implementó esta ley de control en LabVIEW. En este caso sólo se ejemplifica la forma en que se alcanzan los puntos que seguirá el robot en base a una trayectoria ya definida. En la Figura A.3 se muestran los estados diseñados para cumplir con la tarea de seguimientos de trayectoria.

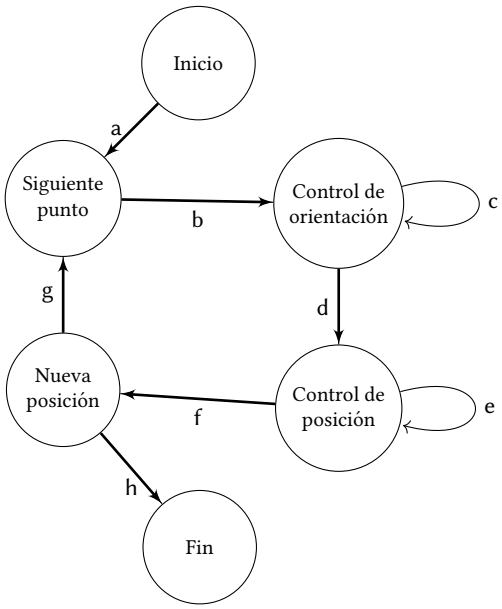


Figura A.3 Diagrama de estados para conmutación de control

Las condiciones para efectuar el cambio de estado son las siguientes:

a .- *Tiempo de espera cumplido*

b .- *Acción de estado realizada (Véase sección A.3.2)* .

c .- *Error de tolerancia no cumplida.*

d .- *Error de tolerancia cumplida.*

e .- *Error de tolerancia no cumplida.*

f .- *Error de tolerancia cumplida.*

g .- *Existe un punto siguiente.*

h .- *Ningún punto siguiente.*

La descripción del funcionamiento de los estados así como su implementación en LabVIEW se describe en las secciones siguientes. Cabe mencionar que para la implementación de una máquina de estados se puede emplear una plantilla diseñada para este propósito, la cual consiste en la mezcla estructuras del tipo *While Loop* y *Case*, que de acuerdo a una constante del tipo *Enum* define los estados a seguir.

A.3.1

Estado: Inicio

Este primer estado consiste simplemente en esperar un segundo para poder realizar una transición al siguiente estado. El motivo de este estado es generar un retraso en el inicio del seguimiento de la trayectoria.

A.3.2

Estado: Siguiente punto

En este estado, dado que la trayectoria se encuentra definida por una serie de puntos a seguir, se realiza un incremento del índice que se emplea para indicar la posición del arreglo que se tomará en cuenta para fijar el nuevo punto deseado.

A.3.3

Estado: Control de Orientación

Este estado es el encargado de orientar al robot en una posición deseada, la cual toma en cuenta el error de ángulo que se genera mediante las ecuaciones (A.4). Posteriormente, tomando en cuenta este dato, se le aplica una ganancia para después aplicar estas velocidades en las ruedas del carro de acuerdo a la ley de control propuesta mediante las ecuaciones (A.5). En la Figura A.4 se muestra el estado implementado, así como las distintas transiciones entre estados dependiendo si se alcanza o no la tolerancia de error definida.

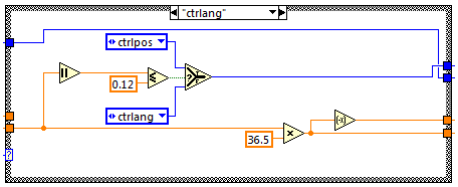


Figura A.4 Estado "Control de Orientación" implementado en LabVIEW

A.3.4

Estado: Control de posición

En el momento en que se alcanza la tolerancia deseada en el control de orientación se prosigue a este estado, el cual busca alcanzar la posición deseada empleando el error de posición definido mediante las ecuaciones (A.6). De la misma forma que el estado anterior se aplica una cierta ganancia a este error para posteriormente aplicar las velocidades a las ruedas de acuerdo a la ley propuesta en las ecuaciones A.7.

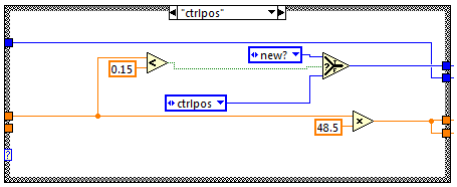


Figura A.5 Estado "Control de posición" implementado en LabVIEW

A.3.5

Estado: Nueva posición

Este estado realiza una comparación empleando la condicionante para considerar que la trayectoria ha sido finalizada o no, ya que dependiendo del número de puntos definidos en la trayectoria, se decide si se retornará al estado **Siguiente punto** o **Fin**.

A.3.6

Estado: Fin

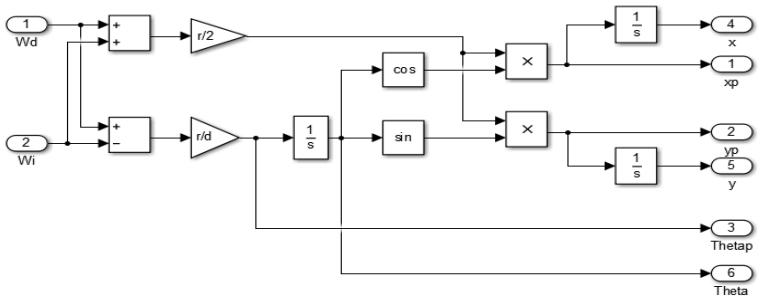
En este último estado se asignan las velocidades de cero a ambas ruedas y se finaliza la ejecución del programa.

A.4 Pruebas del control de posicionamiento

Se realizaron distintas actividades para probar el funcionamiento del programa descrito en la sección anterior, el cual es el encargado del seguimiento de trayectorias, dentro de las que se incluyen la prueba del funcionamiento de las leyes de control en el software MATLAB y la sintonización de las ganancias del controlador para obtener el mínimo error.

A.4.1 Simulación del sistema de control en MATLAB

Mediante las ecuaciones del sistema (Ec. 2.11) y las leyes de control (A.8) y (A.9), se desarrolló el modelo en el programa MATLAB, el cual otorga la posibilidad de observar el funcionamiento de las leyes implementadas mediante simulaciones.
En la Figura (A.6a) se muestra el modelo del sistema (robot móvil diferencial). Como entradas se tienen wd y wi, las cuales corresponden a las velocidades angulares de cada una de las ruedas, mientras que las salidas $\dot{x}, \dot{y}, \dot{\theta}, x, y, \theta$ representan las velocidades y posición del robot en un marco de referencia (X, Y). En el segmento derecho de la misma figura se encuentra el script que representa la asignación de las condiciones del modelo, como lo son: la distancia entre las ruedas actuadas, el radio de las ruedas y las condiciones iniciales de posición del robot.



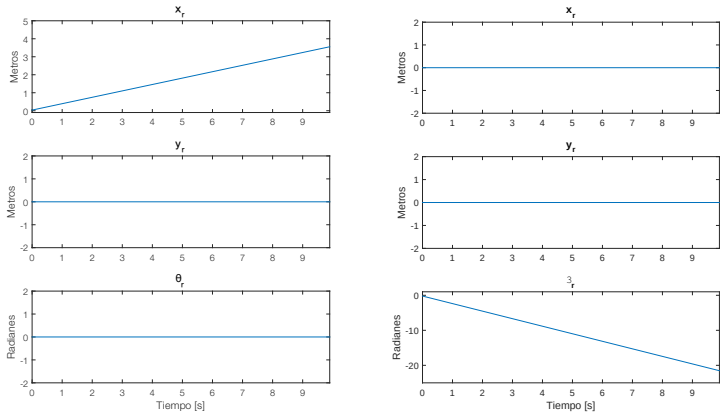
(a) Modelo en Simulink.

```
%% Parametros del robot
d=0.3302; %% Distancia entre ruedas
r=0.0508; %% Radio de las ruedas
%% Condiciones iniciales
x0=0;
y0=0;
Theta0=0;
%% Velocidad de las ruedas
wd=-7;
wi=7;
```

(b) Script de condiciones.

Figura A.6 Modelo del robot diferencial en MATLAB

En la Figura A.7 se aprecia cuál es el desplazamiento del robot a distintas condiciones de velocidad angular en las ruedas. En el primer caso (izquierda), se aprecia que si las velocidades angulares de las llantas son iguales, existe un desplazamiento en línea recta. Este cambio se percibe con el crecimiento de la variable x_r mientras las variables y_r y θ_r (orientación) se mantiene constante. En caso contrario, si las velocidades son contrarias (caso de la derecha), existe un cambio de orientación del robot respecto al marco de referencia, el cual es apreciado al mantenerse las variables de posición (x_r e y_r) se mantienen constantes y el aumento de valor se ve presente en la variable θ_r.



(a) Velocidades iguales.

(b) Velocidades distintas.

Figura A.7 Respuestas del sistema con condiciones iniciales iguales a cero.

A.4.1.1 Implementación del control en MATLAB

En la simulación de la ley de control se implementó el diagrama de bloques que se aprecia en la Figura A.8.
El subsistema recibe como argumentos los datos que se generan en el bloque que simboliza el comportamiento del robot (Figura A.6a) referente a su posición y retorna los valores a emplear como velocidades angulares en las ruedas, las cuales tienen como objetivo el posicionamiento del robot en un punto deseado. El diseño del sistema está compuesto por bloques correspondientes a las operaciones que forman las ecuaciones (A.8) y (A.9) y contienen un elemento de la paleta *Stateflow*, el cual es empleado para la realización de una máquina de estados que realiza la conmutación del control.

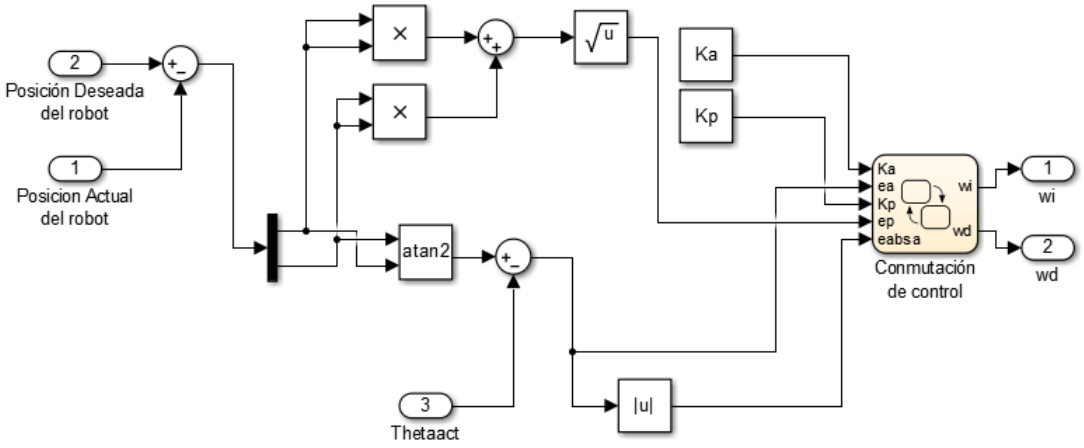


Figura A.8 Modelo de control implementado en Simulink.

Los componentes de la máquina de estados se muestran en la Figura A.9, los cuales son: los bloques de estados y las condiciones de transición entre estos. Como se describió en la sección anterior, se asignaron tolerancias, que en este caso son las encargadas de efectuar el cambio de estados. El funcionamiento de esta máquina consiste primero en orientar al robot, a continuación realizar el desplazamiento en línea recta hasta alcanzar el punto deseado y por último asignar la velocidad cero a las ruedas para concluir con el desplazamiento del robot.

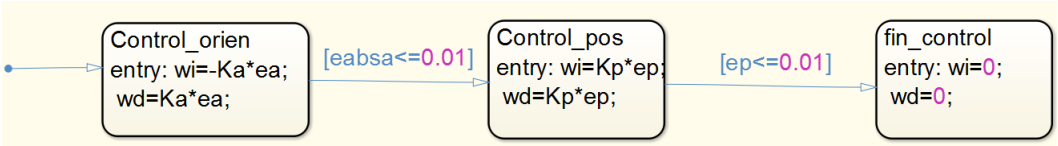


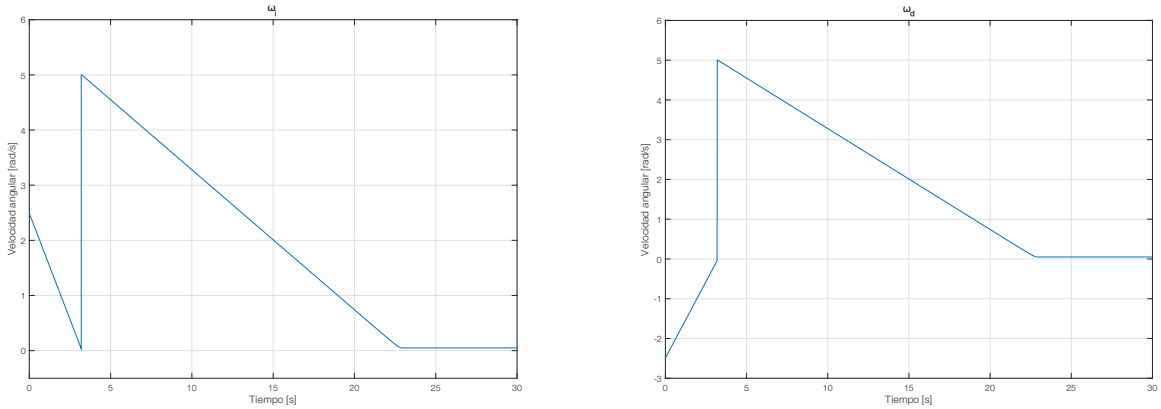
Figura A.9 Diagrama de estados implementados.

A.4.1.2 Respuesta de la simulación

Se probó la respuesta de la ley de control con las siguientes restricciones:

- Condiciones iniciales:
  - $x_0 = 0.\, y_0 = 0,\, \theta_0 = 0$
- Posición deseada:
  - $x_d = -4,\, y_d = -3$
- Ganancias:
  - $K_a = K_d = 1$

Con las condiciones antes descritas, se obtuvieron los siguientes resultados que se aprecian en la Figura A.10, en los cuales se aprecia la variación de velocidad de las ruedas a lo largo del tiempo, además de que claramente se nota en qué momento se lleva a cabo la conmutación del tipo de control. Sin embargo, en estas gráficas no se aprecia del todo cómo el robot alcanza la posición deseada, por lo que empleando las ecuaciones del modelo odométrico (2.11) se obtienen los resultados de la aproximación al punto deseado.



(a) Rueda izquierda.

(b) Rueda derecha.

Figura A.10 Velocidades obtenidas de las ruedas.

En la Figura A.11 se nota el resultado de la aplicación de la ley de control y es posible observar que el robot alcanza la posición deseada en un tiempo finito así como la linealidad de la respuesta.

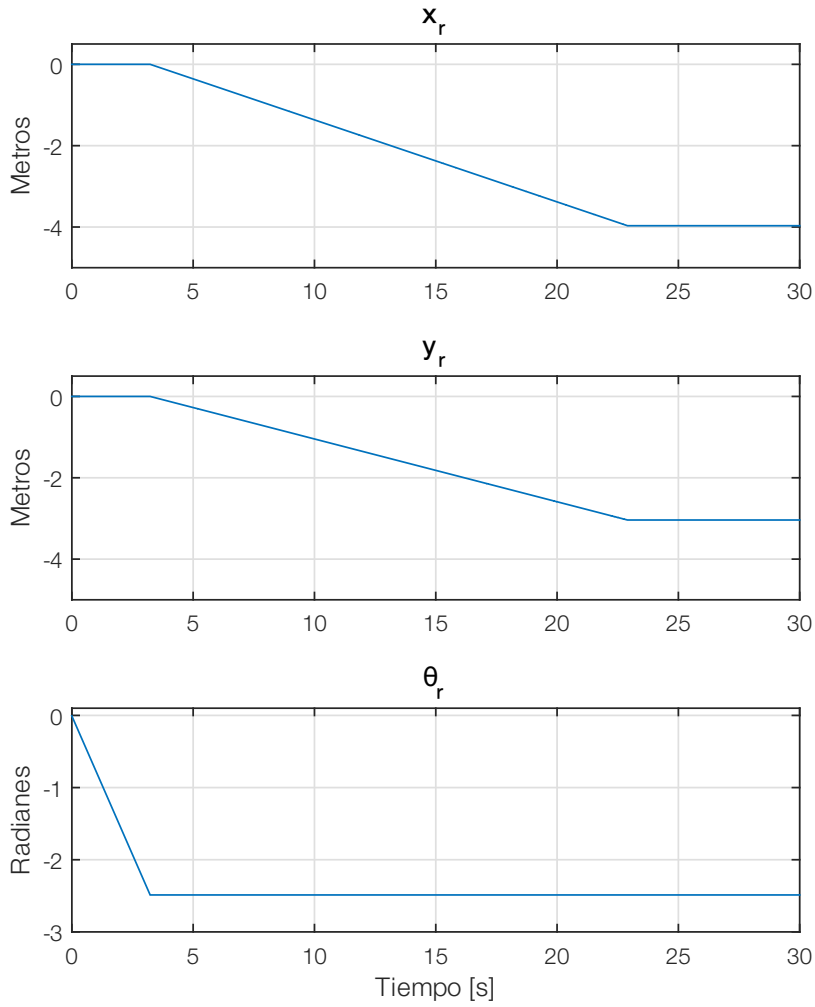


Figura A.11 Comportamiento odométrico.

Sintonización de ganancias del controlador

Después de verificar el funcionamiento de la ley de control propuesta en simulación, se prosiguió a la sintonización de ganancias. Esto se llevó a cabo mediante una sencilla prueba, la cual consistía en mover al robot de un posición inicial $P_0 = (0, 0)$ a una posición deseada $P_D = (-2, 0)$. Esto quiere decir que el robot realizará un giro de 180 grados y avanzará en línea recta una distancia de dos metros. En la Figura A.12 se muestra el robot y la línea que se empleó para tener una referencia respecto al funcionamiento deseado del robot. En las tablas (A.1) y (A.2) se aprecian los distintos resultados que se obtuvieron a distintas ganancias aplicadas, por lo que con base en estos experimentos las ganancias asignadas fueron $K_a = \mathbf{36.5}$ y $K_p = \mathbf{48.5}$.

Figura A.12 Prueba para sintonización de ganancias

Error de angulo [°]	K_a	tol_{ang}
60	50	0.03
25	25	0.03
25	30	0.03
22	35	0.03
23	40	0.03
20	35	0.03
20	40	0.03
21	38	0.03
10	36	0.03
13	36.5	0.03
9	36.5	0.03
14	36.5	0.03
12	36.5	0.03
14	36.5	0.03
17	36.5	0.03

Tabla A.1 Pruebas para determinación de ganancia K_a

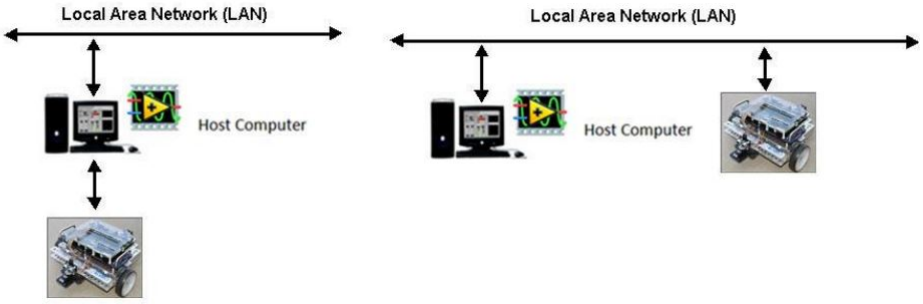
Error pos. [m]	K_p	tol_{pos}
1.5	50	0.1
0.36	50	0.1
0.11	50	0.1
0.12	50	0.1
0.1	50	0.1
0.11	50	0.1
0.12	50	0.1
0.5	50	0.1
0.23	40	0.1
0.4	45	0.1
0.3	47	0.1
0.3	48	0.1
0.5	49	0.1
0.3	48.5	0.1
0.2	48.5	0.1

Tabla A.2 Pruebas para determinación de ganancia K_p

B

Conexión a una red

En este apartado se detalla el procedimiento para la conexión de los dispositivos en la misma red.
En el caso de la conexión entre DaNI y la computadora huésped existen dos formas de realizar la conexión, las cuales se muestran en la Figura B.1. La primera de ellas es a través de una red punto a punto entre el *Host* y DaNI y la segunda mediante una red de área local (LAN).



(a) Conexión Punto a Punto (b) Conexión a una red LAN

Figura B.1 Formas de conexión

El primer paso para la configuración de la red, es declarar cual será la dirección IP que tendrá DaNI dentro de la red. Para lograr esto es necesario conectar DaNI con la computadora huésped a través de cable Ethernet (UTP-5). En la computadora se ejecuta la aplicación *NI Measurement and Automation Explorer (MAX)*, la cual es una herramienta provista por National Instruments que está desarrollada para la identificación y configuración de los dispositivos, instrumentos y software distribuido por esta compañía.

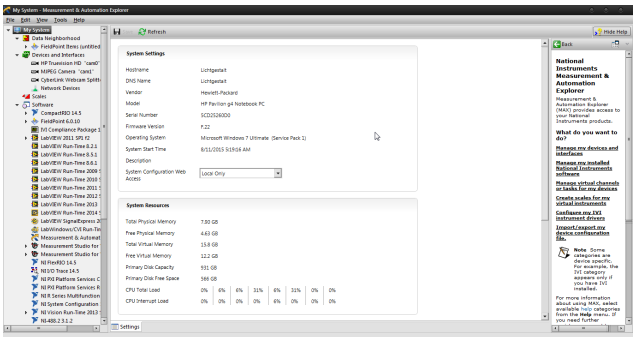


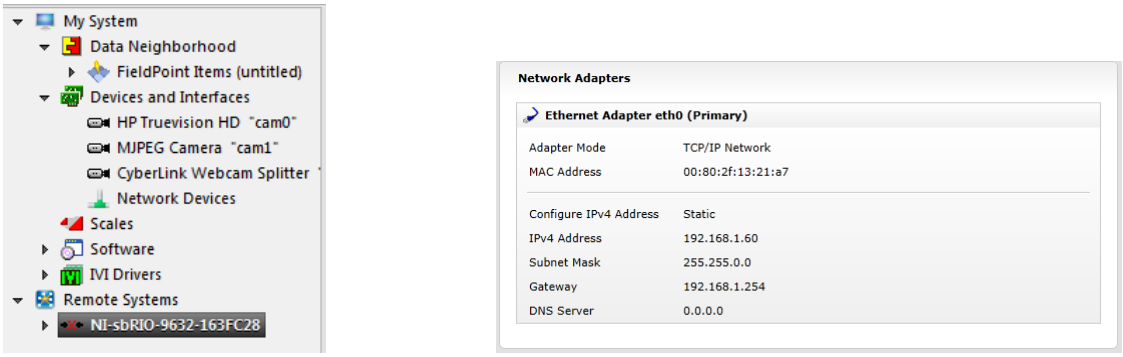
Figura B.2 Measurement and Automation Explorer

En la parte izquierda se muestran dos secciones importantes las cuales son:

- *My System*: Es donde se encuentran los dispositivos(cámaras, tarjetas de adquisición de datos, instrumentos PXI, etc.) y software (Módulos de expansión de LabVIEW y distintos controladores para equipos) que contiene la computadora huésped.

- *Remote Systems*: Aquí se muestran los dispositivos que se encuentran en la misma red que la computadora huésped, este apartado es el empleado para la configuración de DaNI.

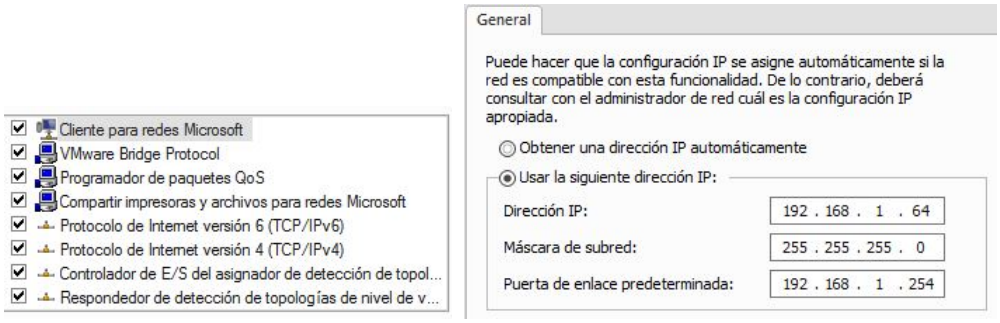
Cuando se selecciona el apartado *Remote Systems*, se despliega la lista dispositivo y se elige el que se desea configurar que en este caso es la tarjeta sbRIO-9632 (Fig. B.3a), lo que ocasiona el despliegue en pantalla de la información sobre el dispositivo. Seleccionando la pestaña *Network Settings* aparecen los datos relacionados a la red en que se pretende conectar. Como se muestra en la figura B.3b, la dirección IP que se le asignó a DaNI fue la **192.168.1.60**.



(a) (b)

Figura B.3 Measurement and Automation Explorer (Remote Devices)

Después de esto, se procede a configurar la computadora huésped para que tanto esta como DaNI se encuentren en la misma red, por lo que dependiendo el modo de conexión que se realice es necesario configurar el adaptador de red para asignarle una dirección IP estática. En la dirección: *Panel de control\Redes e Internet\Conexiones de red*, es posible seleccionar el adaptador que se desea configurar. En este caso se seleccionó el adaptador para conexiones LAN (alámbrica). Al verificar sus propiedades (Figura B.4a) se modifican los parámetros **TCP/IPv4** de tal forma que ambos dispositivos se encuentren en la misma red (Figura B.4b), la dirección asignada a la computadora fue la **192.168.1.64**.



(a) (b)

Figura B.4 Configuración de red del Host

Para comprobar que el cambio de dirección ha sido exitoso se puede acceder a la consola de comandos e ingresar la instrucción *ipconfig*. Como respuesta aparecen las configuraciones de los adaptadores de red disponibles en ese momento, en la Figura B.5 se aprecia que la configuración ha sido exitosa al visualizar la dirección asignada.

```
Adaptador de LAN inalámbrica Conexión de red inalámbrica:

   Sufijo DNS específico para la conexión. . : lan
   Vínculo: dirección IPv6 local. . . : fe80::25d2:daff:df0:e425%11
   Dirección IPv4. . . . . . . . . . . . : 192.168.1.64
   Máscara de subred . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.1.254
```

Figura B.5 Dirección IP de la computadora huésped

De la misma manera, se configuró el teléfono celular con una dirección IP estática, aunque es posible conectarse mediante el protocolo DHCP, es recomendable realizar esta configuración dado que el adaptador requiere conocer la

dirección de la que provienen los datos de vídeo y audio. Para configurar el protocolo a emplear se accede al menú *Configuración\ Wi-Fi* y seleccionando la red que se desea configurar se selecciona la opción *Opciones Avanzadas*. En este caso, como se muestra en la Figura B.6, se decidió asignarle al móvil la dirección **192.168.1.116**.

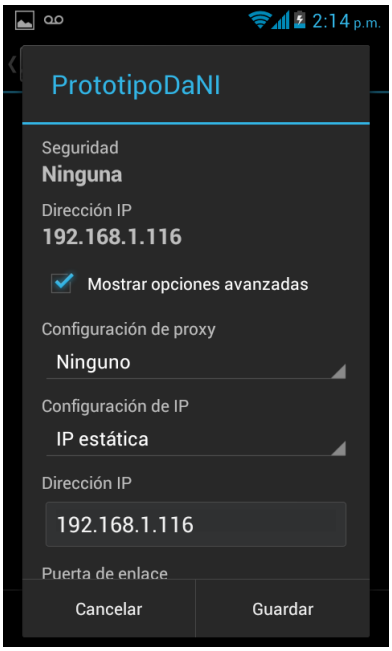


Figura B.6 Configuración para una IP estática.

C

Configuración FPGA

Cuando se realizan aplicaciones donde se involucran FPGA's, es necesario saber el procedimiento para poder acceder a sus distintos puertos. En este anexo se explica la forma en que se realiza la configuración de éstos.

El primer paso consiste en agregar un objetivo FPGA y esto se logra al seleccionar dentro del proyecto el dispositivo que contiene el FPGA. En este caso es la tarjeta sbRIO-9632; cuando esto se lleva a cabo se despliegan en el contenido del proyecto los elementos del FPGA que pueden ser configurados (Figura C.1).

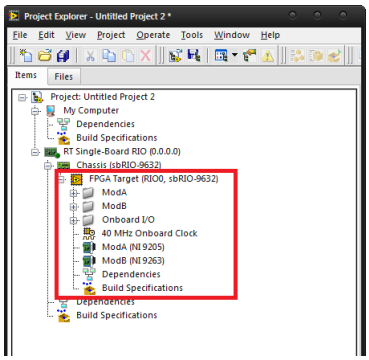
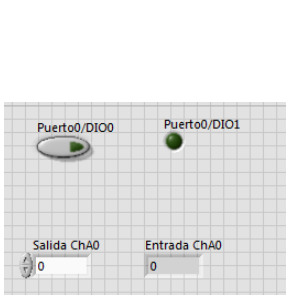


Figura C.1 Elementos del proyecto (FPGA Target)

El siguiente paso es agregar un nuevo VI dentro del objetivo, donde se seleccionarán los puertos a configurar. Esto se lleva a cabo mediante las herramientas provistas en la paleta *Programming\FPGA I/O* de donde se selecciona el elemento *FPGA I/O Node*, el cual tiene la función de listar los puertos disponibles del FPGA para su configuración. Para esta prueba se seleccionaron cuatro posibilidades, que son puertos de entrada y salida, analógicos y digitales.



(a) Panel frontal

(b) Diagrama de bloques

Figura C.2 VI dentro del FPGA objetivo

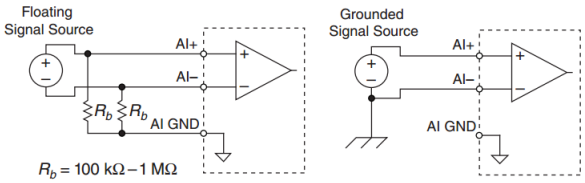
Como se muestra en la Figura C.2 se creó un VI con los elementos deseados a configurar. En el caso de los puertos analógicos de entrada existe la posibilidad de configurar en qué modo se tomará la lectura; esto se realiza mediante la selección de propiedades del módulo que contiene la tarjeta (Módulo NI-9263). En la Figura C.6 se muestra la ventana que se despliega para esta configuración, en donde es posible asignar el de rango de entrada y el modo en que se realizará la lectura de cada canal.

Los modos de lectura son los siguientes:

- **DIFF (Configuración Diferenial):** Esta configuración es con la que se cuenta predefinida en el paquete de *Robotics*. Es usada para tener una mejor lectura ya que reduce el ruido. El único inconveniente es que disminuye el número de canales disponibles ya que usa dos para cada lectura. La relación entre canales se muestra en la Figura C.3a. Para realizar la lectura en este modo es necesario identificar el tipo de lectura que se realizará. La manera de realizar mediciones para fuentes flotantes y referenciadas a tierra se muestran en la Figura C.3b.

Canal	Señal +	Señal -	Canal	Señal +	Señal -
0	AI0	AI8	16	AI16	AI24
1	AI1	AI9	17	AI17	AI25
2	AI2	AI10	18	AI18	AI26
3	AI3	AI11	19	AI19	AI27
4	AI4	AI12	20	AI20	AI28
5	AI5	AI13	21	AI21	AI29
6	AI6	AI14	22	AI22	AI30
7	AI7	AI15	23	AI23	AI31

(a) Parejas diferenciales.



(b) Conexión para entradas diferenciales

Figura C.3 Datos útiles sobre la configuración diferencial

- **RSE (Referenced Single-Ended):** En este modo de lectura es posible emplear los 32 canales disponibles. Se emplea en el caso de que todos los canales compartan la misma referencia a ”tierra”, es recomendado para la medición de fuentes flotantes ya que en el caso de las referenciadas a tierra pueden existir errores de medición debido a la diferencia de potencial entre la tierra y la parte de menor potencial en la fuente (Figura C.4).

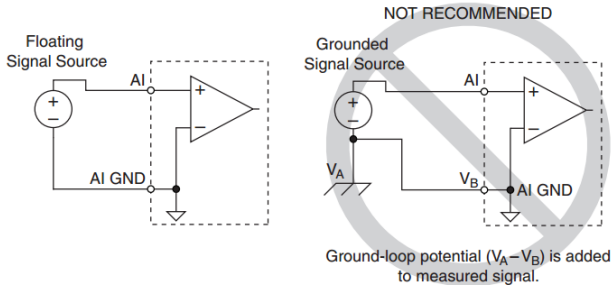


Figura C.4 Conexión de entradas en modo RSE

- **NRSE (Non-Referenced Single-Ended):** Este modo es similar al anterior con la diferencia de que es más efectivo en el rechazo del ruido. Esta configuración provee un sensado remoto para la entrada negativa del amplificador programable de instrumentación. En la Figura C.5 se muestran los modos de conexión para distintos tipos de fuentes.

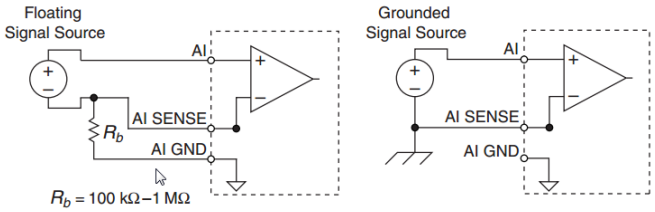


Figura C.5 Conexión de entradas en modo NRSE

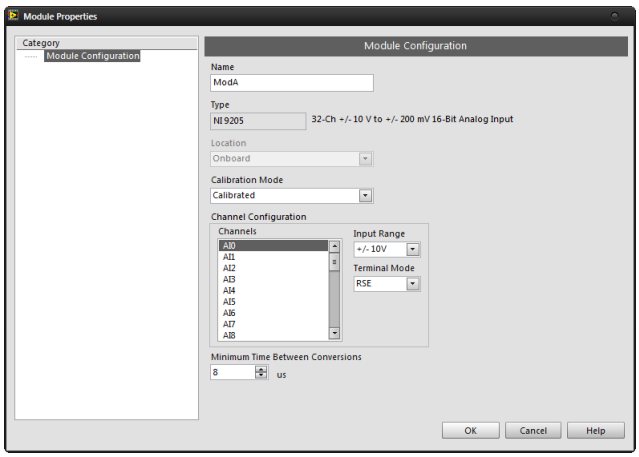
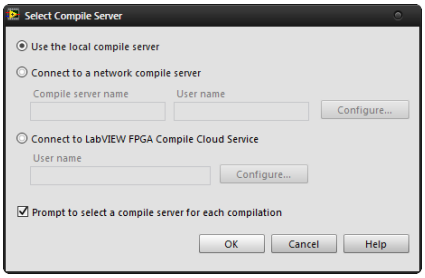
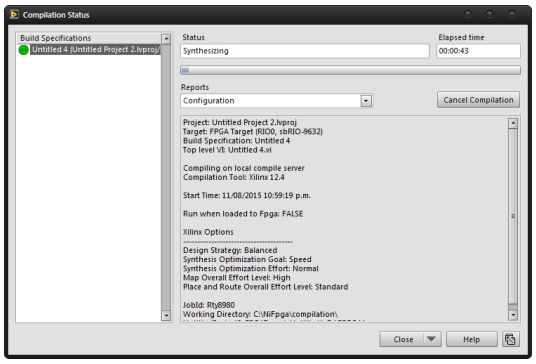


Figura C.6 Configuración de propiedades del módulo de entradas analógicas

En el momento que se ejecuta este VI, se realiza el proceso de compilación necesario para generar los archivos de configuración del FPGA. El primer paso consiste en seleccionar el servidor que efectuará la compilación, en este caso se seleccionó el servidor local, ya que el módulo *LabVIEW Robotics* ofrece un compilador para los FPGA's Xilinx. También es posible utilizar servidores que se encuentran disponibles en la nube.



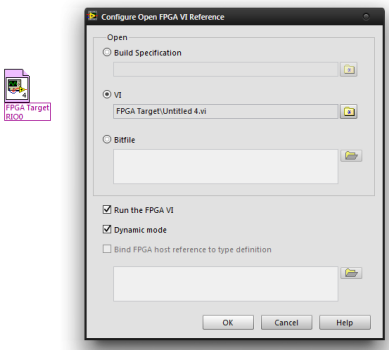
(a) Selección del servidor



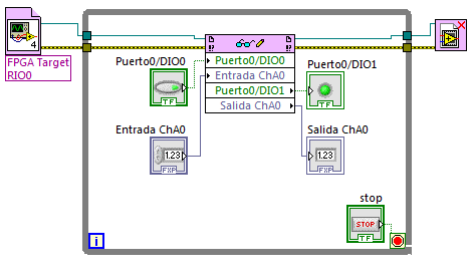
(b) Detalles del proceso

Figura C.7 Proceso de compilación

En el momento de finalizar el proceso de compilación, es posible emplear los archivos generados para el acceso a los puertos del FPGA utilizando los elementos provistos por la paleta *FPGA Interface*. En la Figura C.8a se muestra que al seleccionar el objeto *Open FPGA VI Reference* es posible decidir el tipo de archivo que se usará como referencia ya sea un VI o los archivos binarios generados en la compilación. Al seleccionar el modo de referencia, es posible utilizar los puertos en el modo que se decidieron anteriormente mediante el objeto *Read/Write Control* con el cual se accede a los puertos configurados (Figura C.8b).



(a) Apertura de los archivos generados



(b) Diagrama de bloques básico

Figura C.8 Uso del FPGA

D

Pruebas básicas de funcionamiento

En este Anexo se incluyen las pruebas empleadas para verificar el funcionamiento individual de los elementos que conforman el robot, así como pruebas de conectividad, localización y procesamiento de video.

Objetivos

- ☒ Entender el funcionamiento de un robot móvil con configuración diferencial.
- ☒ Realizar las distintas conexiones entre los sensores y la tarjeta sbRIO-9632.
- ☒ Realizar la transmisión de video y audio desde un teléfono celular a una computadora.
- ☒ Obtener las soluciones del modelo odométrico del robot móvil con configuración diferencial.
- ☒ Desarrollar una Interfaz Hombre-Maquina (HMI) para la teleoperación del robot.

Material y Equipo

Software:

- ☒ LabVIEW 2011 *Service Pack 1*.
- ☒ Controladores para NI-RIO.
- ☒ Programa de apoyo para procesamiento de video.

Elementos físicos:

- ☒ Computadora con conector Ethernet (RJ-45) y adaptador Wi-Fi.
- ☒ *Router* con conectores Ethernet y soporte a redes WLAN.
- ☒ Módulo de desarrollo NI DaNI.
- ☒ Joypad genérico.

Sensores:

- ☒ Sensor de temperatura LM35.
- ☒ Módulo detector de flama SEN05082P.
- ☒ Película piezoeléctrica.
- ☒ Sensor de calidad del aire MQ-135.
- ☒ Módulo GPS EVA2035-H.

Desarrollo de la práctica

Todos los instrumentos virtuales que se emplearán en esta práctica se encuentran contenidos en la carpeta*Actividades de la práctica* dentro del proyecto **Teleop v2.0.lvproj**. El cuál se encuentra disponible en la coordinación de laboratorios de Medición e Instrumentación.

Actividad 1 **Prueba de elementos de control en un entorno simulado**

1.- En el proyecto elaborado buscar en la dirección *Actividades de la práctica/Actividad 1* y ejecutar el archivo **Prueba de elementos de control.vi**(Asegúrese de tener conectado el Joypad).

Figura D.1 Interfaz para la prueba de controles

2.- En la interfaz del programa se mostrará la pantalla que se muestrá en la figura D.1.

3.- El objetivo de este programa es verificar el correcto funcionamiento del Joypad y las teclas de desplazamiento del teclado. Presionando las teclas de dirección tanto del control de juego como del teclado, se activarán los indicadores de tipo LED indicando cuál es la instrucción que se ha seleccionado. En caso de que el Joypad no se encuentre reaccionando verificar que la lectura analógica del control esta activada.

4.- Posterior a la verificación del funcionamiento se procederá con la simulación de la operación de un robot móvil, para lo cual se empleará el simulador de robótica provisto por el paquete *LabVIEW Robotics*. Para esto se ejecutará el instrumento **Simulador de desplzamiento.vi**.

5.- Al ejecutar el programa se desplegará una ventana como la que se muestra en la figura D.2, al ejecutar este programa se mostrará una nueva ventana en la cual se puede observar un robot móvil (como el que se emplea en esta práctica) en un entorno de desastre. La tarea a realizar consiste en desplazar al robot utilizando las teclas de desplazamiento y visualizar la trayectoria que este describe y compararla con la que se dibuja en la interfaz de la ventana principal.

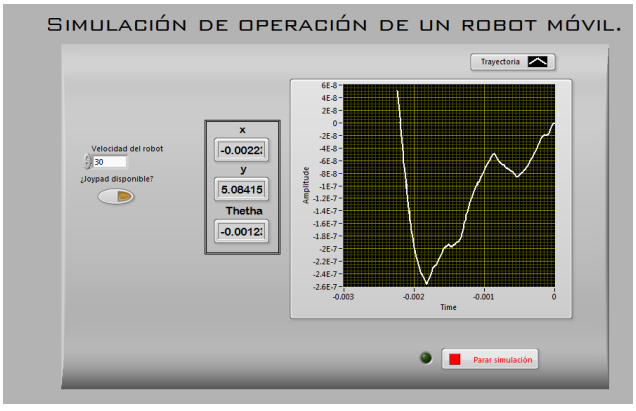


Figura D.2 Simulación de la operación de un robot móvil

Actividad 2 **Conexión entre la computadora del operador y el robot DaNI**

Para realizar esta actividad se recomienda leer el Anexo B.

1.- Con el dispositivo conectado a la computadora del operador, ejecutar el programa *NI Measurement & Automation Explorer* y en el apartado de **Remote Devices** seleccionar la tarjeta **NI-sbRIO-9632-163FC28** y abrir la pestaña **Network Settings**, en esta ventana se verificarán los datos de conexión pertenecientes al robot. En caso de que los campos se encuentren vacíos, se asignarán los datos siguientes:

Configure IPv4 Address **Static**
IPv4 Address **192.168.1.60**
Subnet Mask **255.255.0.0**
Gateway **192.168.1.254**
DNS Server **0.0.0.0**

2.- Configurar el adaptador de red de la computadora huésped de tal forma que la dirección IP asignada se encuentre en la misma red que el robot, para esto ingresar en el **Centro de Redes y Recursos Compartidos** y presionar la opción **Cambiar configuración del adaptador**, desde esta ventana se seleccionará el adaptador de red mediante el cual se va a realizar la conexión. Los parámetros que se definirán para la red, son los siguientes:

Dirección IP: **192.168.1.65**
Máscara de subred: **255.255.255.0**
Puerta de enlace predeterminada: **192.168.1.254**

Actividad 3 Prueba individual del funcionamiento de sensores

1.- Realizar la conexión de los sensores como se explica en la sección 3.6.

2.- En la carpeta *Actividad 3*, la cual se encuentra contenida en la sección *RT Single-Board RIO (192.168.1.60)* abrir el instrumento virtual **Prueba_modulo_GPS.vi**. Se abrirá una ventana como la mostrada en la figura D.3, durante la ejecución del programa se obtendrán las cadenas de datos emitidas por el receptor GPS, se sugiere habilitar la opción *Almacenar datos* para verificar los datos recibidos.

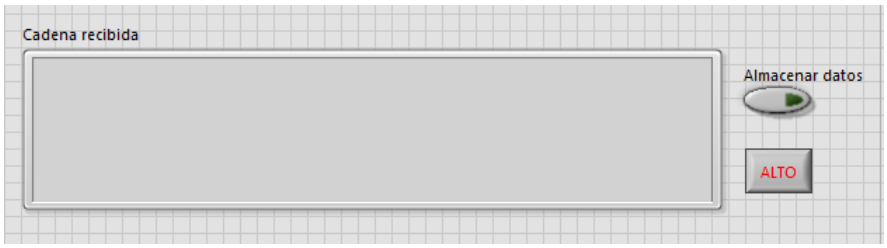


Figura D.3 Recepción de cadenas del GPS

3.- Verificar cuáles son los distintos tipos de oraciones recibidas, así como los datos que éstas contienen. Revisar el Anexo E.

4.- En la carpeta *Actividad 3* que se encuentra contenida en la sección *My Computer* abrir el instrumento virtual **Lectura_de_sensores.vi**. Este contiene el control y los indicadores necesarios para probar la lectura de cada uno de los sensores con los que cuenta el robot (Fig. D.4). Para probar el funcionamiento de estos sensores, se recomienda hacer la siguientes pruebas:

Detector de flama: Generando una flama controlada a una distancia de aproximadamente un metro del robot en sentido a donde apunta el fototransistor, ajustar el potenciómetro del modulo detector de flama hasta que se emita una señal booleana verdadera que indique la presencia de la flama.
Sensor de temperatura: Verificar el incremento de voltaje mediante el contacto del sensor con una fuente de calor.
Sensor de calidad del aire: Corroborar el incremento de voltaje ante una fuente de humo.
Detector de vibración: Generar distintos tipos de impacto (vibraciones) en torno al robot y verificar la presencia de ”picos” en las lecturas de este detector. Para esta prueba se recomienda emplear un objeto esférico rígido (como una bola de billar) sobre una tarima para tener resultados mas apreciables.

Actividad 4 Prueba de recepción de video

1.- Ingresar a la pagina «*ip-webcam.appspot.com*» y realizar la descarga e instalación del adaptador para cámaras IP.

2.- Conectar el teléfono celular a la red en la que se encuentra la computadora encargada de la manipulación del robot y ejecutar la aplicación **IP Webcam**, verificar cual es la dirección IP asignada al teléfono.

3.- En la computadora ejecutar el programa **Configure IP Camera Adapter** e ingresar la dirección IP del celular.

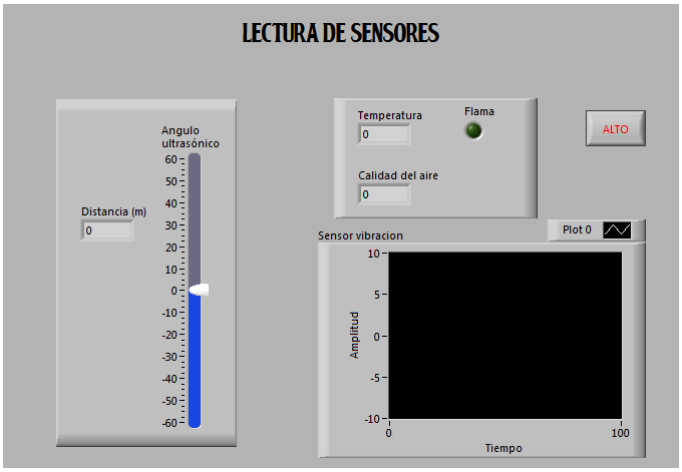


Figura D.4 Prueba de lectura de sensores

4.- Dentro del proyecto donde se contienen los instrumentos virtuales para esta práctica, abrir la aplicación **Recepción_de_video.vi** que se encuentra en la dirección *My Computer/Actividades de la práctica/Actividad 4*. Seleccionar el identificador de la fuente de video (Verificar el origen de video mediante el programa NI MAX) y ejecutar el instrumento para comprobar la transmisión de video del celular.

Actividad 5 **Detección de color y movimiento**

1.- Ejecutar el programa **VideoProcesamientov1.exe**.

2.- Seleccionar la pestaña **Abrir cámara** y seleccionar la fuente desde la que se obtendrá el video. Desde esta ventana también es posible configurar la resolución con la que se mostrar el video.

3.- Para verificar el funcionamiento del detector de color, es necesario emplear tres objetos de distintos colores (de preferencia que contrasten entre si). Desde la pestaña **Procesamiento**, se seleccionará la opción **Color**, lo cual ocasionará que se habiliten los controles para ajustar las opciones de procesamiento. Desde los controles habilitados seleccionar el color de cada uno de los objetos que se emplearán y verificar el seguimiento del objeto en pantalla. De la misma manera se sugiere verificar el valor de la propiedad **Área mínima de reconocimiento [pixeles]** y observar el comportamiento de esta propiedad en pantalla.

4.- Para verificar el funcionamiento del detector de movimiento, seleccionar desde la pestaña **Procesamiento** y probar alguno de los dos algoritmos distintos para la detección de movimiento. Con cada uno de los algoritmos de detección probar los distintos métodos de visualización contenidos en la sección **Herramientas de movimiento** y observar las ventajas y desventajas de cada uno.

5.- Con la opción de detección de movimiento habilitada modificar el valor de la propiedad **Sensibilidad de movimiento** y habilitar la opción **Activar captura por alarma**. Anote sus observaciones.

Actividad 6 **Operación del robot explorador** Para saber mas sobre la operación del robot se sugiere leer el capitulo 2.

1.- Conectar el robot DaNI a la red en la que se encuentra la computadora desde la que se operará.

2.- Desde la ventana que contiene el proyecto seleccione la tarjeta perteneciente a la plataforma DaNI y dando clic derecho seleccioné la opción **Connect**, posterior a esto despliegue la carpeta **Ejecutar exploración del robot** y abra el instrumento **MainDaNI.vi**.

3.- Al ejecutar el programa se verificara la recepción de los datos de los distintos sensores que contiene el robot. En caso de que no exista una recepción de datos, desconectar el robot y repetir el proceso anterior.

4.- A continuación con el Joypad conectado ejecutar el instrumento **HostPrincipal.vi** localizado en la sección **My Computer** en la carpeta **Robot teleoperado para exploración**.

5.- Desplazar el robot empleando los botones del Joypad y verificar el funcionamiento conjunto de los sensores.

E

Norma NMEA 0183

La información que reciben los receptores está normalizada por la *National Marine Electronics Association*, ésta es una organización sin fines de lucro, la cual está compuesta por instituciones académicas, distribuidores y fabricantes de dispositivos utilizados para la comunicación marina. Este organismo realizó la norma NMEA 0183 la cual sea emplea en la recepción y transmisión de datos, de instrumentos marinos y receptores GPS.

Configuración serial

Los parámetros para la transmisión de datos a través del protocolo serial se encuentran normalizados. Los valores definidos para la comunicación son los siguientes:

- **Baud Rate:** 4800.
- **Numero de bits de Datos:** 8(Bit 7 es 0).
- **Bits de alto:** 1(o mas).
- **Paridad:** Sin paridad.

Formato de datos

Esta norma señala el formato en que se enviarán los datos, esto es de la siguiente forma:

$ttsss,d1,d2,..., ⟨CR⟩⟨LF⟩

Las primeras dos letras seguidas del signo '$' son los identificadores del emisor. Los siguientes tres caracteres *sss*, son los identificadores de la oración, seguidos por un número de campos de datos separados por comas, opcionalmente seguido por una 'suma de chequeo', y terminado por un retorno de carro/salto de linea ⟨CR⟩/⟨LF⟩. Estos campos de datos están únicamente definidos para cada tipo de oración. Un ejemplo de este tipo de sentencias es el siguiente:

$HCHDM,238,M,⟨CR⟩⟨LF⟩

Donde:

HC corresponde a una brújula magnética.
HDM corresponde a cabecera magnética del mensaje.
238 es el valor de la cabecera.
M denota que el valor es magnético.

Oraciones NMEA

En esta norma se mencionan las oraciones para enviar y recibir información, las cuales en el caso del GPS son muy variadas y se emplean para distintos propósitos, por lo que en este apartado, solo se mencionarán las oraciones que envía el receptor empleado en el proyecto.

GGA(Global Positioning System Fixed Data): Esta oración contiene datos acerca de la posición y precisión tridimensional(latitud,longitud y altitud).
Ejemplo:

$GPGGA,002153.000,3342.6618,N,11751.3858,W,1,10,1.2,27.0,M,-34.2,M,,0000*5E⟨CR⟩⟨LF⟩

Nombre	Ejemplo	Unidad	Mensaje
Identificador de Mensaje	$GPGGA		Cabecera de protocolo GGA
Tiempo UTC	002153.000		hhmmss.sss
Latitud	3342.6618		ddmm.mmmm
Indicador N/S	N		N= Norte , S = Sur
Longitud	11751.3858		dddmm.mmmm
Indicador Este/Oeste	W		E=Este, W=Oeste
Indicador de posición fija	1		0-Fijación no disponible o inválida 1-Modo GPS SPS valido, fijación válida 2-Modo GPS Diferencial
Satélites usados	10		Valor entre 0-12
HDOP	1.2		Dilución horizontal de precisión.
Altitud sobre el nivel del mar (MSL)	27.0	metros	
Unidades	M	metros	
Separación Geoidal	-34.2	metros	Separación Geoide a elipsoide Altitud elipsoidal = Altitud MSL + Separación Geoidal
Unidades	M	metros	
Etapas de correlación diferencial		segundos	Este campo se encuentra vacío cuando no se usa en modo diferencial.
Identificador de la diferencia de la estación de referencia.	0000		
Chequeo de suma	*5E		
⟨CR⟩⟨LF⟩			Terminación de fin de mensaje

Tabla E.1 Datos oración GGA.

GSA(GNSS DOP and Active Satelites): Esta oración provee detalles acerca de la naturaleza de la posición. Eso incluye el número de satélites que están siendo usados en la actual solución y la DOP (Dilución de Precisión). La DOP es una indicación del efecto de la geometría satelital y la resolución de la posición:
Ejemplo:

$GPGSA,A,3,07,02,26,27,09,04,15, , , , , ,1.8,1.0,1.5*33⟨CR⟩⟨LF⟩

Nombre	Ejemplo	Unidad	Descripción
ID Mensaje	$GPGSA		Cabecera del protocolo GSA
Modo 1	A		M= Manual (Forzado a operar en 2D o 3D) A=Automático en 2D
Modo 2	3		1= Posición no Disponible 2= 2D (<4 satélites usados) 3=3D(>3 satélites usados)
Satélite Usado(1)	07		Satélite en canal 1
Satélite Usado(1)	02		Satélite en canal 2
...			...
Satélite Usado(1)			Satélite en canal 12
PDOP(2)	1.8		Dilución de la Precisión (Posición)
HDOP(2)	1.0		Dilución de la Precisión(Horizontal)
VDOP(2)	1.5		Dilución de la Precisión (Vertical)
Chequeo de la suma	*33		
⟨CR⟩⟨LF⟩			Terminación de fin de mensaje.

Tabla E.2 Datos oración GSA

1.- Satélite usado en la solución.
2.- El valor máximo de DOP reportado es 50. Cuando este valor se presenta, el error actual puede ser demasiado grande.

GSV (GNSS Satellites in View) Esta oración muestra información sobre los satélites que se encuentran en posibilidad de ser encontrados a la vista del dispositivo y en el almanaque de datos. Cabe hacer mención que una oración GSV sólo puede proveer información para uno de los cuatro satélites, por lo tanto son necesarias tres oraciones más para una información completa.
Ejemplo:

$GPGSV,2,1,07,07,79,048,42,02,51,062,43,26,36,256,42,27,27,138,42*71
$GPGSV,2,2,07,09,23,313,42,04,19,159,41,15,12,041,42*41⟨CR⟩⟨LF⟩

Nombre	Ejemplo	Unidad	Descripción
ID del Mensaje	$GPGSV		Cabecera del protocolo GSV
Numero de mensajes (1)	2		Número total de mensajes GSV a ser enviado en este grupo.
Numero de mensaje(1)	1		Número de mensaje en el grupo
Satélites a la vista(1)	07		
ID del satélite	07		Canal 1(Rango de 1 a 32)
Elevación	79	grados	Canal 1 (Máximo 90)
Azimut	048	grados	Canal 1(Verdadero, Rango de 0 a 359)
SNR (C/N0)	42	dBHz	Rango de 0 a 99
			0 cuando no hay rastreo
...			...
ID del satélite	27		Canal 4
Elevación	27	grados	Canal 4
Azimut	138	grados	Canal 4
SNR (C/N0)	42	dBHz	Rango de 0 a 99 ...
Chequeo de suma	*71		
⟨CR⟩⟨LF⟩			Terminación de fin de mensaje

Tabla E.3 Datos oración GSV

RMC (Recommended Minimum Specific GNSSData): Esta oración contiene su propia versión acerca de los datos esenciales (posición, velocidad y tiempo). De ahí se deriva el nombre de su identificador, *Datos Específicos Mínimos Recomendados.*
Ejemplo:

$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598, ,*10⟨CR⟩⟨LF⟩

Nombre	Ejemplo	Unidad	Descripción
ID del Mensaje	$GPRMC		Cabecera de protocolo RMC
UTC Time	161229.487		hhmmss.ss
Status(1)	A		A-Dato Válido V- Dato Inválido
Latitud	3723.2475		ggmm.mmmm
Indicador N/S	N		N-Norte S-Sur
Longitud	12158.3416		ggmm.mmmm
Indicador W/E	W		W-Oeste E-Este
Velocidad sobre la tierra	0.13	nudos	
Curso sobre la tierra	309.62	grados	Verdadero
Fecha	120598		ddmmaa
Variación Magnética(2)		grados	E- Este W-Oeste
Indicador E/W(2)	E		E-Este
Modo	A		A-Autónomo D-DGPS E-DR N-Salida de Datos Inválida R-Posición Gruesa(3) S-Simulador
Chequeo de Suma	*10		
⟨CR⟩⟨LF⟩			Terminación de fin de mensaje

Tabla E.4 Datos oración RMC

1.- Un estado válido corresponde a una correcta recepción de todos los datos. Este incluye el número mínimo de satélites requeridos, cualquier máscara DOP, etc...
2.- No todos los fabricantes son capaces de detectar la declinación magnética.
3.- La posición fue calculada con base en uno o más de los satélites a la vista, obteniendo sus estados mediante los parámetros de un almanaque

F

Código fuente de las aplicaciones realizadas en C#

F.1 Programa para reproducción de audio

```
using System;

namespace Linkaaudio
{
  class Program
  {
    static void Main(string[] args)
    {
      System.Diagnostics.Process.Start("http://192.168.1.116:8080/audio.wav");
    }
  }
}
```

F.2 Programa de apoyo para el procesamiento de video

F.2.1

Código para el formulario principal

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Media;

using AForge;
using AForge.Video.DirectShow;
using AForge.Video.VFW;
using AForge.Video;
using AForge.Vision;
using AForge.Vision.Motion;
using AForge.Imaging;
using AForge.Imaging.Filters;

namespace WindowsFormsApplication1
{
  public partial class Form1 : Form
  {
    //Inicio de variables para el detector de color.
    public Color colorD;
    public int mincolarea = 5000;

    //Incializar fuente de video
```

```
private IVideoSource videoSource = null;

// detector de movimiento
MotionDetector detector = new MotionDetector(null,null);

private int flash = 0;
private double MotionAlarmLevel;

//seleccion de funcion.
public bool movimiento=false,coloract=false;

//Metodo para la captura de imagenes.
private void getimage()
{
  if (VideoPlayer.VideoSource != null)
  {
    Bitmap CurrentFrame = VideoPlayer.GetCurrentVideoFrame();

    if (CurrentFrame != null)
    {
      Currentframeform framewindow = new Currentframeform();
      framewindow.frame = CurrentFrame;
      framewindow.ShowDialog(this);
    }
    if (captureframemovementrdioBtn.Checked) captureframemovementrdioBtn.Checked = false;
  }
  else
  MessageBox.Show("Fuente no disponible");
}

public Form1()
{
  InitializeComponent();
  CheckForIllegalCrossThreadCalls = false;
  GroupColorTools.Enabled = false;
  GroupMovementTools.Enabled = false;
  SetMotionDetectAlg(null);
  movdetecindic.BackColor = Color.Black;
  colorindic.BackColor = Color.White;
  MotionAlarmLevel =  Convert.ToDouble(Sensibilitytxt.Text);
  alarmTimer.Enabled = true;
  colorD = (Color)colorindic.BackColor;

}

//Seleccion de la fuente de video.
private void abrirCamaraToolStripMenuItem_Click(object sender, EventArgs e)
{
  VideoCaptureDeviceForm camaraselect = new VideoCaptureDeviceForm();

  if (camaraselect.ShowDialog(this) == DialogResult.OK)
  {
    VideoCaptureDevice myVideoSource = camaraselect.VideoDevice;

    OpenVideoSource(myVideoSource);
  }
}

private void OpenVideoSource(IVideoSource source)
{
  // Cierra la anterior fuente de video
  CloseVideoSource();

  // Inicia la fuente de video
  VideoPlayer.VideoSource = new AsyncVideoSource(source);
  VideoPlayer.Start();

  //Inicio de ejecucion de timers
  alarmTimer.Start();

  videoSource = source;
}


// Close current video source
private void CloseVideoSource()
{
  // Detiene senal de video
  VideoPlayer.SignalToStop();

  for (int i = 0; (i < 50) && (VideoPlayer.IsRunning); i++)
  {
    Thread.Sleep(100);
  }
  if (VideoPlayer.IsRunning)
  VideoPlayer.Stop();

  //Detiene ejecucion de timers
  alarmTimer.Stop();

  //Elimina algoritmo de reconocimiento
```

```
  if (detector != null)
  detector.Reset();

  VideoPlayer.BorderColor = Color.Black;
}


//Seleccion de accion a realizar
private void VideoPlayer_NewFrame(object sender, ref Bitmap image)
{
  //Deteccion de movimiento
  if (movimiento && !coloract)
  {
    lock (this)
    {
      if (detector != null)
      {
        float motionLevel = detector.ProcessFrame(image);

        //Verifica activacion de la alarma
        if (motionLevel > MotionAlarmLevel)
        {
          // Alarma visual y auditiva
          flash = (int)(2 * (1000 / alarmTimer.Interval));
          SystemSounds.Beep.Play();

          //Captura de imagen si se activa la alarma
          if (captureframemovementrdioBtn.Checked)
          getimage();
        }
      }
    }
  }

  //Deteccion de color

  if (coloract && !movimiento)
  {
    Bitmap video = image;
    Bitmap temp = video.Clone() as Bitmap;

    EuclideanColorFiltering filtro1 = new EuclideanColorFiltering();
    filtro1.CenterColor = new AForge.Imaging.RGB(colorD.R, colorD.G, colorD.B);
    filtro1.Radius = 100;
    filtro1.ApplyInPlace(temp);

    BlobCounter bcount = new BlobCounter();
    bcount.MinHeight = 5;
    bcount.MinWidth = 5;
    bcount.FilterBlobs = true;
    bcount.ObjectsOrder = ObjectsOrder.Size;
    bcount.ProcessImage(temp);

    //Muestra areas donde se ha detectado el color
    Rectangle[] rects = bcount.GetObjectsRectangles();
    foreach (Rectangle recs in rects)
    if (rects.Length>0)
    {
      if (recs.Top*recs.Width >= mincolarea)
      {
        Rectangle Objectrect = recs;
        Graphics g = Graphics.FromImage(video);
        using (Pen pen = new Pen(Color.FromArgb(colorD.R, colorD.G, colorD.B), 2))
        {
          g.DrawRectangle(pen, Objectrect);
        }
        g.Dispose();
      }
    }

  }
}

//Cierra fuente de video
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
  this.CloseVideoSource();
}


//Fija el nuevo algoritmo para la deteccion de movimiento
private void SetMotionDetectAlg(IMotionDetector detectionAlgorithm)
{
  lock (this)
  {
    detector.MotionDetectionAlgorithm = detectionAlgorithm;
  }
}

//Seleccion de algoritmo para deteccion
private void twoFramesDifferenceToolStripMenuItem_Click(object sender, EventArgs e)
```

```
{
  SetMotionDetectAlg(new TwoFramesDifferenceDetector());
}

private void simpleBackgroundModelingToolStripMenuItem_Click(object sender, EventArgs e)
{
  SetMotionDetectAlg(new SimpleBackgroundModelingDetector(true, true));
}

// Fija el metodo para la visualizacion de movimiento
private void SetMotionViewAlg(IMotionProcessing processingAlgorithm)
{
  lock (this)
  {
    detector.MotionProcessingAlgorithm = processingAlgorithm;
  }
}

//Habilita controles para la deteccion de color.
private void colorToolStripMenuItem_Click(object sender, EventArgs e)
{
  alarmTimer.Enabled = false;
  SetMotionDetectAlg(null);
  SetMotionViewAlg(null);
  GroupMovementTools.Enabled = false;
  GroupColorTools.Enabled = true;
  movimiento = false;
  coloract = true;
  lessvalarcolorTxtbox.Text = mincolarea.ToString();

}

//Elimina cualquier tarea de procesamiento
private void nigunoToolStripMenuItem_Click(object sender, EventArgs e)
{
  SetMotionDetectAlg(null);
  SetMotionViewAlg(null);
}

//Habilita controles para la deteccion de movimiento
private void movimientoToolStripMenuItem_Click(object sender, EventArgs e)
{
  if (alarmTimer.Enabled == false) alarmTimer.Enabled = true;
  GroupColorTools.Enabled = false;
  Btnalgproc1.Checked = true;
  GroupMovementTools.Enabled = true;
  movimiento = true;
  coloract = false;
}

//Fija metodo para la visualizacion de movimiento
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
  SetMotionViewAlg(new GridMotionAreaProcessing(32, 32));
}

private void Btnalgproc1_CheckedChanged(object sender, EventArgs e)
{
  SetMotionViewAlg(new MotionAreaHighlighting());
}

private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
  SetMotionViewAlg(new MotionBorderHighlighting());
}

private void alarmTimer_Tick(object sender, EventArgs e)
{
  if (flash != 0)
  {
    movdetecindic.BackColor = (flash % 2 == 1) ? Color.Black : Color.Green;
    flash--;
  }
}

//Fija la sensibilidad de los algoritmos de deteccion.
private void Addsensbtn_Click(object sender, EventArgs e)
{
  if (MotionAlarmLevel <= 0.1)
  {
    MotionAlarmLevel += 0.01;
    Sensibilitytxt.Text = MotionAlarmLevel.ToString();
  }
}

private void lesssenbtn_Click(object sender, EventArgs e)
{
  if (MotionAlarmLevel >= 0.01)
  {
    MotionAlarmLevel -= 0.01;
    Sensibilitytxt.Text = MotionAlarmLevel.ToString();
```

```
      }
    }

    private void lessareasensBtn_Click(object sender, EventArgs e)
    {
      if (mincolarea > 0)
      {
        mincolarea -= 100;
        lessvalarcolorTxtbox.Text = mincolarea.ToString();
      }
    }

    private void moreareasensBtn_Click(object sender, EventArgs e)
    {
      if (mincolarea < 10000)
      {
        mincolarea += 100;
        lessvalarcolorTxtbox.Text = mincolarea.ToString();
      }
    }


    //Selecciona el color a detectar.
    private void Selectcolorbtn_Click(object sender, EventArgs e)
    {
      ColorDialog colorchoose = new ColorDialog();
      colorchoose.ShowDialog();
      colorD = (Color)colorchoose.Color;
      colorindic.BackColor = colorD;
    }

    //Evento para la captura de la imagen recibida.
    private void getimageBtn_Click(object sender, EventArgs e)
    {
      getimage();
    }
  }
}
```

F.2.2

Código del formulario para captura de imagen

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
  public partial class Currentframeform : Form
  {

    public Bitmap frame
    {
      set { pictureframebox.Image = value; }
    }
    public Currentframeform()
    {
      InitializeComponent();
    }

    private void guardaImagenToolStripMenuItem_Click(object sender, EventArgs e)
    {
      SaveFileDialog savepicture = new SaveFileDialog();
      savepicture.Filter = "JPEG(*.JPG)|*.JPG|BMP(*.BMP)|*.BMP|PNG(*.PNG)|*.PNG";
      Image currentframe = pictureframebox.Image;
      savepicture.ShowDialog();

      try { currentframe.Save(savepicture.FileName); }
      catch { }
    }
  }
}
```

Bibliografía y Referencias

[1] OLLERO, Anibal , *Robótica: Manipuladores y robots móviles.*, Marcombo-Boixareu Editores, 2001.

[2] NIEMEYER, Günter; PREUSCHE, Carsten; HIRZINGER, Gerd, *Telerobotics. En Springer handbook of robotics.*, Springer Berlin Heidelberg, 2008. p. 741-757.

[3] SIEGWART, Roland; NOURBAKHSH, Illah Reza; SCARAMUZZA, Davide, *Introduction to autonomous mobile robots.*, MIT press, 2011.

[4] TZAFESTAS, Spyros G., *Introduction to mobile robot control.*, Elsevier, 2013.

[5] ARANDA BRICAIRE, Eduardo; SALGADO JIMÉNEZ, Tomás; VELASCO VILLA, Martín. *Control no lineal discontinuo de un robot móvil.* Computación y Sistemas, 2002.

[6] DUDEK, Gregory; JENKIN, Michael. *Inertial sensors, GPS, and odometry.* En Springer Handbook of Robotics. Springer Berlin Heidelberg, 2008. p. 477-490.

[7] National Instruments, *LabVIEW User Manual* [En linea], Disponible en: «*http://www.ni.com/pdf/manuals/320999d.pdf*», [Consultado: Julio 2015].

[8] DEITEL, Harvey M.; DEITEL, Paul J., *Como programar en C#.*, Pearson Education, 2007.

[9] Aforge.NET Framework, *Aforge.NET Framework Documentation* [En linea], Disponible en «*http://www.aforgenet.com/framework/docs/*» [Consultado: Mayo 2015].

[10] National Instruments, *NI LabVIEW Robotics Starter Kit Robotics Platform for Teaching, Research, and Prototyping* [En linea], Disponibles en: « *http://www.ni.com/datasheet/pdf/en/ds-217*» [Consultado: Junio 2015].

[11] KING, Robert, *Mobile Robotics Experiments with DaNI* [En linea], Disponible en: «*http://download.ni.com/pub/devzone/epd/mobile_robotics_experiments.pdf*» [Consultado: Febrero 2015].

[12] National Instruments, *USER GUIDE NI sbRIO-961x/963x/964x and NIsbRIO-9612XT/9632XT/9642XT.* [En linea], Disponible en: «*http://www.ni.com/pdf/manuals/375052c.pdf*» [Consultado: Junio 2015].

[13] PITSCO Education, *Technical Specs - TETRIX MAX DC Gear Motor 39530* [En linea], Disponibles en: «*https://c10645061.ssl.cf2.rackcdn.com/resources/tetrix/152motor739530.pdf*», [Consultado: Junio 2015].

[14] Dimension Engineering, *Sabertooth 2x10 RC User's Guide* [En linea], Disponible en: «*http://www.dimensionengineering.com/datasheets/Sabertooth2x10RC.pdf*»,[Consultado: Junio 2015].

[15] PITSCO Education, *NI Encoder Kit* [En linea], Disponible en: «*http://www.pitsco.com/NI_Encoder_Kit*», [Consultado: Junio 2015].

[16] Parallax INC, *PING))) Ultrasonic Distance Sensor (#28015)* [En linea], Disponible en: «*https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf*», [Consultado: Junio 2015].

[17] Hitec multiplex, *HS-485HB Deluxe HD Ball Bearing Standard Servo* [En linea], Disponible en: «*http://hitecrcd.com/products/servos/sport-servos/analog-sport-servos/hs-485hb-deluxe-hd-ball-bearing-servo/product*», [Consultado: Junio 2015].

[18] HERRERA, Guillermo *Navegación Autónoma de un Robot Móvil*, Facultad de Ingeniería UNAM, 2014.

[19] Olimex, *TECHNICAL DATA MQ-135 GAS SENSOR* [En linea], Disponible en: «*https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf*»

[20] Measurement Specialties, Inc., *Piezo Film Sensors Technical Manual* [En linea], Disponible en: «*https://www.sparkfun.com/datasheets/Sensors/Flex/MSI-techman.pdf*», [Consultado: Julio 2015].

[21] National Semiconductors., *Precision Centigrade Temperature Sensors* [En linea], Disponible en: «*http://pdf.datasheetcatalog.net/datasheet/nationalsemiconductor/DS005516.PDF*», [Consultado: Junio 2015].

[22] RAI, Anil. *INTRODUCTION TO GLOBAL POSITIONING SYSTEM.*

[23] Department of Defense. United states of america *GLOBAL POSITIONING SYSTEM STANDARD POSITIONING SERVICE PERFORMANCE STANDARD* [En linea], Disponible en: «*http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf*», [Consultado: Julio 2015].

[24] STALLINGS, William. *Comunicaciones y Redes de Computadores.*, Pearson Prentice Hall, 2004.

[25] GUZMÁN, Leonardo Enrique Solaque; VILLA, Manuel Alejandro Molina; VÁSQUEZ, Edgar Leonardo Rodríguez. *SEGUIMIENTO DE TRAYECTORIAS CON UN ROBOT MÓVIL DE CONFIGURACIÓN DIFERENCIAL.*, Revista Ingenierias USBmed, 2015, vol. 5, no 1, p. 26-34.

[26] Maxim Integrated *MAX3222/MAX3232/MAX3237/MAX3241* 3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1µF External Capacitors* [En linea], Disponible en «*http://pdfserv.maximintegrated.com/en/ds/MAX3222-MAX3241.pdf*», [Consultado: Julio 2015].