ROBOT MOVIL CONTROLADO POR DISPOSITIVO LOGICO PROGRAMABLE

JUAN SEBASTIAN DIAZ RUEDA

XAVIER ORLANDO RODRIGUEZ QUIÑONEZ

UNIVERSIDAD AUTONOMA DE BUCARAMANGA

FACULTAD DE INGENIERIAS FISICO MECANICAS

INGENIERIA MECATRONICA

BUCARAMANGA

2007

ROBOT MOVIL CONTROLADO POR DISPOSITIVO LOGICO PROGRAMABLE

Director

Ing. NAYIBE CHIO CHO

JUAN SEBASTIAN DIAZ RUEDA

XAVIER ORLANDO RODRIGUEZ QUIÑONEZ

UNIVERSIDAD AUTONOMA DE BUCARAMANGA

FACULTAD DE INGENIERIAS FISICO MECANICAS

INGENIERIA MECATRONICA

BUCARAMANGA

2007

Nota de aceptación

_____

_____

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

_____

Bucaramanga, Mayo 22 de 2.007

A nuestros padres, con su apoyo, siempre adelante.

A nuestros amigos, compañeros en la vida.

AGRADECIMIENTOS

Los autores expresamos los más sinceros agradecimientos a las siguientes personas:

De manera especial, a la directora de proyecto, la ingeniera Nayibe Chio Cho, por la asesoría y la colaboración prestada desde el momento en que se gestó la idea de este proyecto y durante su desarrollo.

El docente Diego Tibaduiza Burgos, director del laboratorio de electrónica, por la colaboración con los equipos y el espacio de trabajo, además de su asesoría

Nuestros amigos, Carlos Fernando Rueda Trujillo y Rafael Enrique Montaño Salcedo, por las ideas y sugerencias que nos brindaron.

El ingeniero Jhon Faber Archila, el ingeniero Javier Eduardo Jurado Salcedo, Mauricio Suárez y Marvin Torres, por la asesoría y ayuda brindada para el desarrollo de este proyecto.

Y ante todo a nuestros padres, incansables en su apoyo para lograr nuestros objetivos y continuar adelante en la vida profesional.

TABLA DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE FIGURAS

LISTADO DE ANEXOS

INTRODUCCION

En robótica móvil existe gran cantidad de trabajos enfocados a diferentes tareas, entre algunas de estas tenemos, la navegación entre laberintos, la recolección de objetos o el reconocimiento de entornos. En este caso la tarea consiste en desplazarse de un punto a otro en un plano, evadiendo los obstáculos, en caso de que estos existan.

Dos características diferencian este trabajo de otros, una de ellas, la ausencia de algún mapa o guía para lograr la trayectoria, ya que solo se tiene los dos puntos identificados por medio de coordenadas cartesianas. La otra característica es la forma de controlar el robot, utilizando un FPGA o arreglo de compuertas lógicas de la marca Xilinx, y perteneciente a la familia Spartan 3E.

Este FPGA viene montado en una tarjeta de desarrollo en la cual se conecta a diferentes interfaces y dispositivos, algunos de estos utilizados para el funcionamiento del robot, como lo son: botones, conectores, pantalla LCD, conversor digital-análogo, entre otros. El FPGA es programado utilizando el lenguaje de descripción hardware VHDL, mediante el cual se especifican las funciones que deben realizar los componentes del robot para luego ser sintetizadas en el formato que entiende el FPGA utilizando la herramienta de síntesis proveída con la tarjeta de desarrollo.

La implementación de FPGA ofrece la gran ventaja del procesamiento en paralelo. De esta forma se reciben las señales de los sensores e inmediatamente se efectúan las acciones de control sin necesidad de esperar a que terminen otras secuencias.

El desarrollo de este libro esta dividido en tres capítulos, en el primero se presenta el marco teórico, en el cual se compila de manera resumida los conceptos aplicados en el diseño del robot.

El segundo capítulo refiere al diseño del robot, empezando por una ilustración de la secuencia de diseño y continuando con la explicación de cada uno de los pasos de esta secuencia, que incluye: la selección el tipo de robot, diseño mecánico, interfaces con sensores y actuadores, programación del robot para terminar con la interfaz usuario.

En el tercer capítulo se presenta la construcción y acople de los elementos mecánicos y electrónicos del robot.

Finalmente, se presentan las conclusiones del desarrollo del presente proyecto.

PLANTEMIENTO DEL PROBLEMA

La situación que se desea afrontar se sustenta básicamente en la necesidad de implementar las nuevas tecnologías en lo que a dispositivos programables se refiere, en donde existe una variada oferta que va más allá de los microcontroladores y ofrece grandes ventajas. Bien es sabido que la tecnología avanza a pasos agigantados y en el desarrollo como ingenieros y profesionales es importante tener la capacidad de aplicar los avances tecnológicos de una forma adecuada y coherente, pues bien, con este proyecto se pretende dar el primer paso en la universidad e incluso en la región como un proyecto de investigación en la aplicación de dispositivos lógicos programables (PLD's) que junto con lenguajes de programación y descripción de hardware como el VHDL son las herramientas mas utilizadas a nivel industrial y en el proceso de diseño de sistemas electrónicos complejos. Un aspecto importante que vale la pena destacar radica en el hecho de que en la facultad ya se encuentran aprobados recursos para proyectos de investigación relacionados a la implementación de estos dispositivos.

El concepto de redes neuronales es otro de los aspectos a tratar en este proyecto, esencialmente porque es otro punto que hasta ahora no se ha trabajado, y de un modo u otro hacia allá se dirige la computación, la programación y los modelos de inteligencia artificial (IA), en este modelo también conocido como modelo de computación conexionista se basará la concepción del control y la toma de decisiones de la "inteligencia" del robot en su funcionamiento.

El problema a solucionar consiste básicamente en concebir una forma muy sencilla de inteligencia artificial utilizando el concepto de redes neuronales e implementando dispositivos lógicos programables y lenguaje VHDL, mediante el cual un pequeño robot móvil tenga la capacidad de identificar y memorizar una forma de ir de un punto a otro en un plano, es decir, una trayectoria. Posterior al

aprendizaje de esta trayectoria, el robot deberá estar en la capacidad de recorrerla de forma eficiente y continua hasta que se le indique el aprendizaje de una nueva trayectoria, punto el cual borrara la trayectoria anterior. Para el funcionamiento del prototipo se dispone de dos etapas, una primera de aprendizaje, en la que se le aplican como entradas las coordenadas de los dos puntos, aquí entonces se encargará de reconocer la trayectoria; en una segunda etapa o modo de operación el robot se desplazara por la trayectoria que se encuentre almacenada en su memoria.

La importancia de este proyecto radica, como ya se ha explicado anteriormente, en el hecho de dar un primer paso a la implementación de nuevas tecnologías y conceptos basándonos en la investigación y aprovechando los recursos que ofrece la universidad y la facultad en este aspecto, con un proyecto de investigación ya aprobado.

OBJETIVOS

OBJETIVO GENERAL

Diseñar, construir e implementar un prototipo de robot móvil que funcione en dos etapas, en la primera que tenga la capacidad de identificar y aprender una trayectoria entre dos puntos dados en una superficie plana, para posteriormente en la segunda etapa de su funcionamiento recorrer la trayectoria aprendida almacenada en su memoria.

OBJETIVOS ESPECIFICOS

Realizar el diseño mecánico del robot móvil, seleccionando la configuración adecuada según los movimientos y trayectorias que deberá seguir el robot

Construir el robot y dimensionarlo para que pueda operar en superficies pequeñas, a la vez tenga el espacio suficiente para albergar los elementos mecánicos, además de los dispositivos y el montaje electrónico propios de su funcionamiento.

Seleccionar e implementar uno o varios sensores para detectar y esquivar los obstáculos que se encuentren entre los dos puntos a recorrer por parte del robot.

Implementar el lenguaje VHDL en dispositivos lógicos programables, hardware en el cual se lleve a cabo el proceso de control del robot, según las directrices establecidas para tal fin.

Investigar y aplicar el concepto de redes neuronales para la concepción de la estrategia de control y el aprendizaje de la trayectoria.

ANTECEDENTES

Este es un proyecto que pretendía aplicar nuevas tecnologías y conceptos, contó además del apoyo de los autores con el de la Universidad Autónoma de Bucaramanga y la Facultad de Ingenierías físico mecánicas, especialmente con el proyecto de investigación que le fue aprobado a la Ing. Nayibe Chio Cho en su momento, y con el cual se contó con recursos para implementar estas tecnologías.

En cuanto a los robots móviles, se tiene ya una experiencia en la facultad, en la cual se han desarrollado diferentes prototipos a nivel de proyectos de grado y proyectos integradores, de aquí se destaca la importancia de aprovechar la experiencia que se tiene en este sentido en cuanto al diseño y construcción del prototipo.

VHDL EN LA ACTUALIDAD

La actividad que se ha generado en torno a VHDL es muy intensa. En muchos países como España se han creado grupos de trabajo alrededor de dicho lenguaje y se realizan periódicamente conferencias, reuniones, etc., donde se presentan trabajos tanto en Estados Unidos (en el VIUF, VHDL International User's Forum) como en Europa (VHDL Forum for CAD in Europe), así como en el congreso EuroVHDL celebrado desde 1992.

El proceso de estandarización del VHDL no se detuvo con la primera versión del lenguaje (VHDL '87), sino que ha continuado con la nueva versión (VHDL '93) y constantes actualizaciones, mejoras y metodologías de uso. Entre estas adiciones 0 actualizaciones se encuentra una muy importante: la extensión analógica, que permite la utilización de un lenguaje único en todas las tareas de especificación, simulación y síntesis de sistemas electrónicos digitales, analógicos o mixtos.

CAMPO DE APLICACIÓN DE LA LÓGICA PROGRAMABLE

La lógica programable es una herramienta de diseño muy poderosa, que se aplica en el mundo industrial y en proyectos universitarios en todo el mundo. En la actualidad se usan desde los PLD más sencillos (como el GAL, PAL, PLA) como reemplazos de circuitos LSI y MSI, hasta los potentes CPLD y FPGA, que tienen aplicaciones en áreas como telecomunicaciones, computación, redes, medicina, procesamiento digital de señales, multiprocesamiento de datos, microondas, sistemas digitales, telefonía celular, filtros digitales programables, entre otros.

En general, los CPLD son recomendables en aplicaciones donde se requieren muchos ciclos de sumas de productos, ya que pueden introducirse en el dispositivo para ejecutarse al mismo tiempo, lo que conduce a pocos retrasos. En la actualidad, los CPLD son muy utilizados a nivel industrial, ya que resulta fácil convertir diseños compuestos por múltiples PLD sencillos en un circuito CPLD. Por otro lado, los FPGA son recomendables en aplicaciones secuenciales que no suponen grandes cantidades de términos producto. Por ejemplo, los FPGA desarrollados por la compañía A TMEL ofrecen alta velocidad en cómputo intensivo, aplicaciones en procesadores digitales de señales (DSP) y en otras fases del diseño lógico, debido a la gran cantidad de registros con los que cuentan sus dispositivos (de 1024 a 6400). Esto los hace ideales para su uso en dichas áreas.

1 MARCO TEORICO

1.1 ROBOT MÓVIL

1.1.1 Locomoción [1]

Existe una gran variedad de modos de moverse sobre una superficie sólida; entre los robots, las más comunes son las ruedas, las cadenas y las patas.

Los vehículos de ruedas son, con mucho, los más populares por varias razones prácticas. Los robots con ruedas son más sencillos y más fáciles de construir, la carga que pueden transportar es mayor, relativamente. Tanto los robots basados en cadenas como en patas se pueden considerar más complicados y pesados, generalmente, que los robots de ruedas para una misma carga útil. A esto podemos añadir el que se pueden transformar vehículos de ruedas de radio control para usarlos como bases de robots.

Los vehículos con ruedas son una solución muy eficaz por muchas razones prácticas. Los robots con ruedas son muy sencillos de construir, la carga que pueden transportar es relativamente mayor. Las configuraciones basadas en cadenas y patas generalmente son consideradas más pesadas y complicadas.

La principal desventaja de las ruedas es su empleo en terreno irregular, en el que se comportan bastante mal. Normalmente un vehículo de ruedas podrá sobrepasar un obstáculo que tenga una altura no superior al radio de sus ruedas, entonces una solución es utilizar ruedas mayores que los posibles obstáculos a superar; sin embargo, esta solución, a veces, puede no ser práctica.

[1] http://cfievalladolid2.net/tecno/ctrl_rob/robotica/movil.htm

Para robots que vayan a funcionar en un entorno natural las cadenas son una opción muy buena porque las cadenas permiten al robot superar obstáculos relativamente mayores y son menos susceptibles que las ruedas de sufrir daños por el entorno, como piedras o arena. El principal inconveniente de las cadenas es su ineficacia, puesto que se produce deslizamiento sobre el terreno al avanzar y al girar. Si la navegación se basa en el conocimiento del punto en que se encuentra el robot y el cálculo de posiciones futuras sin error, entonces las cadenas acumulan tal cantidad de error que hace inviable la navegación por este sistema. En mayor o menor medida cualquiera de los sistemas de locomoción contemplados aquí adolece de este problema.

Potencialmente los robots con patas pueden superar con mayor facilidad que los otros los problemas de los terrenos irregulares. A pesar de que hay un gran interés en diseñar este tipo de robots, su construcción plantea numerosos retos. Estos retos se originan principalmente en el gran número de grados de libertad que requieren los sistemas con patas. Cada pata necesita como mínimo un par de motores lo que produce un mayor costo, así como una mayor complejidad y menor fiabilidad. Es más los algoritmos de control se vuelven mucho más complicados por el gran número de movimientos a coordinar, los sistemas de patas son un área de investigación muy activo.

A continuación se presentan las características más significativas de los sistemas de locomoción más comunes en robots móviles:

- Diferencial

La configuración diferencial se presenta como la más sencilla de todas. Consta de de dos ruedas situadas diametralmente opuestas en un eje perpendicular a la dirección del robot. Cada una de ellas dotada de un motor, de forma que los giros se realizan dándole diferentes velocidades. Así si queremos girar a la derecha,

daremos mayor velocidad al motor izquierdo. Para girar a la izquierda, será el motor derecho el que posea mayor velocidad.

Con dos ruedas es imposible mantener la horizontalidad del robot. Se producen cabeceos al cambiar la dirección. Para resolver este problema, se colocan ruedas "locas". Estas ruedas no llevan asociadas ningún motor, giran libremente según la velocidad del robot. Además, pueden orientarse según la dirección del movimiento, de forma análoga a como lo hacen las ruedas traseras de los carritos del supermercado. En la Figura 1 se puede observar la configuración.

Figura 1. Configuración diferencial

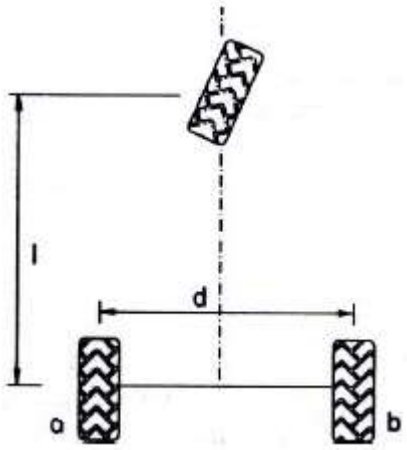
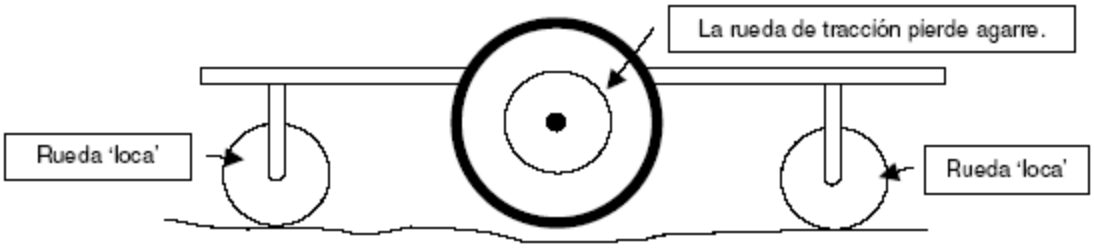


Figura 2. Pérdida de tracción por exceso de apoyos



Dependiendo de las necesidades, se puede colocar una o dos ruedas “locas”. Sin embargo, la presentación de mas de tres apoyos en el robot (incluidas las dos ruedas de tracción), pueden llevar a grandes cálculos de odometría en terrenos irregulares, e incluso a perdida de la tracción total. En la Figura 2 se aprecia como la rueda de tracción pierde agarre, haciendo imposible el avance del motor.

Para llevar a cabo una navegación por odometría, es necesario acoplar a los motores de las ruedas laterales sensores o encoder, de forma que contando los pulsos que avanza cada rueda y teniendo en cuenta el radio de la misma y la reducción del motor, no hay mas que aplicar las ecuaciones cinemáticas del robot para hallar la posición exacta en la que se encuentra y el ángulo de desviación respecto a una dirección de referencia.

- Triciclo

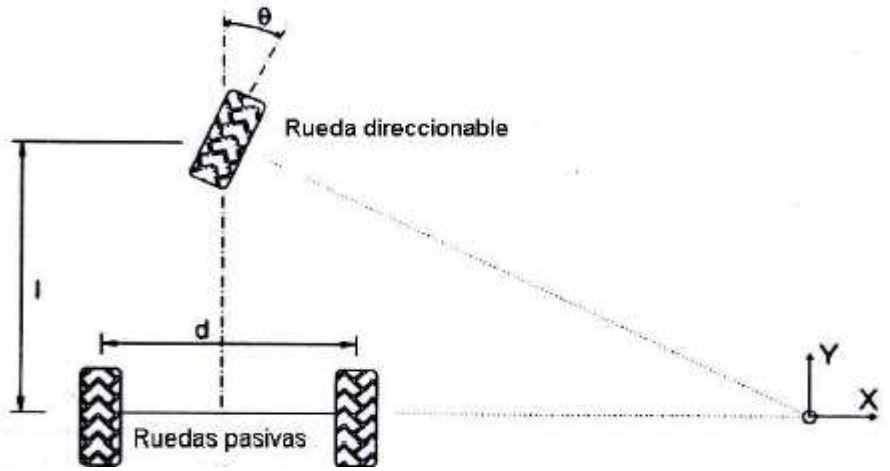
En este caso, se dispone de tres ruedas en el robot, situadas de forma similar a los triciclos de lo niños, de ahí su nombre. Tendremos por tanto, dos ruedas traseras, que no lleven acopladas ningún motor. La tracción estará en la rueda delantera, que además, será la que usaremos para dirigir al robot. (Ver grafico)

En este caso, el cálculo de la odometría es mucho más sencillo. La posición del robot vendrá dada por el numero de pulsos que avanza el encoger de la rueda motora, y la dirección es simplemente la que lleve dicha rueda.

Un problemas asociado a esta configuración es que el centro de gravedad tiene que alejarse de de la rueda de tracción en terrenos inclinados cuando el robot lleva dirección de subida. Esto se traduce en una perdida de la tracción del robot. Al perderse el contacto con el suelo la rueda motora sigue girando, pero el robot no avanza. Esto supone un error grande al hacer el cálculo de la odometría, ya

que el robot indica que esta en un punto más avanzado, cuando en realidad se encuentra más atrás.

Figura 3. Configuración triciclo



Ollero, A. Robótica: Manipuladores y robots móviles

- Ackerman

Se usa exclusivamente en la industria del automóvil. Es la configuración que llevan los automóviles, dos ruedas de tracción traseras, dos ruedas de dirección delanteras. Esta configuración esta diseñada para que en un giro la rueda delantera interior tenga un ángulo ligeramente mas agudo que la exterior, y evitar así el derrape de las ruedas.

Como se puede apreciar en la Figura 4, las normales a ambas ruedas se cortan en un punto, que se encuentra sobre la prolongación del eje de las traseras. Así, se puede comprobar que las trayectorias de ambas ruedas para ángulos de giro constantes son circunferencias concéntricas.

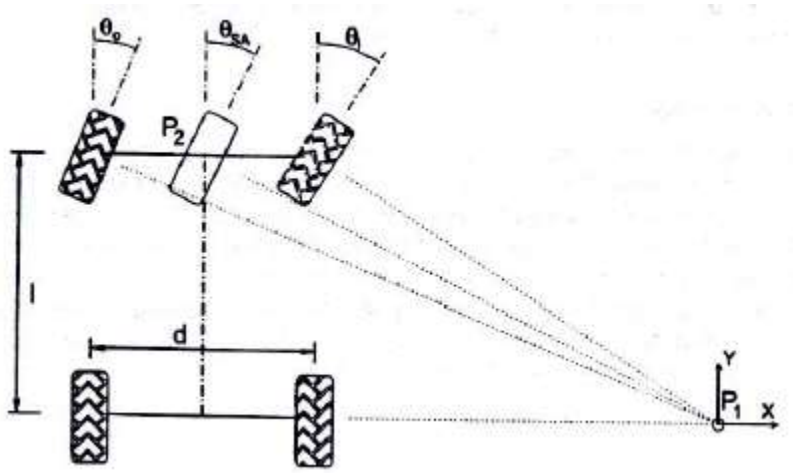
La relación entre los ángulos de las ruedas de dirección viene dada por la ecuación de Ackerman

$$\cot\theta_i - \cot\theta_0 = \frac{d}{l} \qquad (1.1)$$

Donde:

$\theta_i =$ Angulo relativo de la rueda interior
$\theta_0 =$ Angulo relativo de la rueda exterior
$l =$ Separación longitudinal entre ruedas
$d =$ Separación lateral entre ruedas

Figura 4. Configuración Ackerman



Ollero, A. Robótica: Manipuladores y robots móviles

La configuración de Ackerman da una solución bastante precisa para la odometría a la vez que se constituye un buen sistema de tracción incluso con terrenos inclinados. No obstante, su construcción mecánica se complica de forma exponencial respecto de las demás.

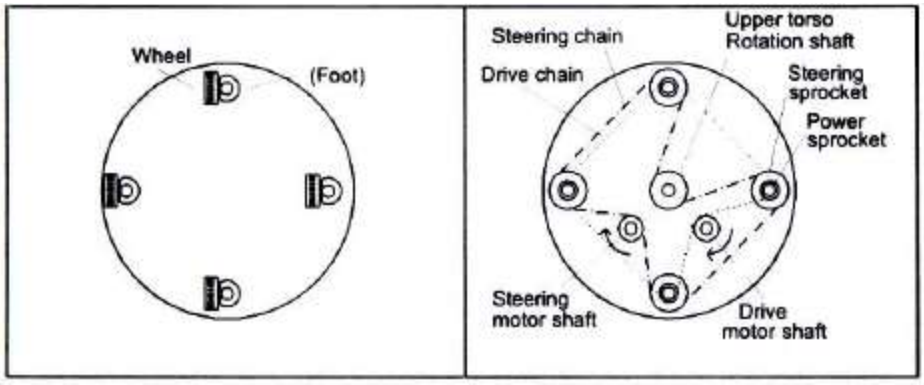
- Síncrono

Una configuración innovadora, tiene 3 o más ruedas mecánicamente acopladas de tal manera que rotan en la misma dirección, a la misma velocidad y pivotan al unísono sobre sus respectivos ejes de dirección cuando ejecutan un giro. Esta

configuración y sincronización de la dirección da lugar a mejorada precisión de la odometría a través de reducido derrape, ya que todas las ruedas generan fuerzas iguales y paralelas en todo momento.

La requerida sincronización mecánica puede ser realizada de un número distinto de formas, la más común es una cadena o correa o tren de engranajes.

Figura 5. Configuración síncrono



Como vemos en la figura anterior la salida rotacional del motor de dirección esta mecánicamente acoplada a través de una correa a cada uno de los ejes de giro de los conjuntos conformados cada uno por una rueda y un motor de tracción (en este caso 4) permitiéndoles pivotar de forma síncrona a través de los 360 grados y por lo tanto siempre apuntan en la dirección de avance.

Una configuración de 3 puntos asegura buena estabilidad y tracción, mientras que unas ruedas de gran diámetro proveen más que una adecuada capacidad para sobrepasar obstáculos. Las desventajas de esta configuración en particular incluyen errores de odometría introducidos por las correas y por las fuerzas de rozamiento con el suelo cuando se pivota en el sitio.

Los cálculos de odometría son sencillos, la orientación del vehículo es simplemente derivada del encoder del ángulo directriz mientras que el desplazamiento en la dirección de movimiento es dado por la fórmula

$$D = \frac{2\pi N}{C} R \qquad (1.2)$$

Donde:

D = Desplazamiento del vehiculo a lo largo del camino.
N = Medida del encoder del eje del motor motriz.
C = Medida del encoder por vuelta completa de la rueda

1.1.2 Modelos cinemáticos [2]

Para el análisis realizado a continuación, se consideran exclusivamente robots móviles con ruedas. Asimismo, se adoptan tas siguientes hipótesis simplificadoras:

a) El robot se mueve sobre una superficie plana.
b) Los ejes de guiado son perpendiculares al suelo.
c) Se supone que las ruedas se mueven con rodadura pura; es decir, el deslizamiento es despreciable en el periodo de control.
d) El robot no tiene partes flexibles.
e) Durante un periodo de tiempo suficientemente pequeño en el que se mantiene constante la consigna de dirección, el vehículo se moverá de un punto al siguiente a lo largo de un arco de circunferencia.
f) El robot se comporta como un sólido rígido, de forma que si existen partes móviles (ruedas de dirección), éstas se situarán en la posición adecuada mediante el sistema de control.

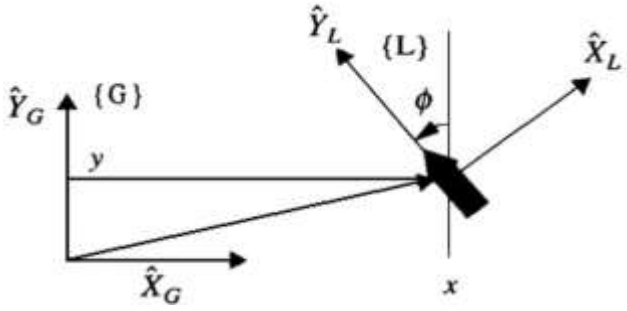
[2] Ollero, A. Robótica: Manipuladores y robots móviles.

En la práctica, existen diferentes tipos de ruedas cuya consideración tiene una notable influencia en el modelo cinemática de vehículo. Así, cabe distinguir entre cuatro tipos de rueda:

- Las ruedas fijas, que sólo pueden rotar sobre su eje.
- Las ruedas de direccionamiento se caracterizan por la rotación alrededor del eje vertical que pasa por el centro de la rueda y que permite orientarla con respecto al vehículo.
- Las ruedas de castor también son orientables respecto al vehículo, pero el eje vertical de rotación no pasa por el centro de la rueda.
- Las ruedas suecas permiten variar la dirección de la velocidad del punto de contacto con relación al plano de la rueda (el vector velocidad puede tener un ángulo con respecto a la tangente).

Considérese un sistema de referencia G y un sistema L con centro en el punto de guiado del vehículo y eje $\hat{Y}_L$ en la dirección del eje longitudinal del vehículo (Figura 6).

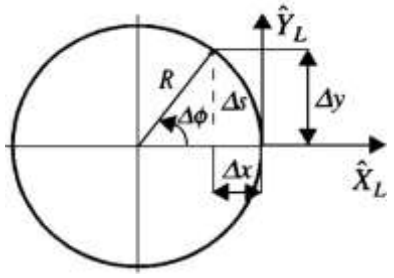
Figura 6. Cambio de sistema de referencia en navegación de robots móviles



Ollero, A. Robótica: Manipuladores y robots móviles.

Supóngase que el vehículo se desplaza en un intervalo de control según un arco de circunferencia, tal como se muestra en la Figura 7. Esta suposición es válida para intervalos de control suficientemente pequeños.

Figura 7. Círculo osculador



Ollero, A. Robótica: Manipuladores y robots móviles.

La velocidad lineal del vehículo viene dada por:

$$v = \frac{\Delta s}{\Delta t} \tag{1.3}$$

y la velocidad angular por:

$$w = \frac{\Delta \phi}{\Delta t} \tag{1.4}$$

siendo *Δs* y *ΔΦ* el espacio recorrido por el punto de guiado de vehículo y su cambio de orientación durante el intervalo de control *Δt*.

La longitud *Δs* del arco recorrido por el robot viene dada por:

$$\Delta s = R\Delta\phi \tag{1.5}$$

siendo *R* el radio de giro o radio de la circunferencia que describe el punto de guiado.

Las ecuaciones de movimiento en el sistema L en la posición inicial son:

$$\begin{aligned} {}^{L}(\Delta x) &= -(R - R\cos(\Delta\phi)) \\ {}^{L}(\Delta y) &= R\sin(\Delta\phi) \end{aligned} \tag{1.6}$$

Si la orientación inicial del vehículo con respecto al sistema G es de ϕ, el movimiento en el sistema G se determina rotando ϕ:

$$\begin{aligned} \Delta x &= R\left[\cos(\Delta\phi)-1\right]\cos(\phi) - Rsen(\Delta\phi)sen(\phi) \\ \Delta y &= R\left[\cos(\Delta\phi)-1\right]sen(\phi) + Rsen(\Delta\phi)\cos(\phi) \end{aligned} \tag{1.7}$$

Suponiendo que el intervalo de control es suficientemente pequeño, también lo será el cambio de orientación $\Delta\phi$ con lo cual se tendrá que

$$\begin{aligned} \cos(\Delta\phi) &\cong 1 \\ \sin(\Delta\phi) &\cong \Delta\phi \end{aligned} \tag{1.8}$$

Sustituyendo en la ecuación (1.7) se tiene que:

$$\begin{aligned} \Delta x &= -R\Delta\phi\sin(\phi) \\ \Delta y &= Rsen\Delta\phi\cos(\phi) \end{aligned} \tag{1.9}$$

y teniendo en cuenta la ecuación (1.5)

$$\begin{aligned} \Delta x &= -\Delta s\sin(\phi) \\ \Delta y &= \Delta s\cos(\phi) \end{aligned} \tag{1.10}$$

dividiendo ambas ecuaciones por *Δt*, teniendo en cuenta la ecuación (1.3) y haciendo tender *Δt* a cero, se llega a:

$$\begin{aligned} x' &= -v\sin(\phi) \\ y' &= v\cos(\phi) \end{aligned} \tag{1.11}$$

ecuaciones a las que puede añadirse la que se obtiene a partir de la (1.4):

$$\phi' = w \tag{1.12}$$

la cual proporciona la variación de la orientación.

1.2 SENSORES

Los sensores son dispositivos de entrada que permiten al robot tener conocimiento de su estado (sensores internos) y del estado de su entorno (sensores externos). Esta información la evalúa el controlador para generar una acción de control sobre los actuadores, cada dispositivo cambia dependiendo del uso o aplicación que tenga el robot. En la Tabla 1 se resumen los sensores comúnmente empleados para obtener información de presencia, posición y velocidad en robots industriales.

Tabla 1. Sensores comúnmente empleados en robótica

Presencia	Inductivo	
	Capacitivo	
	Efecto Hall	
	Célula Reed	
	Optico	
	Contacto	
	Ultrasonido	
Posición	Analógicos	Potenciómetro
		Resolver
		Sincro
		LVDT
	Digitales	Encoder absoluto
		Encoder incremental
		Regla óptica
Velocidad	Tacogenerador	

Ollero, A. Robótica: Manipuladores y robots móviles.

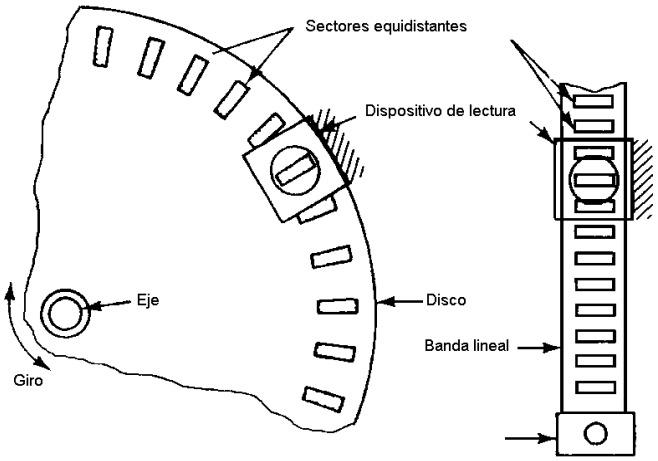
1.2.1 Sensores de posición

Para el control de posición se emplean fundamentalmente los denominados encoders. Los potenciómetros dan bajas prestaciones por lo que no se emplean salvo en contadas ocasiones (robots educacionales, ejes de poca importancia).

Los encoders o codificadores ópticos incrementales, convierten un desplazamiento rotacional en una señal digital sin necesidad de un convertidor análogo-digital. La medida del desplazamiento se realiza contando las interrupciones de un haz de luz que es interferido por un disco perforado.

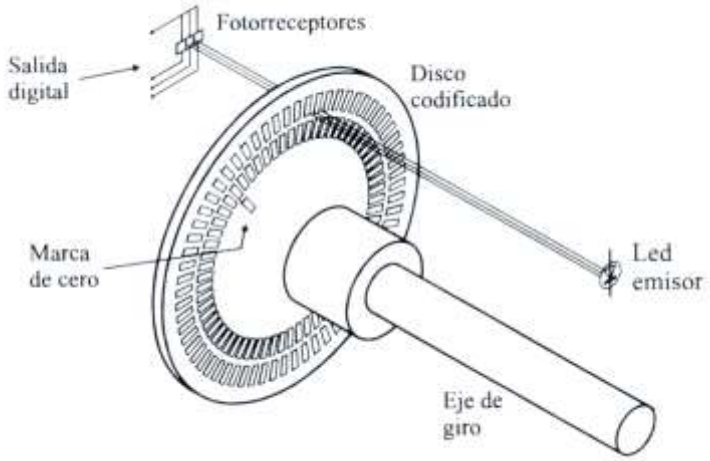
Los encoder constan, en su forma más simple, de un disco transparente con una serie de marcas opacas colocadas radialmente y equidistantes entre sí; de un haz de luz que apunta perpendicular al disco, y de un elemento fotorreceptor. El eje cuya posición se quiere medir va acoplado al disco transparente. Con esta disposición, a medida que el eje gire se van generando pulsos en el receptor cada vez que la luz atraviese cada marca, y llevando una cuenta de estos pulsos es posible conocer la posición del eje. Véase la Figura 8 y la Figura 9.

Figura 8. Codificadores ópticos rotacionales y lineales



Ollero, A. Robótica: Manipuladores y robots móviles.

Figura 9. Disposición de un encoder rotacional



Barrientos, A. Peñín, L. Balaguer C. Aracil, R. Fundamentos de robótica.

1.2.2 Sensores de proximidad ultrasónicos

Estos sensores funcionan mediante el principio del tiempo de vuelo, emitiendo pulsos de sonido y determinando el tiempo hasta que se detecta una vez que ha sido reflejado por el objeto. De esta forma, teniendo en cuenta la velocidad de propagación del sonido, puede llegar a determinarse una distancia, tal como se efectúa en los sensores láser de medida de distancias. Esto se logra, usando la relación:

$$X = V \times \frac{t}{2} \qquad (1.13)$$

Donde:

X = Distancia del obstáculo (metros)
V = Velocidad del sonido en el aire ≈ 340 m/s
t = Tiempo transcurrido entre la activación del emisor y el receptor (segundos)

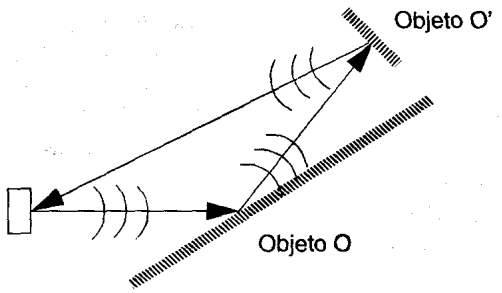
Nótese que la magnitud del tiempo se divide en dos, debido a que el sonido debe ir y volver del obstáculo.

También existen sensores de proximidad ultrasónicos que indican si existe o no un objeto a una distancia menor que un umbral establecido, el cual puede programarse por medio de la ganancia del amplificador del sensor.

Conviene poner de manifiesto que las características de la superficie que refleja la onda y el ángulo de incidencia tienen una notable influencia en la eficiencia de estos sensores.

En efecto, si el ángulo de incidencia excede un cierto valor crítico, la energía reflejada no entrará en la zona de detección. Pueden recibirse también reflexiones desperdigadas de otros objetos generando señales falsas tal como se pone de manifiesto en la Figura 10. Nótese como, en este caso, pudiera no detectarse que el objeto O está muy próximo debido a que se recibe el rebote de O'.

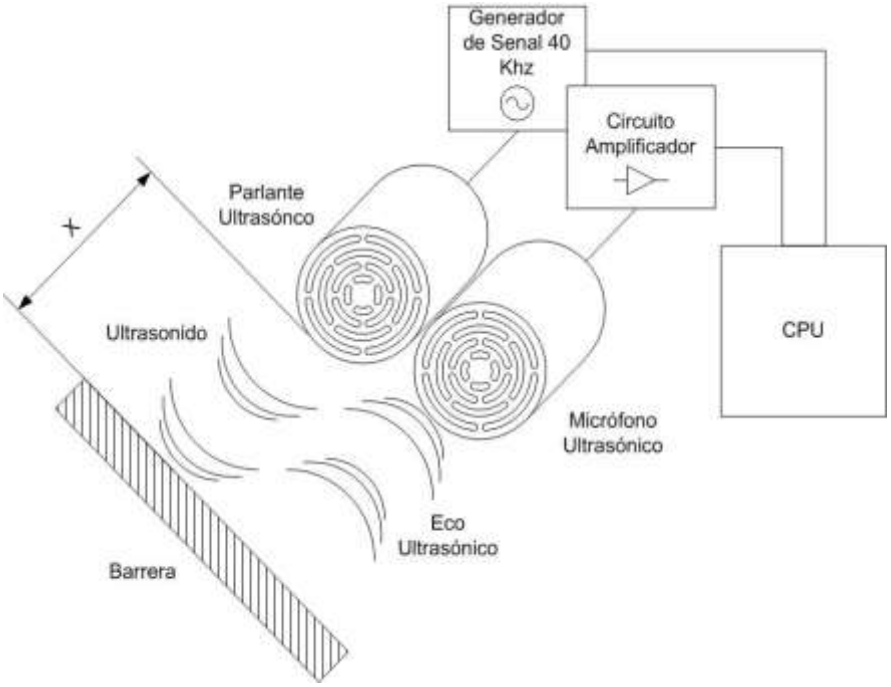
Figura 10. Problemas de rebote en sensores de ultrasonido



Giamarchi, Frédéric. Robots móviles.

La Figura 11, ilustra un sistema básico de detección de proximidad basado en sensores ultrasónicos. Sus elementos fundamentales son el parlante emisor ultrasónico, un micrófono de la misma naturaleza, un sistema generador de señales para excitar el emisor (normalmente se trabaja a una frecuencia a 40 Khz.), un circuito amplificador para la señal obtenida del receptor, y un sistema digital de procesamiento que se encarga de hacer los cálculos de distancia en función del tiempo de vuelo.

Figura 11. Diagrama general para el funcionamiento de sensores ultrasónicos



Gonzalez, H. Jurado, J. Diseño y construcción de un robot móvil guiado por señales moduladas.

Cabe señalar que los sistemas acondicionadores de señal para este tipo de sensores, no requieren de filtros pasabanda puesto que los fabricantes diseñan los dispositivos para que reaccionen únicamente ante la frecuencia nominal de detección.

1.3 ACTUADORES

Los actuadores generan las fuerzas o pares necesarios para animar la estructura mecánica. Los actuadores utilizados en robótica pueden emplear energía neumática, hidráulica o eléctrica. En la actualidad se ha extendido el uso de motores eléctricos, y en especial de motores de corriente continua, empleándose

en algunos casos motores paso a paso y otros actuadores electromecánicos sin escobillas.

Cada uno de estos sistemas presenta características diferentes, siendo preciso evaluarlas a la hora de seleccionar el tipo de actuador más conveniente. Las características a considerar son entre otras:

- Potencia
- Controlabilidad
- Peso y volumen
- Precisión
- Velocidad
- Mantenimiento

Las características de control, sencillez y precisión de los accionamientos eléctricos hacen de estos los más usados en los robots. Para este proyecto, se usaron motores de corriente continua y motores paso a paso, debido a que todos los movimientos requeridos son de naturaleza angular. A continuación se describen este tipo de actuadores.

1.3.1 Motores de corriente continua

Son los más usados en la actualidad por su facilidad de control. En este proyecto el motor incluye un codificador de posición (encoder) para poder realizar su control.

Los motores DC están constituidos por dos devanados internos, inductor e inducido, que se alimentan con corriente continua:

- El inductor, también denominado devanado de excitación, está situado en el estator y crea un campo magnético de dirección fija, denominado de excitación.
- El inducido, situado en el rotor, hace girar al mismo debido a la fuerza de Lorentz que aparece como combinación de la corriente circulante por él y del campo magnético de excitación. Recibe la corriente del exterior a través del colector de delgas, en el que se apoyan unas escobillas de grafito.

Para que se pueda realizar la conversión de energía eléctrica en energía mecánica de forma continua es necesario que los campos magnéticos del estator y del rotor permanezcan estáticos entre sí. Esta transformación es máxima cuando ambos campos se hallan en cuadratura. Según el control de velocidad se tienen dos tipos de motores DC:

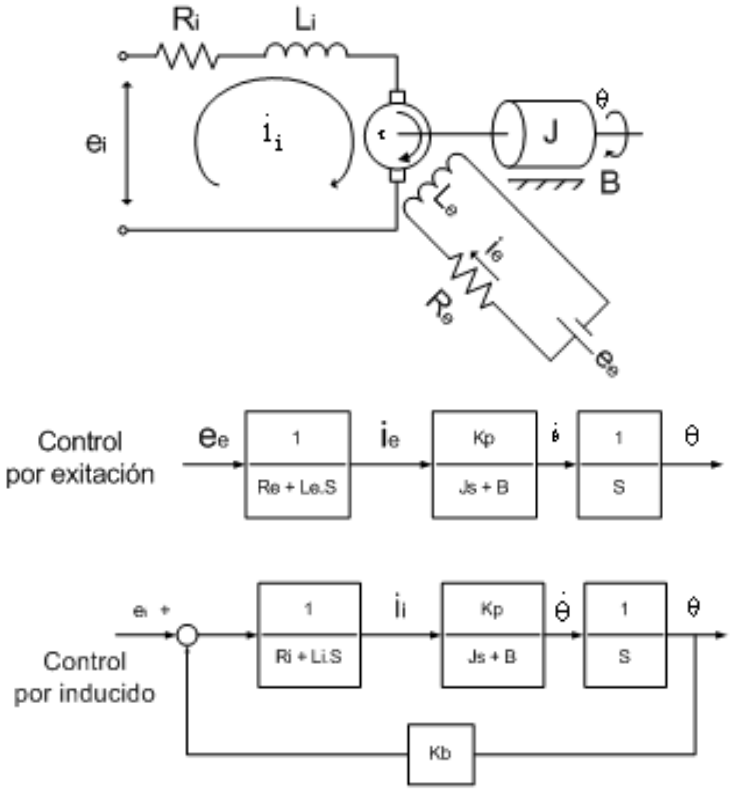
- Controlado por inducido: La intensidad del inductor se mantiene constante, mientras que la tensión del inducido se utiliza para controlar la velocidad de giro.
- Controlado por excitación: Se actúa al contrario que en la excitación por inducido.

Del estudio de ambos motores y realizando las simplificaciones correspondientes, se obtiene que la relación entre tensión de control y velocidad de giro (función de transferencia), responde a un sistema de primer orden en los controlados por inducido, mientras que en el caso de los motores controlados por excitación, esta relación es la de un segundo orden. Véase la Figura 12.

Además, en los motores controlados por inducido se produce un efecto estabilizador de la velocidad de giro por la realimentación intrínseca que posee a

través de la fuerza contraelectromotriz. Por estos motivos, el motor controlado por inducido es el que se usa en el accionamiento de robots.

Figura 12. Motor DC. Esquema y funciones de transferencia.



Barrientos, A. Peñín, L. Balaguer C. Aracil, R. Fundamentos de robótica.

1.3.2 Motores paso a paso

Los motores paso a paso están concebidos de tal manera que giran un determinado ángulo proporcional a la "codificación o secuenciación" de tensiones aplicadas a sus entradas (bobinas). Además son motores de precisión que permiten inclusive sostener cargas pequeñas de baja inercia.

La posibilidad de controlar en todo momento esta codificación permite realizar desplazamientos angulares lo suficientemente precisos, dependiendo del ángulo de paso (o resolución angular) del motor. Véase la Tabla 2.

De la misma manera que se puede posicionar el eje del motor, es posible controlar la velocidad del mismo, la cual será función directa de la frecuencia de variación de la secuencia en las entradas.

De lo anterior se deduce que el motor paso a paso presenta una precisión y repetitividad que lo habilita para trabajar en sistemas de control abiertos sin retroalimentación.

A continuación las características más relevantes del motor paso a paso:

- Angulo de paso: Avance angular que se produce en el motor por cada pulso de excitación. Se mide en grados.

Tabla 2. Resolución de motores paso a paso

Grados por pulso de excitación	Nº pasos por vuelta de motor
0,72º	500
1,8º	200
3,75º	96
7,5º	48
15º	24

Autor.

- Par dinámico de trabajo: Momento máximo que el motor es capaz de desarrollar sin perder paso (resbalar). Esta propiedad varía inversamente con la velocidad de conmutación del circuito de potencia.

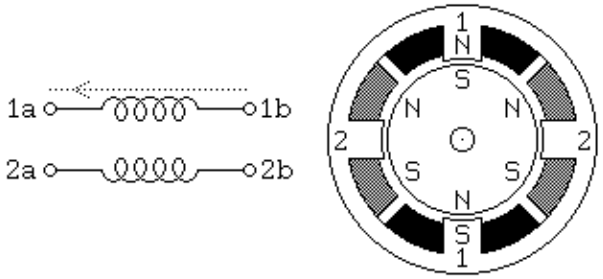
- Par de mantenimiento: Par requerido para desviar, en régimen de excitación, un paso el rotor cuando la posición anterior es estable; es mayor que el par dinámico y actúa como freno para mantener el rotor en una posición estable dada.

- Motor paso a paso bipolar

Como las bobinas se encuentran distribuidas simétricamente en torno al estator, el campo magnético creado dependerá en magnitud de la intensidad de corriente por cada fase, y en polaridad magnética, del sentido de la corriente que circule por cada bobina.

Así, el estator adquiere la magnetización correspondiente, orientándose el rotor según ella.

Figura 13. Motor paso a paso bipolar

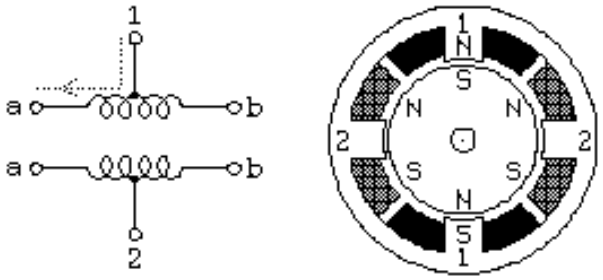


- Motor paso a paso unipolar

En cuanto a construcción son similares a los bipolares, excepto en el devanado de su estator. Cada bobina del estator se encuentra dividida en dos mediante una derivación central conectada a un terminal de alimentación.

Así, el sentido de la corriente que circula a través de la bobina y por consiguiente la polaridad magnética del estator viene determinada por el terminal al que se conecta la otra línea de la alimentación, a través de un dispositivo de conmutación. Por consiguiente las medias bobinas de conmutación hacen que se inviertan los polos magnéticos del estator, en la forma apropiada.

Figura 14. Motor paso a paso unipolar



Existen tres secuencias posibles para energizar las bobinas y con las cuales se obtiene un correcto funcionamiento del motor. Estas son:

- Paso simple

Esta secuencia de pasos es la mas simple de todas y consiste en activar cada bobina una a una y por separado, con esta secuencia de encendido de bobinas no se obtiene mucha fuerza ya que solo es una bobina cada vez la que arrastra y sujeta el rotor del eje del motor, sin embargo es la que requiere un menor nivel de análisis, además de que consume menos corriente por lo que el motor será menos propenso a sobrecalentarse.

- Paso doble

Con el paso doble activamos las bobinas de dos en dos con lo que se genera un campo magnético más potente que atraerá con más fuerza y retendrá el rotor del motor durante el frenado. Los pasos también serán algo mas bruscos debido a que la acción del campo magnético es mas poderosa que en la secuencia anterior. No obstante, el exceso de potencia entregada puede inducir recalentamiento en el motor, por lo que se recomienda aplicar este tipo de secuencia sólo si se requiere de mucha fuerza, pero durante períodos de tiempos no muy prolongados.

- Medio paso

Combinando los dos tipos de secuencias anteriores podemos hacer mover el motor en pasos más pequeños (la mitad de paso normal) y precisos y así pues tenemos el doble de pasos de movimiento para el recorrido total de 360º del motor.

Este tipo de secuencia debe aplicarse en casos donde se requiera de mayor precisión. Ahora bien, debe tenerse en cuenta que los pasos 2, 4, 6 y 8 serán los más bruscos, debido a que el campo magnético es más fuerte. Por tal razón, el motor puede vibrar un poco mas de lo esperado.

1.4 COMPONENTES ELECTRÓNICOS

En esta sección se describen las características utilizadas de cada uno de estos componentes, para mas información se puede remitir al datasheet respectivo del componente, que lo podrá encontrar en el anexo A1.

1.4.1 Driver analógico para motor TA7291

El TA7291 es un driver para control analógico de motores DC de imán permanente. Véase en la Tabla 3 la descripción de cada uno de los pines.

Incorpora una entrada analógica de voltaje de referencia (Vref), por medio de la cual es posible controlar o referenciar la potencia entregada al motor.

Por esta razón, es ideal para aplicaciones de control automático de velocidad y es muy usado en radio caseteras y equipos electrodomésticos en general. En este caso para robótica móvil.

Tabla 3. Descripción de pines del TA7291

Nº pin	Símbolo	Tipo	Descripción
2	Vcc	-	Voltaje alimentación etapa lógica
6	Vs	-	Voltaje alimentación para motor
8	Vref	Entrada	Voltaje referencia para control motor
5	GND	-	Pin tierra
9	IN 1	Entada	Entrada lógica 1
4	IN 2	Entrada	Entrada lógica 2
7	OUT 1	Salida	Salida 1 (al motor)
3	OUT 2	Salida	Salida 2 (al motor)

Datasheet TA7291.

La Tabla 4, ilustra las posibles combinaciones de entradas digitales con sus respectivas salidas. Este circuito integrado incorpora un estado de freno (STOP) que evita los picos de corriente que normalmente se presentan al momento de cambiar el sentido del motor a plena marcha.

Tabla 4. Tabla de la verdad del TA7291

ENTRADA		SALIDA		MODO
IN 1	IN 2	OUT 1	OUT 2	
0	0	∞	∞	Apagado
0	1	L	H	Izquierda
1	0	H	L	Derecha
1	1	H	H	Freno

Datasheet TA7291.

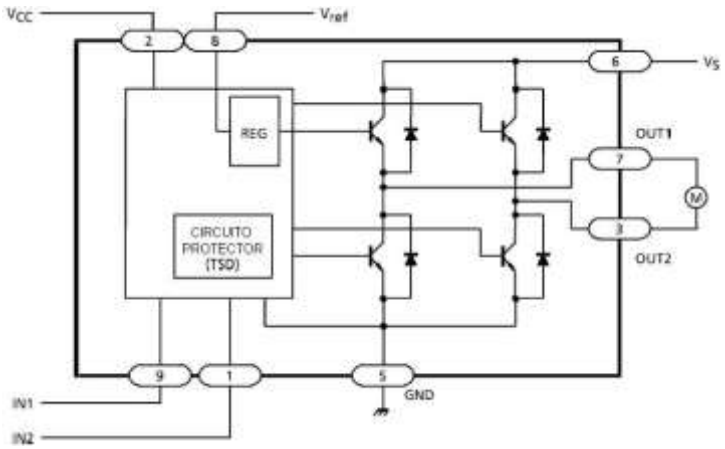
Características del driver:

- Modos permitidos: Derecha, Izquierda, freno y apagado.
- Corriente de salida hasta 1 amperio.
- Rango de voltaje de operación:

  Vcc (opr.) = 4.5 – 20 V
  Vs (opr.) = 0 – 20 V
  Vref (opr.) = 0 – 20 V

- Estructura en cierre térmico, protector contra sobrecargas.
- Disponibilidad de modo de espera (Freno).
- Histéresis en todas las entradas.

Figura 15. Diagrama de bloques del TA7291



Datasheet TA7291.

1.4.2 Driver para motor paso a paso ULN2003

El ULN2003 es un circuito comercial con 7 transistores Darlington con entradas compatibles con TTL (lógica digital de 0 y 5 voltios), cada una protegida con 2 diodos (protegen contra tensiones inversas), ideal para el accionamiento de cargas inductivas, como son las bobinas internas de los motores paso a paso.

Incorpora siete entradas digitales y siete salidas analógicas de voltaje, dos pines de alimentación (tierra y 12V en este caso) por medio de la cual es posible alimentar motores paso a paso.

Tabla 5. Descripción de pines del ULN2003

Nº pin	Símbolo	Tipo	Descripción
9	Vcc		Voltaje alimentación
8	GND		Pin tierra
1-7	IN 1-7	Entrada	Entrada lógicas 1-7
10-16	OUT 1-7	Salida	Salidas 1-7

Datasheet ULN2003.

Características del driver:

- Corriente de colector: 500mA
- Corriente de base: 25mA
- Rango de voltaje de operación:

  Vo (opr.) = 50 V
  Vin (opr.) = 30 V

1.4.3 Amplificador operacional

El amplificador operacional es un dispositivo lineal de propósito general el cual tiene capacidad de manejo de señal desde f = 0Hz hasta una frecuencia definida

por el fabricante; tiene además limites de señal que van desde el orden de los nV, hasta unas docenas de voltio (especificación también definida por el fabricante). Los amplificadores operacionales caracterizan por su entrada diferencial y una ganancia muy alta, generalmente mayor que 10^5 equivalentes a 100 dB.

El A.O. es un amplificador de alta ganancia directamente acoplado, que en general se alimenta con fuentes positivas y negativas, la cual permite que tenga excursiones tanto por arriba como por debajo tierra (o el punto de referencia que se considere).

El nombre de amplificador operacional proviene de una de las utilidades básicas de este, como lo son realizar operaciones matemáticas en computadores análogos.

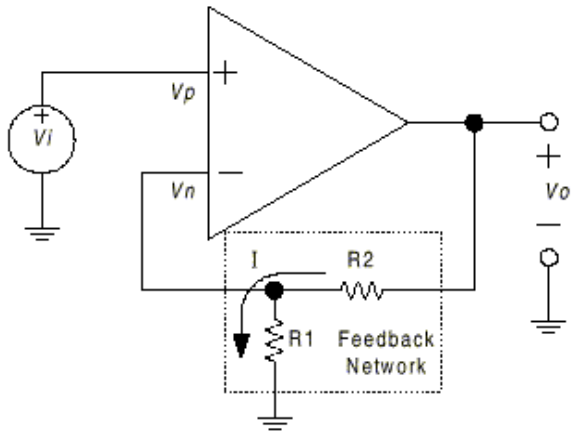
Un amplificador operacional ideal tendría ganancia infinita y ninguna corriente de entrada, y la salida v0 no sería afectada por ninguna carga. Además tendría otras propiedades, tales como un ancho de banda infinito, un intervalo de voltajes infinito a la entrada y la salida. Pero la propiedad más importante para simplificar las ideas de diseño es ganancia infinita. En la práctica esto no puede lograrse, pero el análisis basado en un modelo con ganancia infinita conduce a un acuerdo excelente en cuanto a la actuación real en la mayoría de los casos.

- El amplificador no inversor

La configuración del amplificador no inversor, se muestra en la Figura 16.

En este circuito, la tensión *Vi* se aplica a la entrada (+), y una fracción de la señal de salida, *Vo*, se aplica a la entrada (-) a través del divisor de tensión *R1* - *R2*. Puesto que no fluye corriente en ningún terminal de entrada, y ya que *Vd* = 0, la tensión en *R1* será igual a *Vi*.

Figura 16. Amplificador no inversor



Amplificadores operacionales y filtros activos.

Así pues

$$V_i = I \cdot R_1 \tag{1.14}$$

y como

$$V_o = I \cdot (R_1 + R_2) \tag{1.15}$$

tendremos pues que:

$$V_o = \frac{V_i}{R_1} \cdot (R_1 + R_2) \tag{1.16}$$

que si lo expresamos en términos de ganancia:

$$\frac{V_o}{V_i} = \frac{(R_1 + R_2)}{R_1} \tag{1.17}$$

que es la ecuación característica de ganancia para el amplificador no inversor ideal.

También se pueden deducir propiedades adicionales para esta configuración. El límite inferior de ganancia se produce cuando *R2* = 0, lo que da lugar a una ganancia unidad.

En el amplificador inversor, la corriente a través de *R1* siempre determina la corriente a través de *R2*, independientemente del valor de *R2*, esto también es cierto en el amplificador no inversor. Luego *R2* puede utilizarse como un control de ganancia lineal, capaz de incrementar la ganancia desde el mínimo unidad.

1.5 FPGA: TARJETA DE DESARROLLO

El dispositivo lógico programable seleccionado para controlar este prototipo es un FPGA de la marca Xilinx, debido a la forma en que se realiza su montaje no se tiene el dispositivo o circuito por si solo, al contrario, se implementa una tarjeta o board de desarrollo en la cual viene montado el FPGA y en la cual este está interconectado con una serie de dispositivos e interfaces, algunos de estos son utilizados en el funcionamiento del robot. La tarjeta mencionada hace parte el laboratorio de electrónica y fue adquirida a finales del año 2006, la selección de la misma estuvo a cargo de los autores y la directora del presente proyecto, y se tuvo como criterio de selección la utilidad de la misma en este proyecto y en futuras investigaciones o desarrollos por parte de la universidad. A continuación se hace una breve introducción al FPGA y se enumeran las interfaces y dispositivos de la tarjeta utilizados en este proyecto.

1.5.1 FPGA

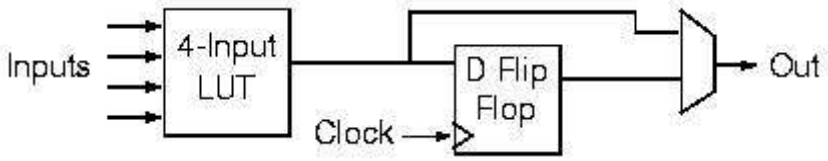
Una “Matriz de puertas programable por campo” o FPGA (field programmable gate array) por sus siglas en inglés es un dispositivo semiconductor que contiene

componentes lógicos programables e interconexiones programables entre ellos, estos componentes pueden ser programados para duplicar la funcionalidad de compuertas lógicas básicas o funciones combinacionales más complejas tales como decodificadores o funciones matemáticas simples. En muchos FPGA, estos componentes lógicos programables (o bloques lógicos, según el lenguaje comúnmente usado) también incluyen elementos de memoria, los cuales pueden ser simples flip-flops o bloques de memoria más complejos.

Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema. Estos bloques lógicos e interconexiones pueden ser programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

La arquitectura básica consiste en un arreglo de bloques lógicos programables (CLB, del inglés *Configurable Logic Block*) y canales de comunicación. La mayoría de los canales de comunicación tienen el mismo ancho (número de cables). Un bloque lógico típico consiste en una tabla de funciones lógicas de cuatro entradas (*LookUp Table*) y un flip-flop como se muestra en la Figura 17.

Figura 17. Bloque lógico programable



http://es.wikipedia.org/wiki/FPGA

También existen bloques usados como interfaz entre el FPGA y otros dispositivos. Estos son llamados IOB (Bloques de Entrada/Salida). Los IOBs son los que

definen si un pin del FPGA será usado como entrada o como salida en el sistema implementado.

La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos, para esto cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de descripción de hardware o HDL (*Hardware Description Language*). Para este proyecto el lenguaje utilizado es el VHDL que se explicará en la sección 1.6.

Tanto los CPLD como los FPGA incluyen un número relativamente grande de elementos lógicos programables. El rango de densidad de los CPLD va desde miles a decenas de miles de compuertas lógicas, mientras que el de los FPGA va típicamente desde decenas de miles hasta varios millones.

La principal diferencia entre los CPLD y los FPGA son sus arquitecturas. Un CPLD tiene una estructura un poco más restringida, consistiendo en la unión de uno o más arreglos lógicos que alimentan a un número pequeño de registros con entrada de reloj. Ello conlleva una flexibilidad reducida, con la ventaja de una mejor predicción de los tiempos de retraso. La arquitectura de los FPGA, por otro lado, cuenta con muchas más interconexiones. Esto los hace más flexibles, es decir, el rango de diseños prácticos en los cuales pueden ser usados es mayor.

Otra notable diferencia entre CPLD y FPGA es la presencia de funciones de más alto nivel (tales como sumadores y multiplicadores) dentro de los FPGA, además de memorias.

1.5.2 Tarjeta de desarrollo

La tarjeta de desarrollo a implementar es la “Spartan-3E Starter Kit” referencia HW-SPAR3E-SK-US del fabricante Xilinx a continuación se explica cada uno de los dispositivos e interfaces de la tarjeta utilizados en este proyecto, para información mas avanzada de estos o de cualquier otro dispositivo de la misma tarjeta, referirse a la guía de usuario que se encuentra en el CD.

- FPGA Xilinx Spartan 3E

El FPGA como ya se dijo es del fabricante Xilinx y pertenece a la familia Spartan 3E y su referencia es XC3S500E y entre sus características más relevantes se tiene: 500.000 compuertas lógicas, 1.164 CLB, 232 pines I/O, frecuencia de funcionamiento desde 5MHz hasta 300MHz (en el proyecto, su funcionamiento es a 50MHz del cristal interno de la tarjeta), 3.3V y 2.5V como voltaje de funcionamiento. Finalmente, hay que decir que tiene soporte completo con el software de desarrollo (programación y síntesis) ISE™, que viene con la tarjeta.

El FPGA no memoriza por si solo la manera de su funcionamiento, por el contrario, cada vez que es encendido, el carga el programa de alguna memoria, dentro de las posibilidades existentes la seleccionada es la memoria Flash PROM de la cual dispone la tarjeta, el programa de funcionamiento del robot se encuentra guardado en esta memoria y la tarjeta se deberá configurar para que al encendido el FPGA cargue el programa de dicha memoria.

- Interruptores, botones y leds

La tarjeta cuenta con 4 interruptores, 5 botones o pulsadores y 8 leds; con los anteriores componentes se realiza la configuración del robot y se indican los parámetros y variables de funcionamiento por parte del usuario.

Internamente, la tarjeta tiene resistencias de protección (pull up y pull down) que se activan por programación en el caso de los interruptores y los botones. Para los leds, estos poseen resistencias en serie para su correcto funcionamiento. Los valores de estás resistencias están definidos por el fabricante de la tarjeta.

- Pantalla LCD de caracteres

La tarjeta de desarrollo posee una pantalla LCD de 2 líneas, cada una de estas con la posibilidad de mostrar hasta 16 caracteres normales y ASCII estándar. Esta pantalla puede funcionar con una interfaz de datos de 8 bits o de 4 bits, en la tarjeta de desarrollo viene implementada con la de 4 bits, que están conectados al FPGA y su uso se explica por el ahorro de pines I/O del mismo.

En el funcionamiento del robot la pantalla se usa para visualizar información relacionada con el proceso que se encuentra realizando el mismo.

- Conversor digital-análogo

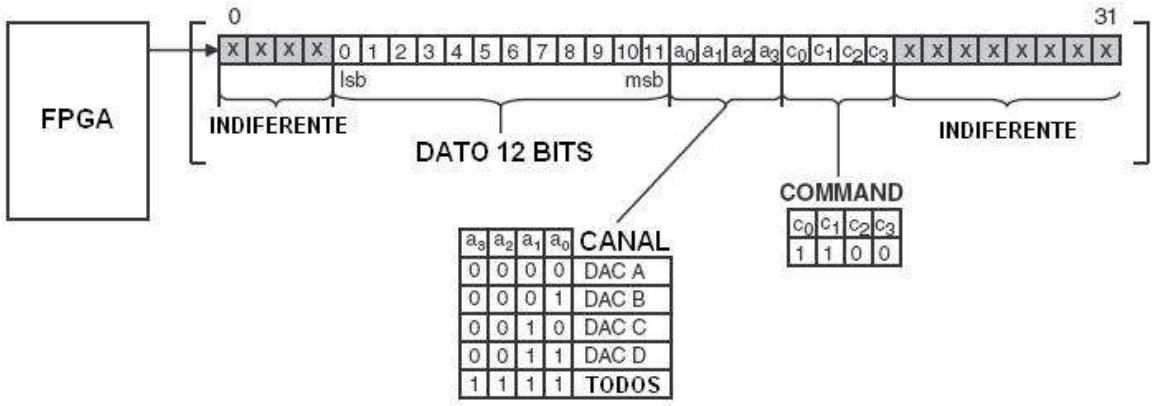
El conversor digital-análogo o DAC (*Digital to Analog Converter*) por sus siglas en inglés que está dispuesto en la tarjeta de desarrollo es de 4 canales y posee una resolución de 12 bits (*D[11:0]*) correspondiente a 4096 en base decimal, dos de los canales funcionan con un voltaje de referencia (*Vref*) de 3.3V y los otros dos con 2.5V. El voltaje de salida está relacionado con el valor digital de 12 bits de acuerdo a la siguiente relación.

$$Vo = \frac{D[11:0]}{4096} xVref \qquad (1.18)$$

La comunicación entre el conversor y el FPGA se realiza a través de una interfaz periférica serial (SPI, *Serial Peripheral Interface*) de 32 bits, esto quiere decir que

el FPGA envía un dato de 32 bits por medio del bus SPI al conversor, estos 32 bits incluyen los 12 del dato binario a convertir, 4 más que definen el canal, otros 4 que indican un código interno y los demás bits son indiferentes, la configuración del dato se puede ver en la siguiente figura.

Figura 18. Dato serial que envía FPGA a conversor digital-análogo



Autor.

El uso del conversor se aplica a la modificación de la velocidad de los motores que desplazan el robot.

- Conectores de expansión

Estos conectores son el medio o la interfaz por medio de la cual se pueden conectar periféricos o dispositivos externos al FPGA, en este caso se hace referencia a los tres conectores de 6 pines que posee la tarjeta de desarrollo, en estos conectores un pin tiene es conexión a tierra, otro a Vcc que para este proyecto se configura este voltaje como 3.3V, y los restantes cuatro pines están conectados directamente a pines I/O del FPGA.

Estos conectores se utilizan cono interfaz para los sensores y actuadores propios del robot.

1.6 VHDL

VHDL, que significa VHSIC (*Very High Speed Integrated Circuit*) *Hardware Description Language*, lo que en español sería, lenguaje de descripción hardware de circuitos integrados de muy alta velocidad, es un lenguaje orientado a la descripción o modelado de sistemas digitales; es decir, se trata de un lenguaje mediante el cual se puede describir, analizar y evaluar el comportamiento de sistemas hardware digitales, placas de circuitos y componentes.

Uno de los objetivos del lenguaje es el modelado, desarrollo de un modelo para simulación de un circuito o sistema previamente implementado, cuyo comportamiento se conoce. El objetivo del modelado es la simulación.

Otro de los usos del lenguaje es la síntesis de circuitos. En este proceso se parte de una entrada con un determinado nivel de abstracción y se llega a una implementación más detallada, menos abstracta. La síntesis por medio de VHDL constituye hoy en día una de las principales aplicaciones del lenguaje.

Algunas ventajas de VHDL son:

- Permite diseñar, modelar y comprobar un sistema desde un alto nivel de abstracción bajando hasta el nivel de definición estructural de puertas.
- Los módulos creados en VHDL pueden utilizarse en diferentes diseños, lo que permite la reutilización de código, además, la misma descripción puede emplearse para diferentes tecnologías sin tener que rediseñar todo el circuito.
- Al estar basado en un estándar (IEEE Std 1076-1987, IEEE Std 1076-1993) los ingenieros de toda la industria de diseño pueden usar este lenguaje para minimizar errores de comunicación y problemas de compatibilidad.

- Modularidad: VHDL permite dividir o descomponer un diseño hardware y su descripción en unidades más pequeños.

1.6.1 Unidades de diseño

La estructura general de un programa en VHDL está formada por módulos o unidades de diseño, donde cada uno de ellos está compuesto por un conjunto de declaraciones e instrucciones que definen, describen, estructuran, analizan y evalúan el comportamiento de un sistema digital.

Existen cinco tipos de unidades de diseño en VHDL: entidad (*entity*), arquitectura (*architecture*), configuración (*configuration*), paquete (*package*) y cuerpo de paquete (*package body*). En el desarrollo de programas en VHDL pueden utilizarse o no tres de los cinco módulos, pero dos de ellos (entidad y arquitectura) son indispensables en la estructuración de un programa.

Las declaraciones de entidad, paquete y configuración se consideran unidades de diseño primarias, mientras que la arquitectura y el cuerpo del paquete son unidades de diseño secundarias porque dependen de una entidad primaria que se debe analizar antes que ellas.

- Entidad (Entity)

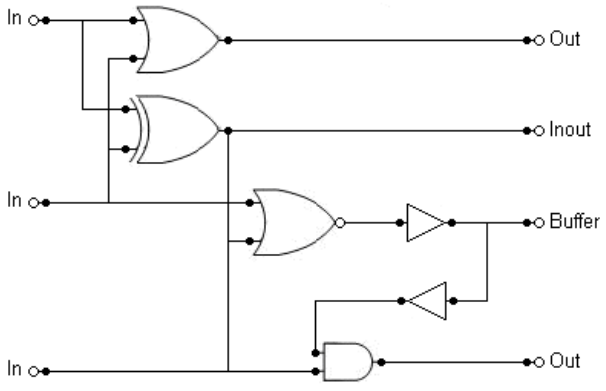
Una entidad es el bloque elemental de diseño en VHDL, las entidades son todos los elementos electrónicos (sumadores, contadores, compuertas, flip-flops, memorias, multiplexores, etc.) que forman de manera individual o en conjunto un sistema digital. La entidad representa un circuito completo a nivel de compuertas, ahora bien, esta se puede indicar a nivel de sistema indicando tan solo las entradas y las salidas del circuito.

Cada una de las señales de entrada y salida en una entidad son referidas como puerto, el cual es similar a un pin de un símbolo esquemático. Todos los puertos que son declarados deben tener un **nombre**, un **modo** y un **tipo de dato**. El nombre es utilizado para llamar al puerto.

El modo permite definir la dirección que tomará la información y puede tener uno de cuatro valores: in (entrada), out (salida), inout (entrada/salida) y buffer. En la Figura 19 se puede observar un ejemplo de los diferentes modos y el curso de las señales.

- Modo in. Indica las señales de entrada de la entidad, solo es unidireccional, es decir, nada más permite el flujo de datos hacia dentro de la entidad.
- Modo out. Indica las señales de salida de la entidad y es igualmente unidireccional.
- Modo inout. Permite declarar un puerto de forma bidireccional, además permite la retroalimentación de señales dentro o fuera de la entidad.
- Modo buffer. Permite hacer retroalimentaciones internas dentro de la entidad, pero a diferencia del modo inout, el puerto es declarado como terminal de salida.

Figura 19. Modos y curso de las señales en una entidad



Maxinez, D. Alcalá, J. VHDL El arte de programar sistemas digitales.

El tipo de dato define que clase de información se transmitirá por el puerto y se asigna de acuerdo con las características del diseño en particular. Algunos de los tipos mas utilizados en VHDL son:

- Bit. El cual tiene valor de 0 y 1 lógico.
- Bolean. Valor de verdadero o falso.
- Bit_vector. Que representa un vector o conjunto de bits para cada puerto.
- Integer. Que representa un número entero.

- Arquitectura (*architecture*)

Una arquitectura se define como la estructura que describe el funcionamiento de una entidad, de tal forma que permita el desarrollo de los procedimientos que se llevarán a cabo con el fin de que la entidad cumpla las condiciones de funcionamiento deseadas.

La gran ventaja que presenta VHDL para definir una arquitectura radica en la manera en que pueden describirse los diseños, es decir, mediante el algoritmo de programación empleado se puede describir desde el nivel de compuertas hasta sistemas complejos.

1.6.2 Descripción en VHDL

Existen dos formas de describir un circuito. La primera es por estructura, donde se indican los diferentes componentes que forman un circuito y su interconexión, de esta forma se tiene especificado el circuito y se sabe como funciona. Esta es la forma habitual en que se describen circuitos, siendo las herramientas utilizadas para ello las de captura de esquemas y las de descripción Netlist (Lista de piezas y sus puntos de conexión a cada red de un circuito).

La segunda forma es por comportamiento, que consiste en describir un circuito indicando lo que hace o como funciona. Naturalmente esta forma de describir un circuito es preferible para un diseñador puesto que lo que realmente le interesa es el funcionamiento del circuito más que sus componentes. Por otro lado, al encontrarse lejos de lo que realmente es un circuito se pueden presentar algunos problemas a la hora de implementarlo.

Ahora bien, VHDL presenta tres estilos de descripción de circuitos dependiendo del nivel de abstracción. El menos abstracto derivado de la primera forma de describir circuitos que es por medio de una descripción netamente estructural o como *Netlist*. Los otros dos estilos representan una descripción comportamental o funcional, y la diferencia proviene de la utilización o no de la ejecución serie.

La descripción por flujo de datos, que es más cercana a la descripción estructural, y que al igual que esta permite la paralelización de instrucciones, siendo aún funcional porque lo que hace es asignación de señales y operaciones entre estas funcionando de forma paralela pero sin definir explícitamente las conexiones entre estás o su estructura. Por último está la descripción comportamental algorítmica que se diferencia de la anterior por la ejecución serie, esta presenta bloques en los cuales internamente se tienen operaciones o asignaciones secuenciales, pero todo el bloque funciona de manera paralela, este estilo presenta una estructura parecida a los lenguajes de programación convencionales.

1.7 REDES NEURONALES ARTIFICIALES

Las redes neuronales se pueden entender como modelos de comportamiento inteligente que pueden ser construidos inspirados en el sistema nervioso de los seres vivos. Las aplicaciones de sistemas basados en redes neuronales han permitido diseñar redes con propósitos específicos como el reconocimiento de

patrones. Este esquema es novedoso e interesante dado que un computador digital supera la velocidad y precisión del cerebro en la relación de operaciones numéricas; aunque operaciones como reconocimiento de patrones, memoria asociativa y en general todas las relaciones con el comportamiento inteligente parecen imposibles de alcanzar mediante las concepciones computacionales tradicionales.

Las similitudes entre los modelos de redes neuronales y el funcionamiento del cerebro parecen sugerir la posibilidad de reproducir ciertas propiedades de la inteligencia. El desarrollo de sistemas artificiales inteligentes se puede obtener no solo con algoritmos más perfeccionados, sino con cambios en la concepción básica de las estructuras computacionales.

Existe una gran cantidad de problemas de ingeniería resueltos con diseños de redes neuronales artificiales capaces de emular algunas funciones realizadas por el sistema nervioso de los seres vivos. No obstante el éxito obtenido, no está comprobado que esta arquitectura sea la solución al problema de generación de verdadera inteligencia artificial.

Sin embargo, se ha comprobado que estas redes poseen nuevas propiedades, algunas muy similares a las que tiene el funcionamiento básico del cerebro, incluida la capacidad de aprender. Para demostrar esto, se aplica la prueba más importante basada en el conocimiento de que no basta ser como un cerebro, en cuanto a su arquitectura, sino que hay que comportarse como tal para emular sus propiedades básicas.

Las redes neuronales artificiales han permitido a los computadores emprender tareas específicas, tales como auxiliares auditivos y transductores de visión, ahora bien, los computadores de arquitectura tradicional abordan este tipo de tarea con mayor dificultad que una arquitectura diseñada con arreglos de bloques en VHDL.

1.7.1 ¿Qué es una red neuronal?

Una red neuronal es un sistema basado en una representación simplificada de la estructura y el funcionamiento del sistema nervioso. Se debe entrenar mediante ejemplos conocidos hasta que es capaz de asociar patrones de entrada con respuestas definidas sin necesidad de una programación explícita para un patrón en particular. Esta característica le permite a la red resolver problemas donde incluso se crean nuevas categorías de patrones para los cuales no es posible diseñar algoritmos de solución, con lo cual puede enfrentar con éxito casos en que no es posible definir una relación entre las causas y los efectos.

Las redes neuronales son modelos de cómputo paralelo que ofrecen una alternativa de procedimientos a problemas cuya solución por técnicas tradicionales resulta de difícil aplicación en la mayoría de los casos.

- Funcionamiento

El principio de funcionamiento de las redes neuronales artificiales es, en términos generales, el de un convertidor vectorial. La información codificada en forma de vector de entrada controla la red y después de cierta cantidad de intentos de comparación entre patrones ésta produce un vector de salida, que es la mejor solución que encuentra en la conversión del vector de entrada. Así las distintas respuestas de la red dependen de su arquitectura, de sus propiedades de conectividad y de las reglas algorítmicas explícitas en el paso de información de una capa de neuronas a las siguientes. El mejoramiento de las respuestas generadas por la red depende del entrenamiento de ésta mediante el cual cambia en función de un algoritmo de aprendizaje o por retroalimentación, también depende de la forma en que pasa la información entre las neuronas de la red y entre las capas de estas.

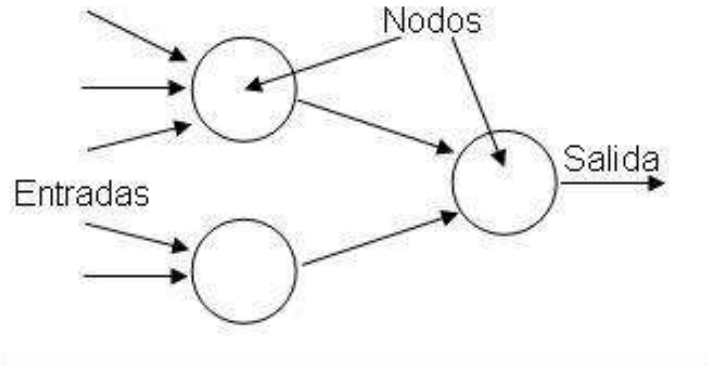
De este modo la red realiza algunas funciones elementales similares a las que realiza el cerebro en tiempos razonables aunque no en tiempo real debido a que estas redes se instalan en computadores secuenciales con algoritmos que se ejecutan paso a paso y no de manera simultánea como ocurriría en una arquitectura en paralelo, donde cada neurona de la red funciona independientemente.

- Elementos de una red neuronal

Las redes neuronales artificiales constan de elementos de procesamiento y conexión de pesos. La cantidad de estos depende de la implementación y diseño de la red. Otros elementos que intervienen dependiendo de la aplicación son las funciones preestablecidas de los vectores de entrada y salida.

En la Figura 20 se observa la estructura de una red neuronal artificial. Consta de un núcleo o nodo, entradas y salidas. La entrada es una señal de voltaje aplicado a un nodo, la cual puede ser una salida. Esta última es la respuesta e la neurona artificial. El núcleo suma las señales de entrada, las procesa y da una respuesta.

Figura 20. Red neuronal artificial



Maxinez, D. Alcalá, J. VHDL El arte de programar sistemas digitales

- Aplicaciones

Las aplicaciones de las redes neuronales y de la computación son muy abundantes. De entre ellos se destaca el procesamiento de señales o reconocimiento de patrones, la extracción de características, la inspección industrial, el pronóstico de negocios, la clasificación de crédito, la selección de seguridad, el diagnóstico médico, el procesamiento de voz, la comprensión del lenguaje natural, el control de robots y la adaptación de procesos de control.

1.7.2 Modelos de redes neuronales

- Neurona hebbiana

Donald Hebb mencionó la primera regla de aprendizaje para una red neuronal artificial, la cual establece que cuando una neurona estimula a otra, la conexión entre ellas se refuerza. La Figura 21 muestra el modelo y las ecuaciones de una neurona hebbiana.

El funcionamiento de ésta es el siguiente, la neurona artificial calcula la entrada ponderada I, donde w es la matriz de pesos, x es el vector de entrada y b es una constante inicializada en 1. Si el valor de la entrada ponderada es mayor o igual a cero, la salida de la neurona es igual a +1; si es menor a cero, la salida es -1.

En el proceso de entrenamiento lo que se hace es calcular el vector de ponderaciones w que le permita a la red comportarse de acuerdo a la información de entrada. Para este modelo, el entrenamiento consiste en cambiar el vector w por cada patrón de entrenamiento, de acuerdo con la siguiente regla hebbiana:

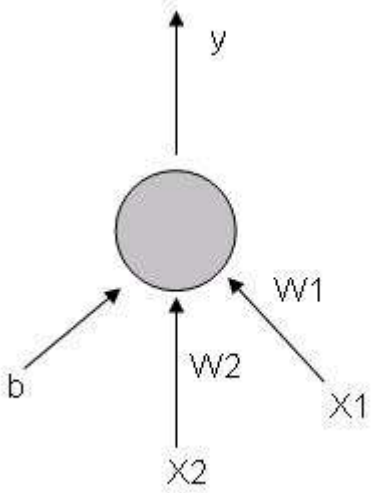
$$w_{nueva} = w_{anterior} + yx \qquad (1.19)$$

Figura 21. Modelo y ecuaciones características de la neurona hebbiana

$$I = B + \sum_{i=1}^{n} w_i x_i = b + w.x$$

$$y = \begin{cases} +1: si I > 0 \\ -1: si I \leq 0 \end{cases}$$

$$w_{nueva} = w_{anterior} + yx$$



Maxinez, D. Alcalá, J. VHDL El arte de programar sistemas digitales.

- Perceptrón

Warren S. McCulloch y Walter Pitss desarrollaron la red neuronal perceptrón en 1943 proponiendo las ecuaciones generales y el diagrama de la Figura 22.

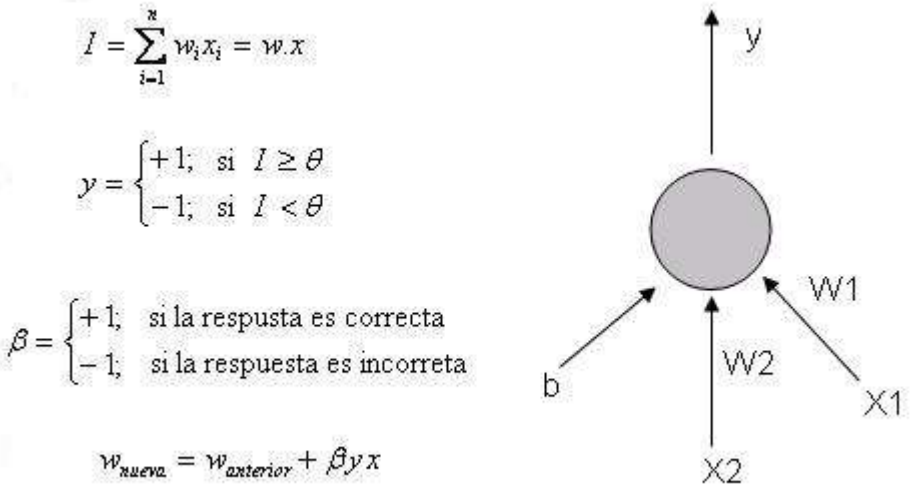
El funcionamiento del perceptrón es sencillo. La neurona suma las señales del vector de entrada x, lo que da la entrada ponderada I.

La ecuación compara la señal con un valor de umbral. Si es mayor, la salida es +1, de lo contrario es -1.

Para entrenar la red, se escogen los valores de entrada *x* que harán de patrones de entrenamiento, y para cada uno de estos se calcula si la salida del perceptrón es correcta o incorrecta, con lo que se tiene β. Finalmente, por cada iteración el vector de ponderaciones cambiará de acuerdo con la ley de Rosenblatt.

$$w_{nueva} = w_{anterior} + \beta yx \qquad (1.20)$$

Figura 22. Modelo y ecuaciones características del perceptrón



Maxinez, D. Alcalá, J. VHDL El arte de programar sistemas digitales.

- La Adelina

Este modelo utiliza la técnica del mínimo error cuadrático, desarrollado en 1960 por Bernard Widrow y Ted Hoff, de la universidad de Stanford. Ofrece la primera neurona con entrenamiento supervisado, es decir, toma en cuenta el posible error cuadrático mínimo obtenido al calcular la salida de la neurona. Su nombre proviene de *adaptive linear element* (elemento lineal adaptativo) y las ecuaciones y el modelo se muestran en la Figura 23.

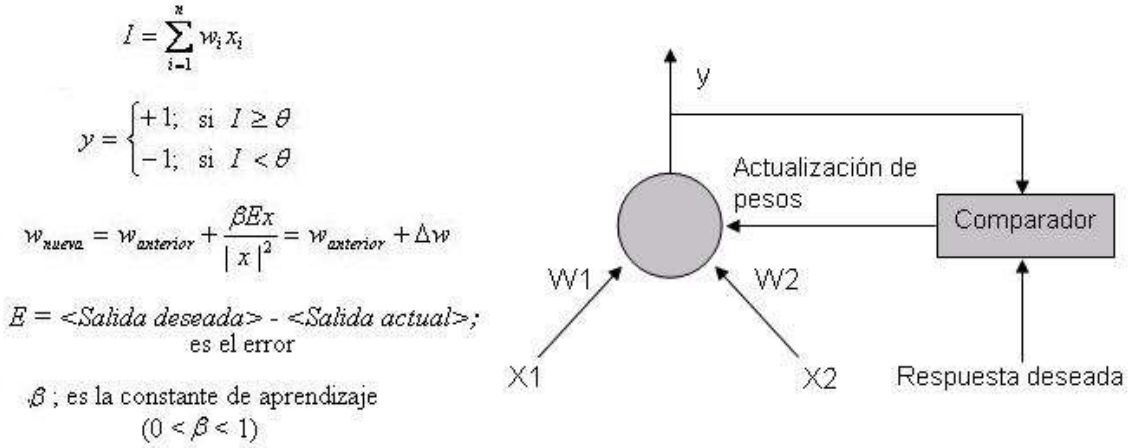
Si la entrada ponderada I es mayor que cero, entonces la salida es +1, por el contrario si es menor o igual a cero, la salida es -1. Esta salida se compara con la salida deseada calculando el error de la adaline E.

Una vez calculado el error, se puede utilizar para ajustar ponderaciones w, utilizando la regla delta.

$$w_{nueva} = w_{anterior} + \frac{\beta E x}{|x|^2} = w_{anterior} + \Delta w \qquad (1.21)$$

Donde β es la constante de aprendizaje que puede tomar un valor entre 0 y 1. E es el error calculado, x es el vector de entrada y w es el vector de ponderaciones.

Figura 23. Modelo y ecuaciones características de la adaline



Maxinez, D. Alcalá, J. VHDL El arte de programar sistemas digitales.

2 DISEÑO MECATRONICO

En términos generales, la secuencia de diseño llevo a realizar primero la selección del tipo de robot, llevando luego al diseño mecánico, pasando a la creación de los circuitos electrónicos de acople y tratamiento de señales de sensores y actuadores. En este punto se tiene el diseño físico o de hardware, se prosigue con la programación del robot para terminar creando la interfaz con el usuario.

La Figura 24 muestra el diagrama de flujo del diseño, la parte izquierda ilustra cada sección de diseño en forma de bloques; la parte derecha agrupa estos bloques en fases, que se explican en las secciones de este capítulo.

En la primera fase se describe la selección de la configuración del robot junto a las ecuaciones que rigen los movimientos del mismo.

Para la segunda fase se tiene lo concerniente al diseño mecánico: la selección de motores y materiales, y el análisis de fuerzas y energía. Con la integración de lo anterior se llego al diseño mecánico final.

La tercera fase presenta la configuración de funcionamiento en el robot de los sensores y actuadores, las conexiones entre estos y el FPGA, y la forma en que el FPGA realiza el control o maniobra de los mismos.

La programación del robot, es decir, los algoritmos que describen el funcionamiento del robot y que se implementan en el FPGA por medio del lenguaje VHDL se describen en la cuarta fase.

Finalmente la quinta fase expone la manera en que el usuario interactúa con el robot.

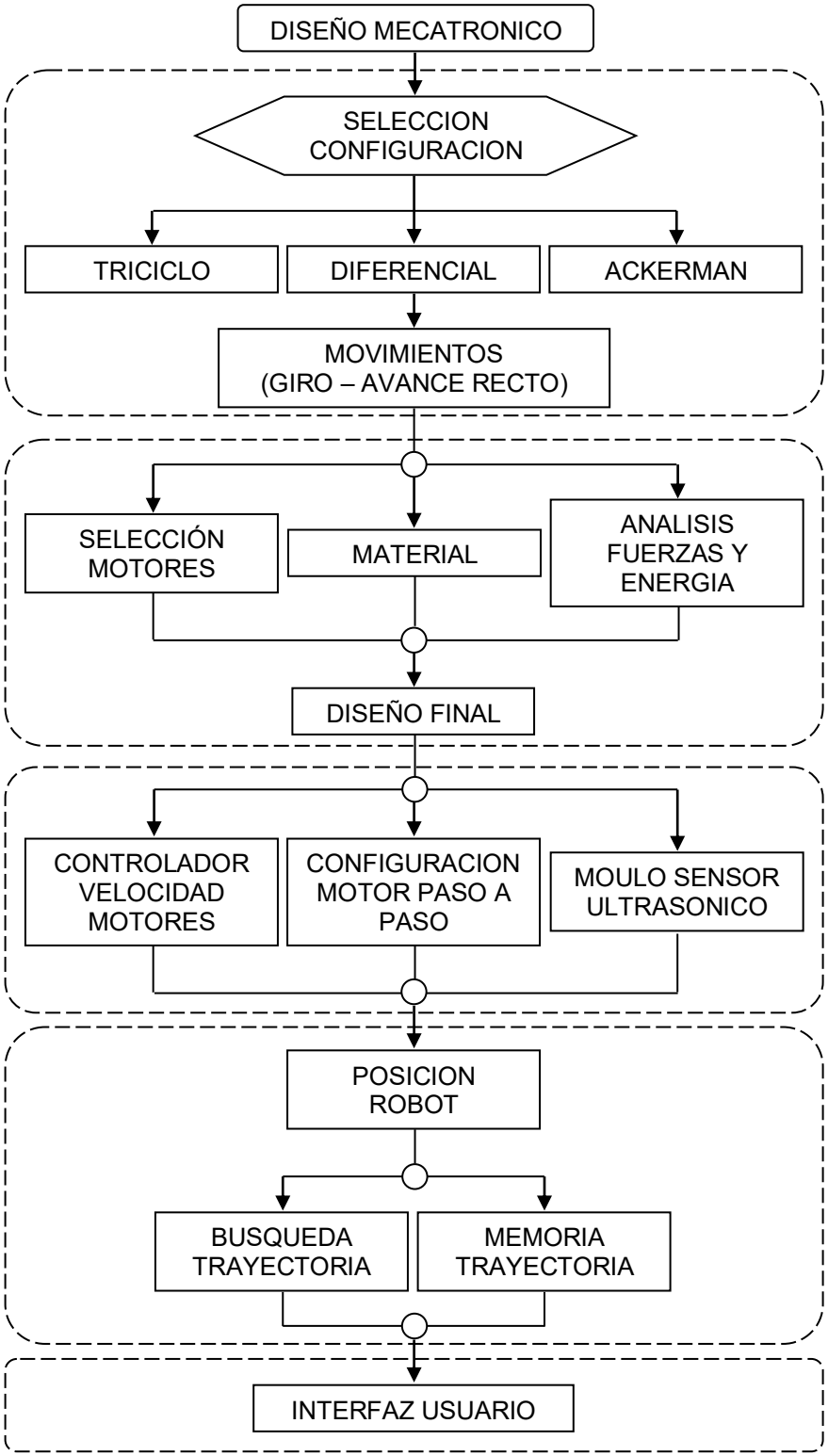


Figura 24. Diagrama de flujo del diseño mecatrónico

Autor.

2.1 SELECCIÓN Y DISEÑO DE ROBOT

2.1.1 Selección de configuración de robot

Teniendo en cuenta las necesidades mecánicas del robot a construir se debe seleccionar un tipo de movimiento el cual sea el más óptimo que supla con la tracción necesaria, la locomoción, y sea lo más eficiente de acuerdo al peso del robot.

En la Tabla 6 se observa el mecanismo de selección, asignamos una puntuación de acuerdo con la necesidad de cada característica y el factor de importancia, se realiza la multiplicación para cada característica y la sumatoria de todos los resultados, así podemos obtener puntajes para cada configuración, seleccionando finalmente el de mayor puntuación.

Tabla 6. Selección de configuración de robot

NECESIDAD	**TIPO DE CONFIGURACION**		
	TRICICLO	DIFERENCIAL	ACKERMAN
Tracción (5)	4	6	5
Espacio (4)	5	4	4
Precio (3)	5	5	5
Consumo (2)	5	5	5
TOTAL	65	71	66

Autor.

Tomando en cuenta los valores resultantes en la tabla anterior se puede observar que las configuraciones con mayor puntaje son las de Ackerman y la diferencial, optando por la configuración diferencial ya que se presenta como la solución más eficiente y más sencilla de todas para este proyecto. Consta de dos ruedas situadas diametralmente opuestas en un eje perpendicular a la dirección del robot.

Cada una de ellas dotada de un motor, de forma que los giros se realizan dándole diferentes velocidades a estos.

2.1.2 Movimientos del robot

En la Figura 25 se observa un modelo esquemático de la tracción diferencial donde se resaltan los parámetros relacionados con los movimientos. El valor *b* es la distancia entre ejes y *r* es la distancia del robot al centro de giro. Con estos parámetros básicos, podemos establecer una serie de relaciones:

$$\begin{aligned} S_i &= r\theta \\ S_d &= (r+b)\theta \\ S_c &= (r+b/2)\theta \end{aligned} \qquad (2.1)$$

Donde S_i es la distancia recorrida por la rueda izquierda y S_d es la recorrida por la rueda derecha. También tenemos S_c, que es la distancia recorrida por el eje central del robot, que tomaremos como el punto medio entre los ejes de las ruedas. En todos los casos θ es el ángulo de giro expresado en radianes (360º equivale a 2π radianes).

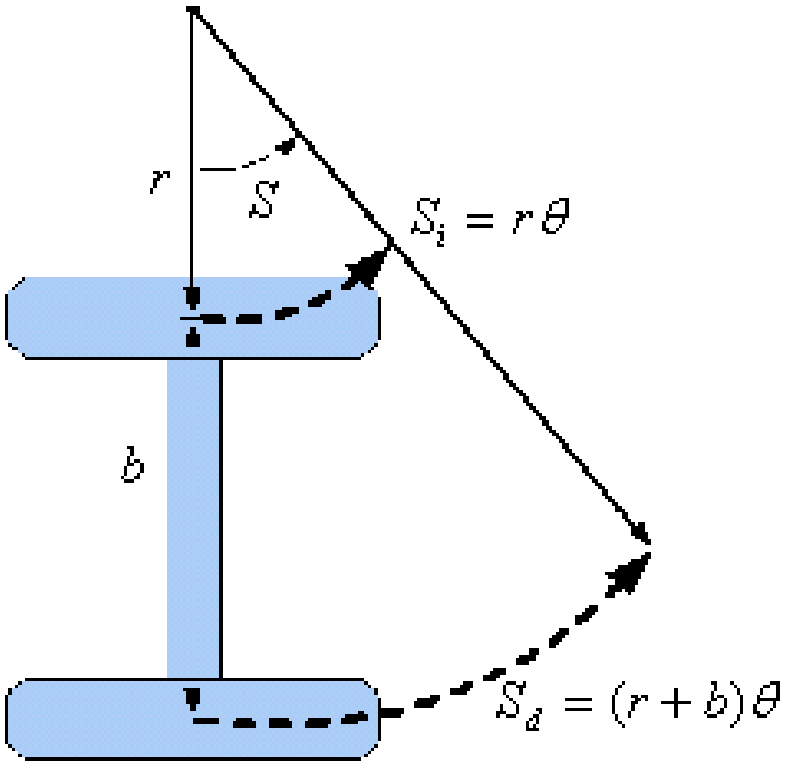
Una vez que tenemos descrito en términos matemáticos los parámetros que afectan al giro, pasamos a analizar cual es la trayectoria que seguirá. Esto solo es válido si se garantiza una condición, que la velocidad de las ruedas debe ser constante en todo momento. Si las ruedas varían su velocidad, el robot ya no traza una trayectoria circular, sino puede tomar formas más complejas.

En el sistema 2.1 tenemos relacionados tanto el avance de las ruedas, como el centro *r* y el ángulo θ que nos describen el giro. Si en este sistema, despejamos las variables θ y *r*, obtendremos las ecuaciones válidas para calcular el radio de

giro a partir únicamente de la información del avance de las ruedas, que se puede obtener a partir de los encoders:

$$\theta = \frac{S_d - S_i}{b}$$
$$r = \frac{S_i b}{S_d - S_i} \qquad (2.2)$$

Figura 25. Modelo del robot



Quintero Daniel, Introducción a la odometría.

Las anteriores ecuaciones nos describen el giro respecto al extremo interior del robot y no están definidas según coordenadas cartesianas del plano. Pueden darnos algunos detalles importantes sobre el comportamiento de la tracción diferencial, que quizás se conocen de forma intuitiva, pero de los cuales hasta ahora no teníamos la demostración matemática. Veamos algunos casos:

- Giro sobre si mismo

$$S_i = -S_d : \begin{cases} \theta = \dfrac{S_d - S_i}{b} = \dfrac{2S}{b} \\ r = \dfrac{S_i b}{S_d - S_i} = -\dfrac{b}{2} \end{cases} \qquad (2.3)$$

En el caso que ambas ruedas avancen lo mismo, pero en sentido contrario, obtenemos una distancia de *-b/2*, por lo que el centro de giro esta sobre el centro del eje (recordemos que el centro de coordenadas es la rueda mas cercana al centro de giro), y el ángulo θ, es proporcional a la distancia recorrida. Las anteriores son las condiciones en las que el robot gira sobre si mismo pero sin avanzar.

- Avance en línea recta

$$S_i = S_d : \begin{cases} \theta = \dfrac{S_d - S_i}{b} = 0 \\ r = \dfrac{S_i b}{S_d - S_i} = \infty \end{cases} \qquad (2.4)$$

Si ambas ruedas avanzan la misma distancia, tenemos un ángulo θ igual a 0, y lógicamente un radio de giro ∞, es decir, el robot no gira en ningún momento. Podríamos interpretar la línea recta como el arco de un circulo de radio infinito, o simplemente que no existe centro de giro. En cualquier caso es obvio que hemos avanzado en línea recta una distancia *S*.

2.2 DISEÑO MECÁNICO

2.2.1 Selección de motores eléctricos DC

Al igual que en la selección de la configuración del robot, para la selección de los motores se realiza una tabla de comparación entre tres referencias de motores de corriente continua, para la creación de la tabla se tienen las cuatro características

o necesidades mas importantes para el diseño, asignando un factor de importancia para cada una de ellas, estas características son en orden de importancia: el torque, por las necesidades de fuerza; el tamaño, por la disposición en el robot sin olvidar que se trata de un prototipo; el consumo de corriente, debido a las limitaciones de la fuente de energía; y finalmente la posibilidad de conseguirlo en el mercado.

Sabiendo lo anterior se realiza la calificación de cada característica para los tres motores, se multiplica este valor por el factor de importancia de la característica y se calcula la sumatoria para cada motor seleccionando finalmente el de mayor puntaje. A continuación se presenta la tabla:

Tabla 7. Selección de motores eléctricos DC

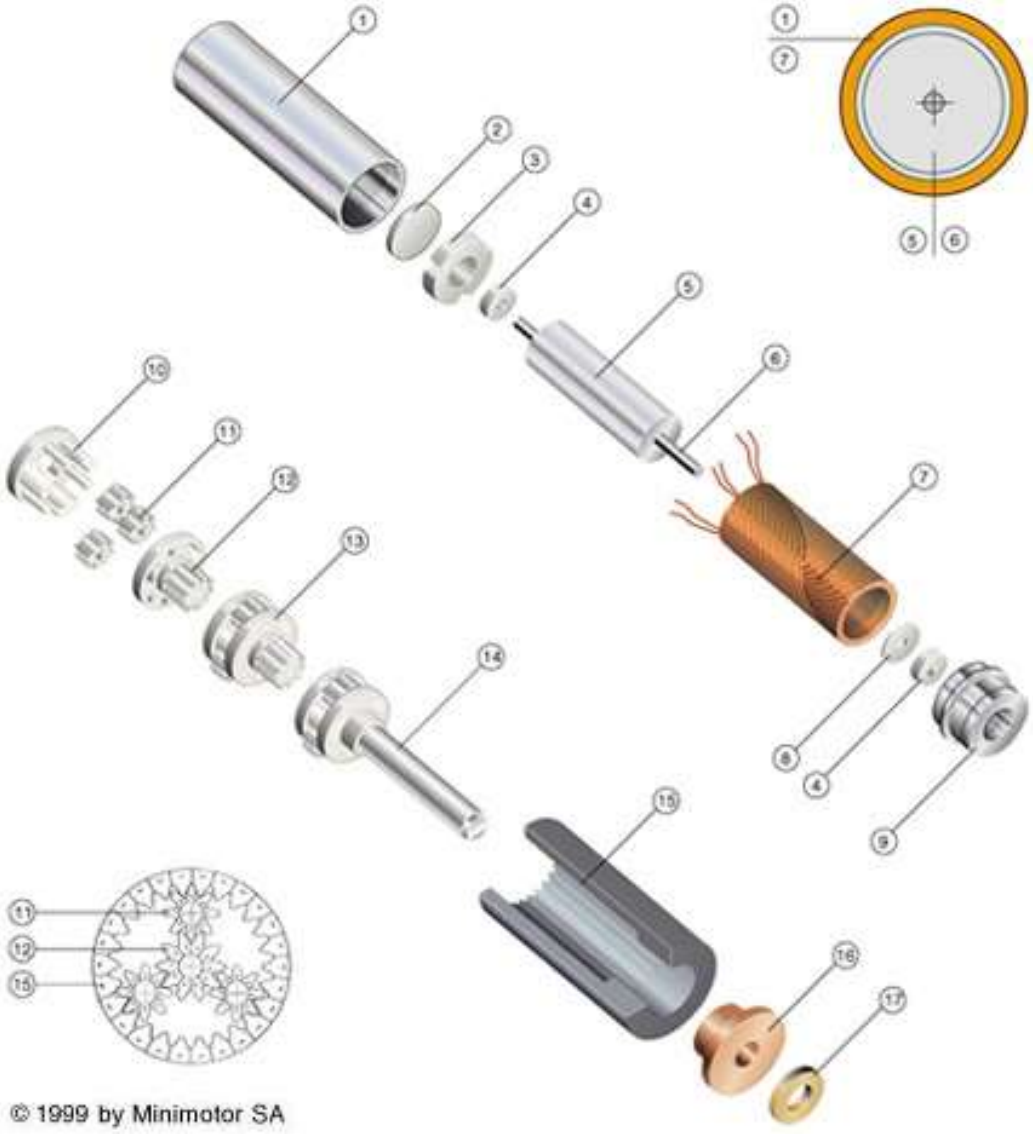
	REFERENCIA DE MOTORES		
NECESIDAD	MINIMOTOR 1612	MINIMOTOR 1624	MOTOR 12V CON REDUCTOR
Torque (5)	5	5	2
Tamaño (4)	2	4	5
Consumo (3)	3	4	4
Mercado (2)	3	3	2
Total	48	59	46

Autor.

Teniendo en cuenta los resultados de la tabla, a implementar en el robot es el Minimotor 1624, que puede funcionar hasta un voltaje de alimentación e 24VDC.

Este motor además cuenta con un encoder acoplado internamente, lo cual hace más sencilla su implementación, evitando los errores propios de una adecuación de este encoder fuera de fábrica. La Figura 26 ilustra la vista explosionada del motor.

Figura 26. Minimotor reductor

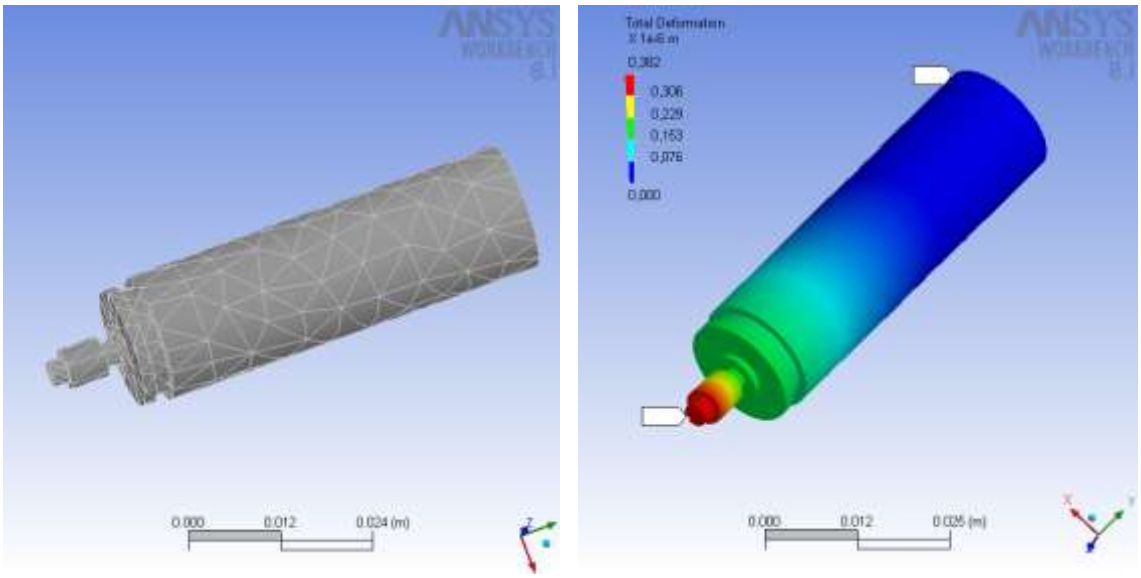


Datasheet Minimotor 1624.

- Prueba al Minimotor

Se determina la fuerza máxima para que no se presente deformación en el eje del motor por medio del software Ansys Workbench 8.1, el cual realiza una simulación por elementos finitos que permite simular cargas y esfuerzos en estructuras modeladas en CAD. La Figura 27 refleja los resultados de esta prueba.

Figura 27. Enmallado y deformación en el eje del motor



Autor.

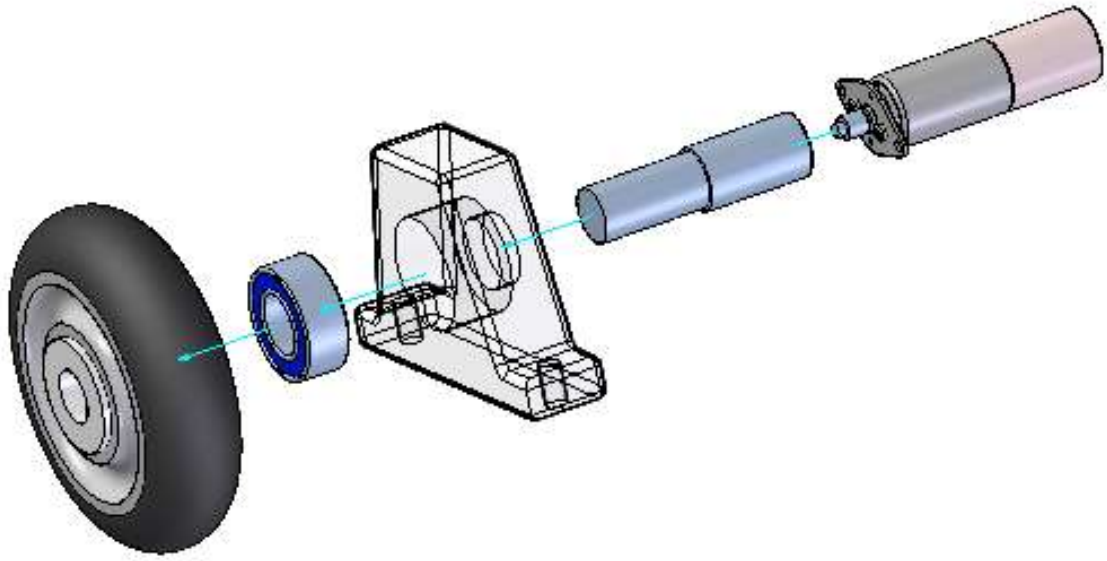
2.2.2 Acople para el motor

Con el análisis de carga desarrollado se determina que la carga máxima que puede soportar cada eje del motor es de 2kg, superando esta carga se vería la presencia de deformación en el eje del motor, siendo este el punto más critico del motor. Para este tipo de inconveniente se optó por diseñar un sistema de chumaceras para soportar la carga en cada motor, especialmente las fuerzas radiales a las que puede estar sometido.

Cada sistema de chumacera cuenta con un rodamiento radial y un buje de acoplamiento para transmitir el movimiento desde el motor hasta cada una de las ruedas directamente.

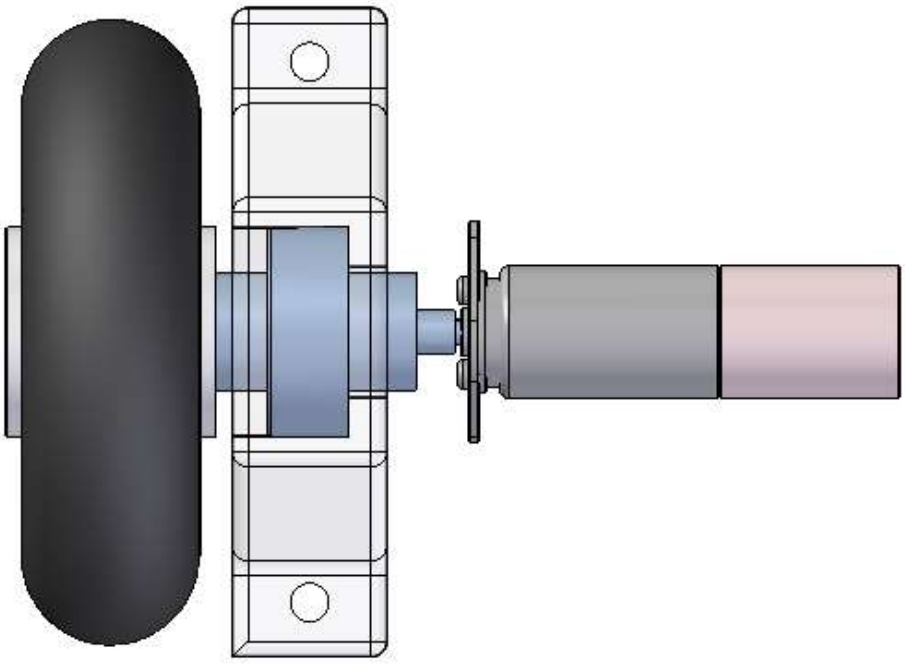
Las Figura 28 y la Figura 29 muestran en detalle el diseño del acople para los motores DC que realizarán el desplazamiento del robot.

Figura 28. Vista explosionada del acople del motor



Autor.

Figura 29. Armado completo de acople y motor

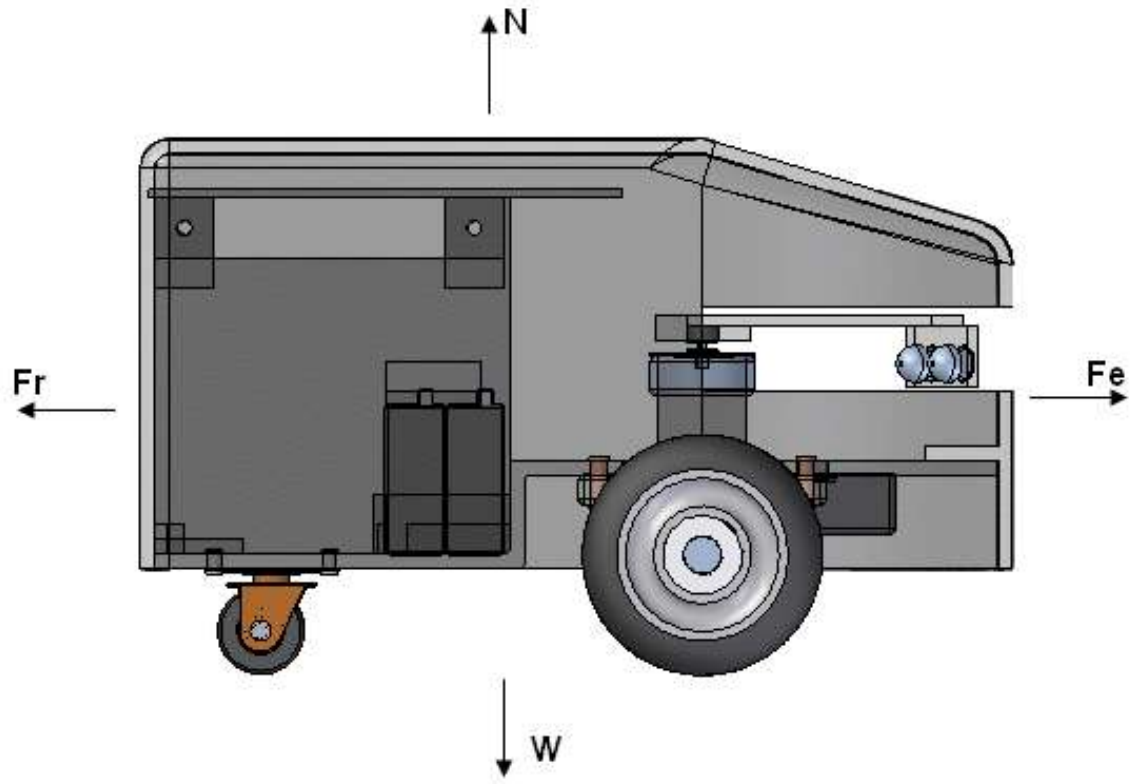


Autor.

2.2.3 Análisis de fuerzas y energía

- Análisis en plano horizontal

Figura 30. Fuerzas en plano horizontal



$$\mu = 0.9 \qquad \omega = 85rpm \qquad V = 12v \qquad r = 0.040 \qquad W = 2.1Kgf$$

$$\sum Fx = m * a$$

Autor.

Conociendo el coeficiente de fricción estático (μ) establecido para el cemento y el caucho (Ver Tabla 8), tomamos el robot en reposo para así hacer el cálculo y hallar la fuerza de empuje que requiere el robot para poder desplazarse, conociendo que su aceleración es cero por estar en reposo. Podemos concluir que $\sum Fx = 0$, entonces $Fr = Fe$.

En el eje Y se tiene que $N = W$, entonces:

$$Fr = \mu \times N$$
$$Fr = 0.9 \times 2.1Kgf \quad (2.5)$$
$$Fr = 1.89N$$

Conociendo el valor Fr, se debe aplicar una fuerza mayor a esta para que haya movimiento.

La fuerza de empuje Fe se distribuye entre los 2 motores del robot, así la fuerza de empuje en cada motor será $Fe/2$, entonces $Fe\_motor = 0.945N$.

Conociendo Fe_motor y conociendo que el par resistente es $T = Fe\_motor \times r$ se tiene que $T = 0.945N \times 0.040m$, $T = 0.0378Nm$.

Con lo anterior se puede calcular la potencia mecánica (Pm) en le momento de arranque conociendo el par resistente que realiza cada motor.

$$Pm = T \times \omega \quad (2.6)$$

Conociendo características propias del motor, velocidad angular 85rpm y par resistente de 0.0378Nm, entonces:

$$Pm = 1.02695 \times 0.0378Nm \times 85rpm$$
$$Pm = 3.29 \quad (2.7)$$

Por consiguiente se calcula la potencia eléctrica (Pe) y la corriente (i) que consume el motor para generar la potencia Pm.

$$Pm = Pe - Pp \quad (2.8)$$

Asumiendo la potencia perdida $(Pp = 0)$ como cero y conociendo que $Pe = i \times v$, entonces:

$$i = \frac{Pm}{v} \Rightarrow i = \frac{3.29}{12v} = 0.274A \quad (2.9)$$

En conclusión para que el robot pueda obtener movimiento se debe suministrar una corriente de 274 mA para cada motor, siendo una corriente total de 548 mA que consumirán los dos motores del robot.

Tabla 8. Coeficientes de fricción del caucho

Superficie	Coeficiente
Caucho sobre terreno firme	0.4 - 0.6
Caucho sobre concreto seco	0.9 - 1.0
Caucho sobre concreto húmedo	0.6 - 0.8
Caucho sobre asfalto	0.8 - 1.0

Autor.

- Análisis en plano inclinado

En el análisis del plano inclinado (ver Figura 31) aparece una nueva fuerza que se opone a la Fe, manteniendo una velocidad constante ya que la aceleración es cero, el sistema se plantea de la siguiente manera:

$$\sum Fx = Wsen\phi + Fr - Fe = 0 \quad (2.10)$$

Partiendo de:

$$N = W\cos\phi \Rightarrow N = 2.1kgf \times \cos 10^{\circ} \Rightarrow N = 2.06N \quad (2.11)$$

Luego

$$Fr = \mu \times N \Rightarrow Fr = 0.9 \times 2.06 \Rightarrow Fr = 1.85N \quad (2.12)$$

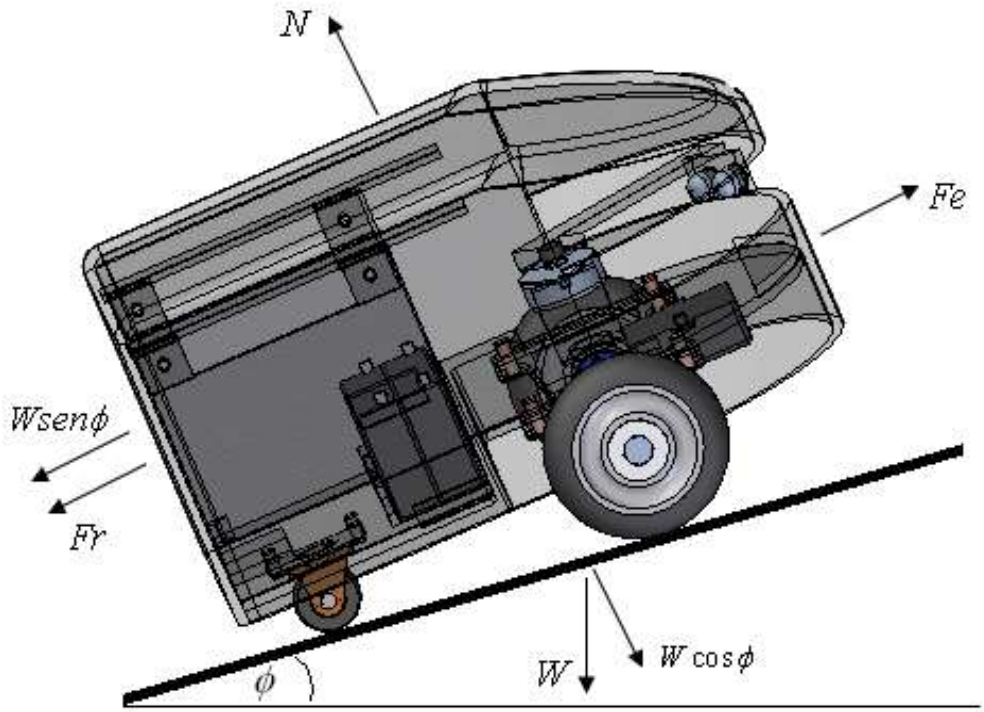
Entonces

$$Fe\_motor = 1.105N \quad (2.13)$$

Con lo cual hallamos el par resistente:

$$Pm = 1.02695 \times 0.0442 \times 85 = 3.85$$
$$i = \frac{Pm}{v} \Rightarrow \frac{3.85}{12} = 320mA \quad (2.14)$$

Figura 31. Fuerzas en plano inclinado



$$\mu = 0.9 \quad W = 2.1Kgf \quad V = 12v \quad r = 0.040m \quad \omega = 85rpm \quad \phi = 10°$$
$$\sum Fx = m * a$$

Autor.

La corriente necesaria para el movimiento del robot en un plano inclinado de 10°, es de 320 mA por cada motor, es decir, 640mA por los dos motores.

Según los parámetros de diseño el robot debe consumir un máximo de 650mA, de los cuales 100mA los consume el FPGA y los otros dispositivos, de donde queda

entonces 550mA para el consumo de los motores. En conclusión, el funcionamiento del robot será, como se mencionó anteriormente, sobre un plano horizontal, donde el consumo total de los motores DC es de 548mA.

2.2.4 Diseño final

Al inicio del proyecto se evaluaron varios tipos acoplamientos para los motores con sus respectivas ruedas, teniendo problemas por que el peso del robot afectaba directamente el eje de los motores, dando solución a este problema con un sistema de chumaceras que soportan todo el peso sobre ellas.

El primer diseño tuvo unos cambios de alturas sencillamente por no tener en cuenta la altura de la ruedas “loca” o rueda de giro libre (necesaria en la configuración diferencial y ubicada en la parte de atrás el robot), que se conseguían en el comercio local, dando como solución hacer un dobles hacia abajo en la parte trasera del chasis.

En este soporte trasero, que queda ubicado en un nivel inferior al que sostiene los acoples de las ruedas, se opto por ubicar los soportes para las dos baterías de 6VDC, que alimentarían todo el sistema con un voltaje total de 12VDC (ubicadas en serie), igualmente aquí se instalan los sopotes para los circuitos impresos, y encima de estos últimos se encuentra la base para la tarjeta de desarrollo que integra el FPGA o dispositivo lógico programable.

Volviendo a la parte delantera del chasis, sobre esta se ubica el soporte del motor paso a paso que guiará el sensor de detección de obstáculos (ultrasónico). Este motor tendrá un ángulo de acción de 180º, centrado sobre la parte delantera del robot, es decir, el sensor detectará obstáculos justo al frente del robot cuando el motor paso a paso este ubicado a un ángulo de 90º con respecto a su origen, que

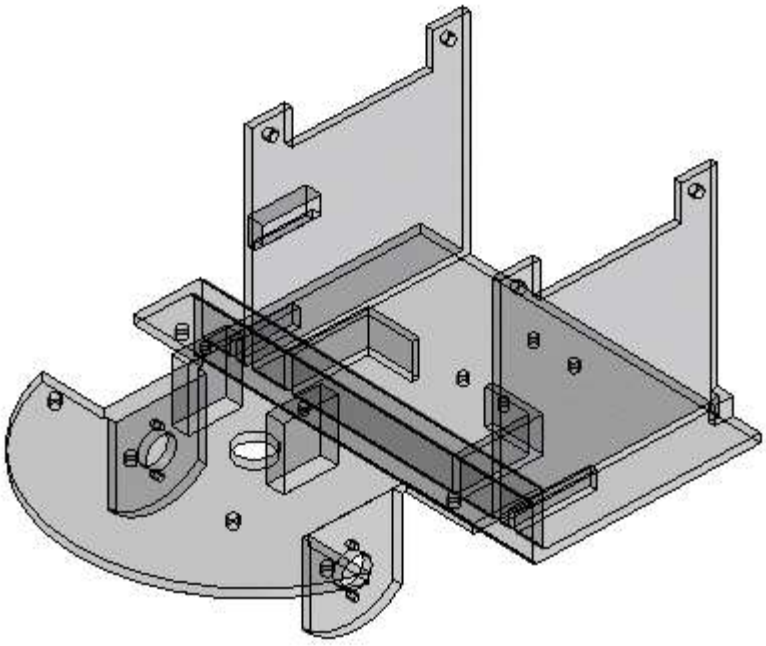
para efectos de programación se ubica a la derecha del robot, teniendo en el límite izquierdo el ángulo de 180º.

Igualmente, por debajo de esta base delantera del chasis y en medio de los acoples de los motores DC de avance, se ubica el módulo del sensor ultrasónico, que se explicará mas adelante.

Finalmente cabe destacar que el acople de los dispositivos y accesorios al chasis se realiza con un sistema de tornillo, arandela, “guasa” (que impide que la tuerca se afloje) y tuerca.

La siguiente figura ilustra el diseño final del chasis.

Figura 32. Diseño final del chasis del robot



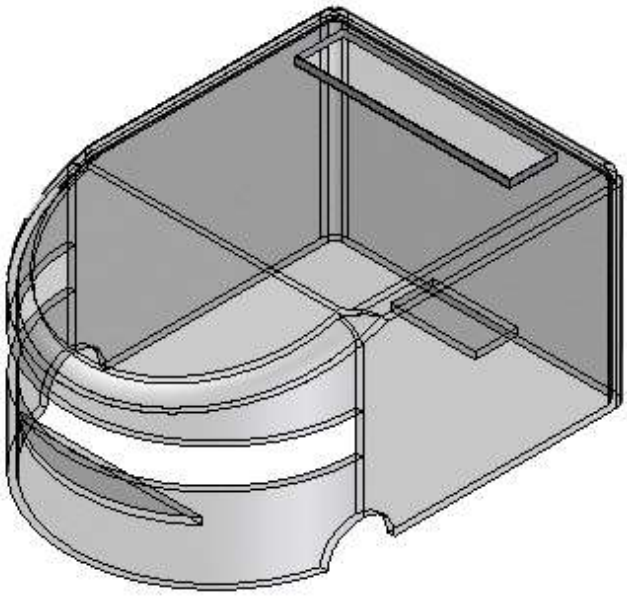
Autor.

El chasis antes descrito será fabricado en acrílico teniendo en cuenta que es un material liviano y al mismo tiempo muy resistente y sencillo de trabajar, esto facilita

el diseño teniendo en cuenta la corriente a consumir por los motores y el costo, que comparado con otros materiales es menor (Véase propiedades físico-químicas del acrílico en el anexo A2).

La carcaza del robot (Figura 33), al igual que el chasis será fabricada en acrílico de un grosor de 5mm en color humo, el diseño consta de una abertura en la tapa superior, para que el FPGA pueda ser manipulado por el usuario y para poder observar el estado del robot a través de la pantalla LCD. Además cuenta con otra abertura en la parte frontal para el correcto funcionamiento del sensor de obstáculos.

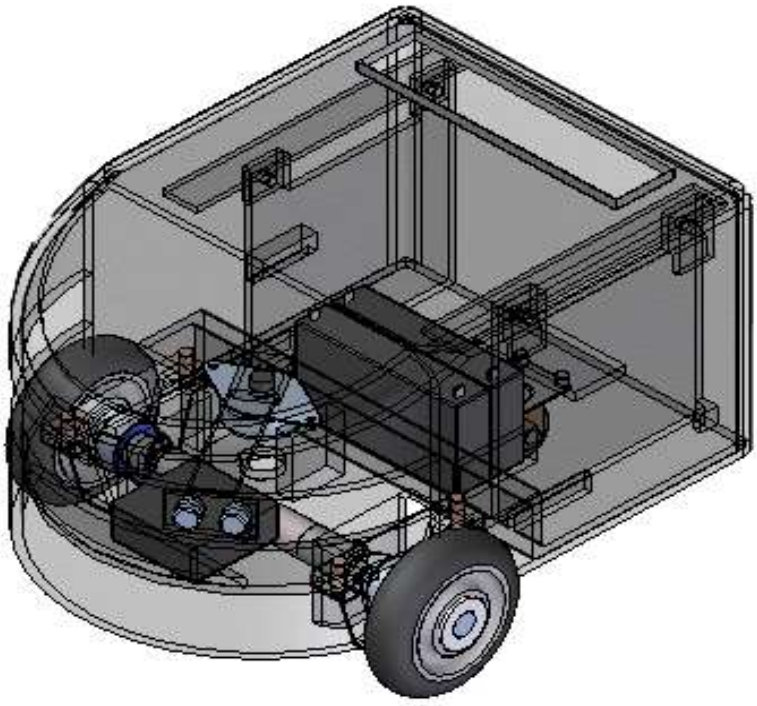
Figura 33. Carcaza del robot



Autor.

La Figura 34 ilustra el diseño completo del robot que con la carcaza anterior y el chasis que incluye: el acople de las ruedas de desplazamiento junto a las ruedas y los motores DC respectivos; el soporte del motor paso a paso junto al sensor ultrasónico; el módulo del sensor; las bases para los circuitos impresos; y el soporte para las baterías.

Figura 34. Diseño mecánico final



Autor.

2.3 CONFIGURACION Y CONTROL DE SENSORES Y ACTUADORES

2.3.1 Módulo sensor ultrasónico

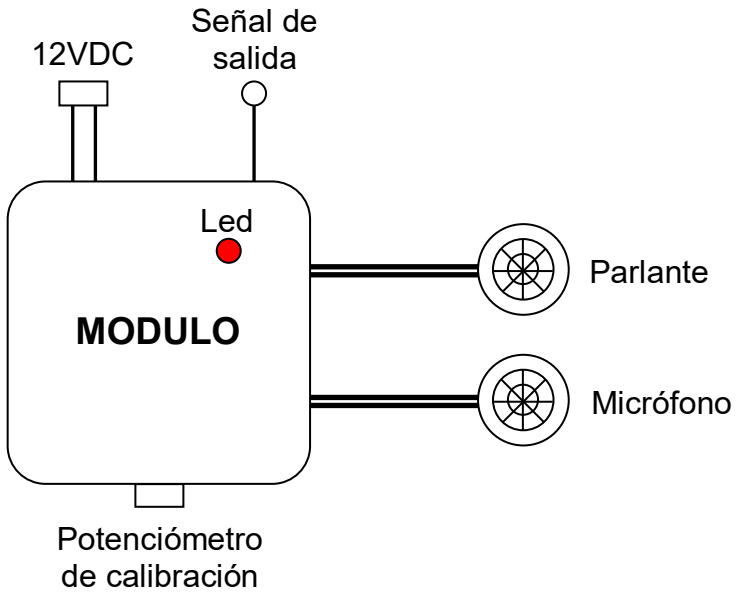
Por la experiencia personal y de otros proyectos de la facultad, en el montaje de este tipo de sensores suelen ocurrir toda una serie de inconvenientes que se relacionan a continuación:

- Falta de precisión en el generador de la señal de frecuencia de 40KHz
- Mal funcionamiento del micrófono o del receptor
- Mala amplificación de la señal el receptor

Estos inconvenientes se presentan generalmente por soldadura, ya que las señales al ser tan “pequeñas”, fácilmente se pueden perder o ser afectadas por ruido, lo que lleva a un mal funcionamiento del sensor.

Para evitar lo anterior, se ha seleccionado un módulo de sensor ultrasónico, que en el mercado se usa en alarmas para automóviles. Este módulo contiene los circuitos para la generación de la señal de excitación del micrófono y para la amplificación de la señal percibida por el receptor, viene con el micrófono y el parlante para ultrasonido con su cable de conexión blindado para evitar la presencia de ruido en la señal, funciona con una alimentación de 12VDC (que la provee la fuente del robot), tiene un potenciómetro con el cual se gradúa la distancia de a la cual se debe detectar el obstáculo, un led que se enciende al detectar un obstáculo, y una señal de salida que también indica la detección del obstáculo; esta última será utilizada por el FPGA. La siguiente figura muestra un esquema del módulo.

Figura 35. Módulo sensor ultrasónico

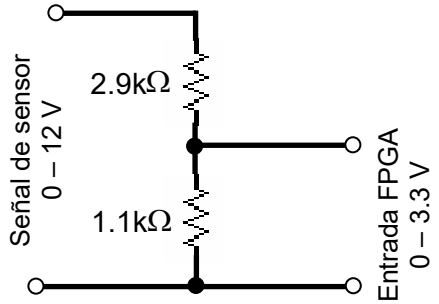


Autor.

Normalmente se tiene una señal de salida constante de amplitud 12VDC, cuando el sensor detecta un obstáculo, la salida genera pulsos que varían entre 0 y 12VDC, generalmente son entre 3 y 5 pulsos, después de esto vuelve a 12VDC aunque el obstáculo aún este presente. Lo anterior es de especial cuidado en la programación, porque esta salida funciona como una interrupción, de modo que el obstáculo puede estar presente y el sensor no estará indicando esta situación. En la sección del motor paso a paso se indica la forma en que se evita este problema.

El voltaje máximo de funcionamiento de los pines de entrada del FPGA es de 3.3V, por lo cual la señal del sensor no se debe aplicar directamente a estos pines. Para esto se incorpora un sencillo circuito divisor de voltaje compuesto por resistencias y presentado en la siguiente Figura 36. De este modo la señal que recibe el FPGA es de 3.3V en funcionamiento normal del sensor. Cuando se detecte un obstáculo, el FPGA recibirá los pulsos, pero esta vez variando entre 0 y 3.3V.

Figura 36. Divisor de voltaje para modulo sensor ultrasónico



Autor.

El parlante y el micrófono ultrasónicos se ubicarán en un soporte de modo que la parte frontal de estos se encuentre ubicada sobre el mismo plano, así, el parlante envía la señal sonora y cuando se presente un obstáculo, esta rebotará y será percibida por el micrófono.

El potenciómetro del módulo se ha de calibrar de modo que el sensor detecte los obstáculos a una distancia 15cm mayor a la que exista entre el punto de giro y la parte más lejana a este punto en el montaje físico del robot. Los 15cm se toman como medida de tolerancia, y de esta calibración depende que el robot detecte los obstáculos a tiempo para que no se estrelle, especialmente por la parte trasera cuando este girando, donde el robot es ciego.

2.3.2 Configuración motor paso a paso

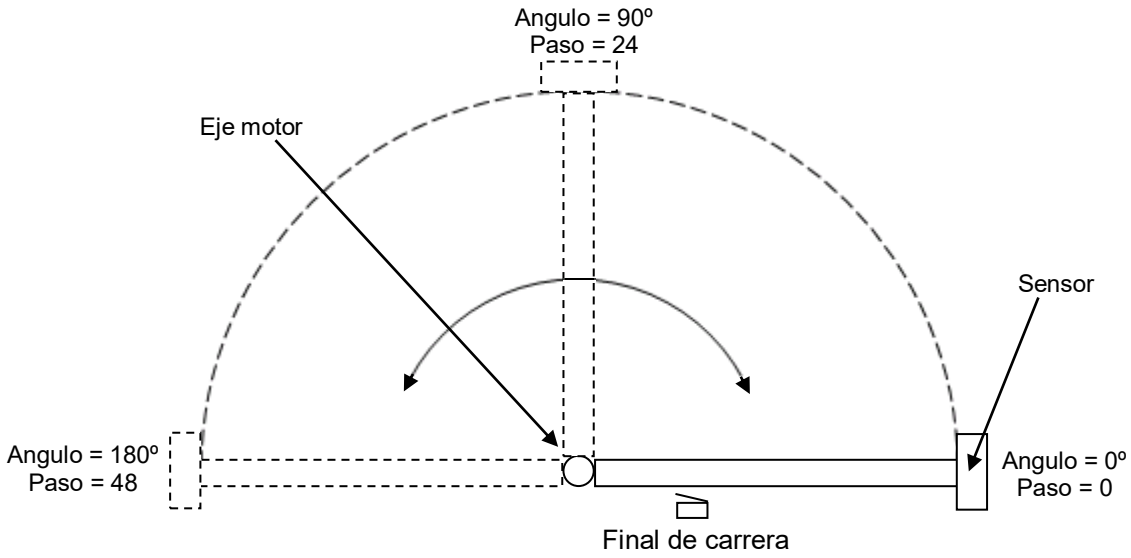
La conexión eléctrica del motor paso a paso es muy sencilla, basta con remitirse al marco teórico en el apartado que se refiere al driver ULN2003, o al datasheet del mismo (Anexo). Hay que decir que el motor es unipolar, se alimenta a 12VDC, tiene 6 cables, de donde cuatro de ellos son las 4 bobinas, y los otros dos son los comunes, que se conectan a 12VDC, ya que el driver es de lógica inversa.

La entrada del driver funciona con lógica CMOS, esto nos indica que no hay que realizar ningún acople entre las salidas del FPGA (0V o 3.3V) y las entradas del driver. El FPGA envía 1 lógico (3.3V) para activar la(s) bobina(s) según sea la secuencia, las demás quedan a 0 lógico; el driver invierte estas entradas, aplicando 0V en las entradas de la(s) bobina(s) a energizar para suministrar la potencia requerida.

El motor tiene una resolución de 7.5º por paso, para aumentar esta resolución se recurre a maniobrarlo utilizando la secuencia de medio paso, descrita en el marco teórico, de este modo la resolución aumenta a 3.75º por paso. En esta secuencia, habrá instantes en que solo esté energizada una bobina, y otros en los cuales estarán energizadas dos bobinas.

Con esta resolución y observando la Tabla 2 tenemos que el motor dará un giro completo en 96 pasos, sin embargo para el robot este motor dará medio giro (Ver sección 2.2.4), es decir, 180º que equivalen a 48 pasos. El movimiento irá de derecha a izquierda y viceversa, de este modo el sensor tiene un "campo de visión" de 180º centrado frente al robot. Véase la siguiente figura.

Figura 37. Angulo funcionamiento de motor paso a paso



Autor.

Sin embargo, al momento de encender el robot, se desconoce la posición que tiene el sensor, es decir, el eje del motor paso a paso. Es por esto que se incorpora un sencillo final de carrera, ubicado en la parte derecha del robot (visto desde arriba), un paso más atrás del límite derecho en el "campo de visión" del sensor. Este final de carrera tiene tres pines, uno común, otro normalmente cerrado y el último normalmente abierto; el común va conectado a una entrada del FPGA con una resistencia de protección, el normalmente cerrado se conecta a 0V y el normalmente abierto irá conectado a 3.3V. En el momento que se active el final de carrera, la FPGA recibirá un 1 lógico, de lo contrario se recibe 0 lógico.

Cuando se enciende el robot, el motor paso a paso gira hacia la derecha hasta que se active el final de carrera, en este punto el motor gira un paso hacia la izquierda. En este punto el eje del motor (y por consiguiente el sensor ultrasónico) se encuentra ubicado en el límite derecho del "campo de visión", tomado como el origen o ángulo 0º. De este punto en adelante el eje del motor va y vuelve describiendo media circunferencia centrada con el frente del robot, así cuando se han avanzado 24 pasos a partir del origen, el sensor se encuentra en el centro; y cuando se avanza 48 pasos, el sensor ha alcanzado el límite izquierdo.

Finalmente, el tiempo de espera entre paso y paso se estima en 25ms con lo cual el sensor tardará 2400ms, es decir, 2.4s en ir desde el punto 0 hasta el límite izquierdo y volver al punto 0.

2.3.3 Controlador de velocidad para motores DC

Teniendo en cuenta la condición que fue planteada en la sección 2.1.2 y que indica que "la velocidad de las ruedas debe ser constante en todo momento", se debe diseñar un controlador que haga cumplir dicha condición.

- Medición de la velocidad del motor

Lo primero que se necesita es conocer la velocidad a la cual está girando cada una de las ruedas en todo momento, esto se logra a través de los encoder. Para empezar se toma la ecuación 1.2 y se reemplazan los valores del encoder obteniendo:

$$D = \frac{\pi\phi}{nC} = \frac{\pi(6.5cm)}{1*1213} = 1.68x10^{-2}\, cm/\, pulso \qquad (2.15)$$

Lo que indica que entre pulso y pulso del encoder, la rueda avanza 0.0168cm/pulso. Ahora por medio del FPGA se puede calcular el tiempo que transcurre entre pulso y pulso, es decir, el periodo del encoder. Con estos dos datos, podemos conocer en todo momento la velocidad de cada motor según la relación:

$$V = \frac{D}{T} = \frac{1.68x10^{-2}\,cm/\,pulso}{(t)ms/\,pulso} = \frac{1.68x10^{-2}}{(n)}\,cm/ms \qquad (2.16)$$

Donde *(t)* representa el periodo del encoder en milisegundos.

- Velocidad deseada

Ahora se procede a determinar el "*set point*" de la velocidad, es decir, la velocidad a la que se quiere que el robot avance o gire. Teniendo la distancia de detección de obstáculos que es de *15cm* y el tiempo que el sensor de obstáculos tarda en regresar a un mismo punto, el cual es de *2.4s*, se calcula la velocidad máxima a la que el robot se deberá desplazar, obteniendo:

$$V = \frac{15cm}{2.4s} = 6.25cm/s = 6.25x10^{-3}\,cm/ms \qquad (2.17)$$

Con lo anterior se establece el valor de *5cm/s* como la velocidad deseada para el desplazamiento del robot, es decir, el "*set point*" para el controlador.

Por motivos prácticos de programación es mejor tener la velocidad en términos de tiempo entre pulso y pulso del encoder, para poder compararla con el valor de periodo que se mide. Para pasar a estos términos el valor de "*set point*" despejamos la parte izquierda de la ecuación 2.16 y reemplazamos:

$$T = \frac{D}{V} = \frac{1.68x10^{-2}\,cm/\,pulso}{5cm/s} = 3.37ms/\,pulso \qquad (2.18)$$

- Consideración especial para el "*set point*"

Durante el funcionamiento del robot se puede presentar una situación inesperada, en la que a pesar de estar funcionando el controlador, una rueda se mueva considerablemente mas lento que la otra, esto puede ser consecuencia de factores mecánicos, como por ejemplo la presencia de pequeñas "cuñas" u obstáculos que no alcanzan a ser detectados por el sensor ultrasónico pero que el robot es capaz de empujar, con la consecuencia de la disminución de la velocidad en la rueda afectada.

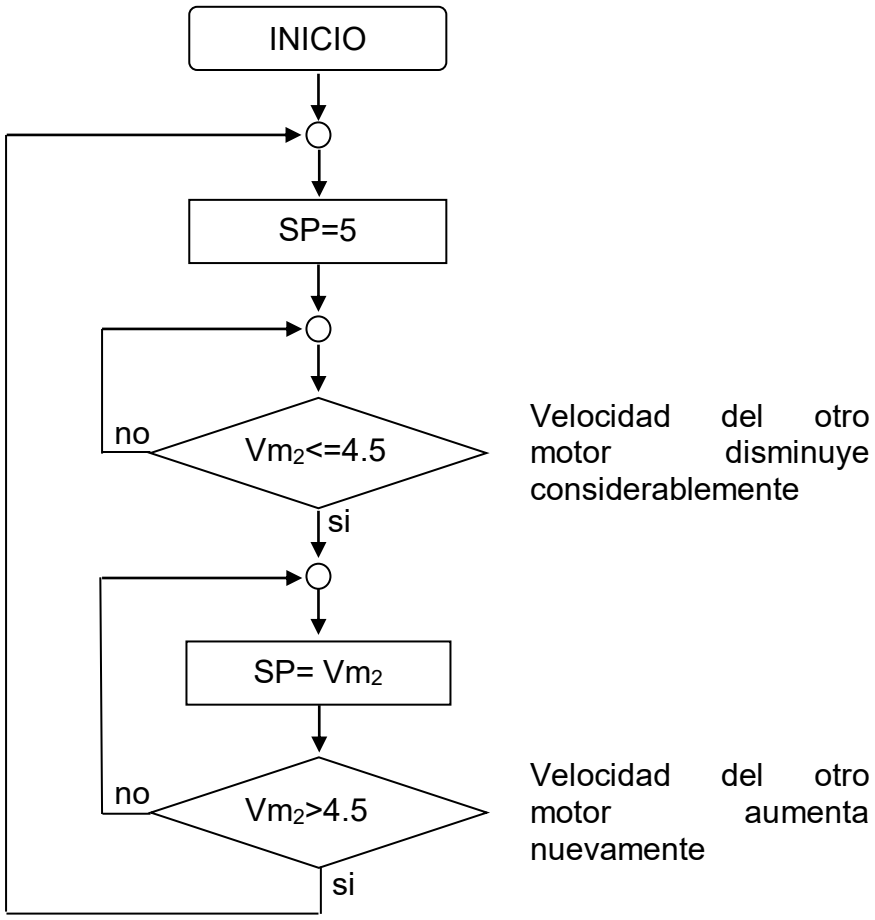
Si se presenta la situación anterior y el robot no se percata de esto, es decir la otra rueda se continúa a la velocidad predefinida, el robot no cumplirá la condición descrita al inicio de esta sección y no se desplazará según los parámetros de diseño seleccionados (giro sobre si mismo y avance recto).

Es por esto que se ha creado un algoritmo para establecer el valor de "set point" en cada controlador según el cual después de iniciar cualquier desplazamiento en el robot, es decir, de salir del reposo; si alguna de las ruedas se mueve a una velocidad promedio (medida durante el último medio segundo de funcionamiento) igual o inferior a 4.5cm/s, en la otra se modificará el valor de "*set point*" de su controlador al valor de velocidad inferior, ahora si la velocidad de la rueda que estaba lenta al principio vuelve al valor predeterminado, posible porque el valor de "*set point*" de su controlador no se modifica; entonces el controlador de la segunda rueda retornara al valor normal de "*set point*".

Se considera el anterior algoritmo como un mecanismo de seguridad para garantizar el buen desplazamiento del robot según los requerimientos de su funcionamiento.

La siguiente figura ilustra el diagrama de flujo de este algoritmo de seguridad, que se aplica para cada motor. *SP* hace referencia al "*set point*" del motor sobre el cual se aplica el algoritmo y Vm_2 a la velocidad promedio del otro motor.

Figura 38. Diagrama selección de "*set point*" para controlador de velocidad



Autor.

- Controlador

Se implementa un controlador proporcional, donde la ganancia se aplica al error entre el "*set point*" y la velocidad real del motor, datos vistos en relación al periodo del encoder. Se establece que se aplican 10 acciones de control por segundo en cada controlador.

- Voltaje de referencia para la velocidad

La salida del controlador se aplica a cada motor a través del driver que lo opera, esto se hace siguiendo una secuencia, como primera medida, la salida del controlador en el FPGA se convierte en un dato digital de 12 bits que es enviado al conversor digital-análogo (ver sección 1.5.2 donde hace referencia al conversor), el cual convierte este dato en un voltaje proporcional en el rango de 0 a 2.5V, esta señal pasa por un amplificador no inversor que cambia este voltaje proporcionalmente al rango 0 a 12V. Finalmente, este último voltaje se aplica al driver, con lo cual este maneja la potencia que entrega al motor (que se traduce en velocidad).

Para el correcto funcionamiento del amplificador no inversor, la ganancia esta dada por:

$$Ganancia = k = \frac{Vsalida}{Ventrada} = \frac{12V}{2.5V} = 4.8 \qquad (2.19)$$

Para definir esta ganancia se aplica el modelo del amplificador no inversor, visto en la sección 1.4.3, de la siguiente forma:

$$k = \frac{(R_1 + R_2)}{R_1} = 1 + \frac{R_2}{R_1} = \qquad (2.20)$$

Teniendo un valor para *R1*=1kΩ tenemos que:

$$\begin{aligned} R_2 &= k * R_1 - R_1 \\ R_2 &= 4.8 * 1k\Omega - 1k\Omega \\ R_2 &= 3.8k\Omega \end{aligned} \qquad (2.21)$$

Con esto queda definido el valor de las dos resistencias para la operación del amplificador según la ganancia necesaria.

2.4 PROGRAMACION DEL ROBOT

2.4.1 Estructura de las unidades de diseño en VHDL

Las dos unidades de diseño en VHDL que se utilizan para la programación del robot son la entidad y la arquitectura. En la primera se establecen las señales de entrada y de salida del FPGA, es decir, las interfaces y dispositivos con los cuales interactúa. En la segunda se realiza la descripción hardware del funcionamiento del robot, esto es, la escritura del código que representa (en ocasiones con un alto nivel de abstracción comparado con sencillos circuitos digitales) los algoritmos y funciones que conllevan a una correcta operación del robot móvil.

- Declaración de la entidad

Tabla 9. Señales de entrada y salida del FPGA

GRUPO	NOMBRE SEÑAL	DESCRIPCION	UBICACIÓN EN TARJETA
	CLOCK	Reloj tarjeta de desarrollo (50MHz)	Interna
PANTALLA LCD	LCD_E	Habilitación de LCD	Interna
	LCD_RS	Modo comando / carácter LCD	
	LCD_RW	Modo lectura / escritura LCD	
	SF_D	Dato enviado a LCD	
BOTONES USUARIO	B_ARRIBA	Botón según la posición que indica el nombre	Botones usuario
	B_ABAJO		
	B_IZQUIERDA		
	B_DERECHA		
	B_CENTRO		
CDA		*Todas son señales que envía el FPGA al conversor digital-análogo*	Interna
MOTORES DC	ENC_I	Encoder motor izquierda	Conector J1
	ENC_D	Encoder motor derecha	Conector J2
	MOTOR	Sentido de giro de los motores	Conectores J1 y J2
PASO A PASO	FINAL_CAR	Señal de final de carrera (Ubicación)	Conector J2
	PAP	Secuencia de motor paso a paso	Conector J4
SENSOR	ULTRA	Señal del sensor ultrasónico	Conector J1

Autor.

La Tabla 9 describe la función de las señales. La columna denominada “UBICACIÓN EN TARJETA” indica si la señal se conecta a otro dispositivo internamente en la tarjeta, o por el contrario indica la interfaz por medio de la cual las otras señales del FPGA se conecta con los dispositivos externos a la tarjeta de desarrollo. Observando la Figura 49 se tiene claridad de la ubicación de estas interfaces en la tarjeta de desarrollo.

Figura 39. Declaración de la entidad

```
entity LCD is
    Port ( CLOCK : in  bit;
--LCD
           LCD_E : out  bit;
           LCD_RS : out  bit;
           LCD_RW : out  bit;
           SF_D : out  bit_vector (11 downto 8);
--Botones
           B_ARRIBA : in  bit;
           B_ABAJO : in  bit;
           B_IZQUIERDA : in  bit;
           B_DERECHA : in  bit;
           B_CENTRO : in  bit;
--CDA
           SPI_SS_B : out  bit;
           AMP_CS : out  bit;
           AD_CONV : out  bit;
           SF_CEO : out  bit;
           FPGA_INIT_B : out  bit;
           SPI_MOSI : out  bit;
           DAC_CS : out  bit;
           SPI_SCK : out  bit;
           DAC_CLR: out  bit;
--Motor
           ENC_I : in bit;
           ENC_D : in bit;
           MOTOR : out  bit_vector (3 downto 0);
--PAP
           FINAL_CAR : in bit;
           PAP : out  bit_vector (3 downto 0);
--UTLRASONIDO
           ULTRA : in bit;);
end LCD;
```

Autor.

La Figura 39 ilustra la sintaxis de la entidad, en ella se observa la declaración del puerto, el cual contiene las señales de entrada del FPGA (in) y las de salida (out).

- Funcionamiento del lenguaje VHDL

Como ya se mencionó el VHDL es un lenguaje de descripción de circuitos, que en este caso se implementa en un FPGA. En la mayoría de lenguajes de computadores y programación de microcontoladores, se diseña de modo que el dispositivo ejecute instrucciones secuencialmente según una memoria de programa. En el caso de VHDL no es así, aquí se indican instrucciones que se convertirán en un complejo circuito digital (compuertas, flip-flops, etc.), esto lleva a que el funcionamiento del FPGA sea en paralelo (teniendo en cuenta los retardos debidos a la cantidad de conexiones que deba cruzar cada señal). Esto modifica sobremanera la concepción tradicional sobre programación.

Sin embargo lo anterior no indica que en VHDL no se puedan programar secuencias, para esto existe un bloque especial llamado *process*, que funciona en paralelo con las demás instrucciones y bloques similares, pero internamente ejecuta instrucciones en serie cuya sintaxis es similar al código C.

En VHDL se pueden declarar constantes (*constan*t), que funcionan igual que en cualquier lenguaje, y señales (*signal*), esto es un concepto diferente al tradicional, estas interconectan instrucciones y bloques, y funcionan de modo similar a las variables con la salvedad que su operación es en paralelo. Las entradas y salidas declaradas en la entidad se comportan del mismo modo que las señales.

Dentro del bloque *process* se pueden declarar variables, que funcionan igual que en la mayoría de los lenguajes, pero estas solo tienen validez dentro del bloque en el cual se les declara.

Estos bloques no pueden declarar señales pero si las pueden modificar, sin embargo, se debe tener especial cuidado, ya que la asignación solo se realiza al final de las instrucciones que contiene el bloque, es decir, la señal no cambia tan pronto como se asigna, sino al terminar las instrucciones del bloque y obtendrá el valor de la última asignación realizada en la lista de instrucciones.

Finalmente, la activación de un bloque *process* se realiza a través de una lista sensible, esto es, una lista de señales que se indican entre paréntesis justo en frente de la palabra *process*. Cuando alguna de estas señales modifique su valor se activa el bloque y este procede a ejecutar as instrucciones secuencialmente, cuando estas terminan, se modifican las señales (si se ha asignado alguna) según la explicación del párrafo anterior y termina la ejecución el bloque. Si el bloque ha modifica alguna de las señales contenidas en la lista sensible, volverá a iniciar su ejecución.

- Estructura de la arquitectura

La arquitectura contiene, como ya se mencionó, el código que representa los algoritmos de funcionamiento. La Figura 40 ilustra la forma en que esta organizada la arquitectura, y se menciona a continuación:

- Lo primero que se tiene es la declaración de todas las señales internas del FPGA, agrupadas según su uso.

- Luego se tiene la instrucción de inicio de la descripción (*begin*).

- Ahora se presenta la descripción en si, compuesta por grupos o módulos, según la parte que se describe (Manejo LCD, retardos, controlador de motores DC, etc.), estos grupos contienen:

  Las asignaciones y operaciones en paralelo.

Los bloques *process* con sus respectivas variables y algoritmos de ejecución serie.

- Terminando con la finalización de la descripción (*end*).

Figura 40. Organización de la arquitectura del programa

```
architecture Behavioral of LCD is
   signal Ini_FPGA : boolean;
--Señales funcionamiento LCD
   signal iniciado : boolean;
   signal LCD_1 : integer range 0 to 5;
--Señales paso a paso
   signal Out_pap : bit_vector (3 downto 0);
      .
      .
begin
--|||||||||||||| MANEJO LCD ||||||||||||||||||||||
Impresion: -- Impresión de mensajes -------------
process
   variable pas : boolean:=false;
      .
      .
begin
   if Ini_FPGA=false then
      .
      .
   end if;
end process;
      .
      .
--|||||||||||||| FIN MANEJO LCD ||||||||||||||||||

--|||||||||||||| MOTOR PASO A PASO ||||||||||||||||
      .
      .
PAP<=out_pap;
--|||||||||||||| FIN MOTOR PASO A PASO ||||||||||||
      .
      .
end Behavioral;
--|||||||||||||| FIN DE PROGRAMA ||||||||||||||||||
```

Autor.

2.4.2 Subrutinas globales

En programación es indudable la necesidad de operadores para la realización de código, VHDL no escapa a esto, de hecho son indispensables para la programación de los controladores y otras rutinas, a continuación se relacionan los operadores utilizados y su significado:

+ Suma y signo positivo
- Resta y signo negativo
* Multiplicación
/ División
ABS Valor absoluto
MOD Módulo entre dos números
REM Resto de la división entera
=, /= Igualdad, desigualdad
<, <= Menor, menor o igual
>, >= Mayor, mayor o igual
NOT, AND, NAND, OR, NOR Lógicos

El resultado de usar las operaciones generalmente se almacena, en VHDL esta acción se realiza como sigue:

:= Cuando se almacena en una variable (bloque *process*).
<= Cuando se almacena en una señal.

Sabiendo esto, se procede a definir subrutinas o partes de código que son usadas por los módulos del programa.

- Secuencia de asignación de valores a señales

Con frecuencia se hace necesario generar secuencias en las cuales se asignen diferentes valores a una señal, por ejemplo, el dato que se envía al conversor digital-análogo de modo serial o la secuencia de datos que se envía a la pantalla LCD para visualizar determinado mensaje.

Como ya se mencionó, el bloque *process* se utiliza para realizar secuencias, sin embargo, no es sencillamente asignar los valores a la señal uno tras otro, porque recordemos que este bloque solo modifica el valor de la señal al final del mismo, por lo tanto la señal solo sufrirá un cambio en su valor.

Entonces, la manera de asignar estos diferentes valores es recurriendo a la sentencia *case* cuyo funcionamiento es igual al *switch* de C, la variable por la que se rige el funcionamiento de esta sentencia será un contador de pasos de la secuencia, que ira aumentando desde 0, entonces en el paso 0 se realiza la primera asignación de la señal y el contador aumenta a 1.

Figura 41. Ejemplo secuencia asignación de señales

```
process (Senal)
   variable n : integer range 0 to 3:=0;
begin
   case vm is
      when 0=> Senal<='0';
               n:=1;
      when 1=> Senal<='1';
               n:=2;
      when 2=> Senal<='0';
               n:=3;
      when others=> null;
   end case;
end process Velocidad;
```

Autor.

Por supuesto, la señal debe hacer parte de la lista sensible del bloque *process*, con esto, al verse modificado el valor de la señal, el bloque se activará de nuevo, pero en este caso la sentencia case no irá al paso 0 sino al paso 1, en el cual se asigna el segundo valor a la señal y el contador aumenta a 2. De este modo se continúa hasta asignar a la señal todos los valores requeridos. Cuando se asigna el último valor a la señal el contador nuevamente aumenta, entonces en la siguiente activación del bloque la sentencia *case* llegará a un paso indicado como otros (*others*), con esto no se vuelve a modificar la señal. La Figura 41 muestra un ejemplo de la aplicación de esta subrutina.

- Retardo

La subrutina anterior funciona, pero presenta un inconveniente, el cambio en el valor de la señal se realiza inmediatamente, es decir, no hay un intervalo de tiempo entre cada cambio del valor de la señal. Los cambios de señal se deben presentar con un retraso entre cambio y cambio, para lo cual se propone el código ilustrado en la Figura 42, en el cual se modificó el bloque del ejemplo anterior y se agregó un nuevo bloque.

El bloque se modifica, agregando más pasos a la sentencia *case*. En los pasos en los cuales se asigna un valor a la señal, también se asigna el valor verdadero (*true*) a una nueva señal denominada como i_ret, con esto se iniciará el retardo que se encuentra en el otro bloque, y se aumenta el contador para cambiar de paso. En el siguiente paso, se espera a que el retardo finalice, condición que se dará en el momento en que la señal f_ret tenga valor *true*. Cuando esto suceda se asigna valor falso (*false*) a i_ret para que no continúe el retardo y se aumenta el contador para que en la siguiente activación del bloque se asigne otro valor a la señal y así continuar. Finalmente, la lista sensible estará compuesta por dos señales, la que se está asignando diferentes valores y la que indica que el retardo ha terminado.

Figura 42. Ejemplo de subrutina de retardo

```
process (Senal, f_ret)
   variable n : integer range 0 to 4:=0;
begin
   case vm is
      when 0=> Senal<='0';
               n:=1;
               i_ret<=true;
      when 1=> if f_ret then
                  i_ret<=false;
                  n:=2
               end if;
      when 2=> Senal<='1';
               n:=3;
               i_ret<=true;
      when 3=> if f_ret then
                  i_ret<=false;
                  n:=4
               end if;
      when others=> null;
   end case;
end process;
-------------------------------------------------
process (CLOCK)
   variable conteo : integer range 0 to 125000:=0;
begin
   if CLOCK='1' and CLOCK'event then
      f_ret<=false;
      if i_ret=true then
         conteo:=conteo+1;
         if conteo=125000 then
            f_ret<=true;
         end if;
      else
         conteo:=0;
      end if;
   end if;
end process;
```

Autor.

El nuevo bloque se encarga del retardo, este funciona de forma síncrona con la señal de reloj que proviene el cristal de la tarjeta de desarrollo, esto se logra ubicando dicha señal (CLOCK) en la lista sensible del bloque y poniendo como primera condición, que las demás instrucciones se ejecuten cuando el reloj presente flanco ascendente. Ahora las instrucciones presentan una condición, el

temporizador funciona cuando la señal de activación i_ret lo indique (evento que ocurre en el bloque anterior). Si la condición se cumple la variable conteo aumenta de uno en uno, hasta alcanzar el valor umbral, momento en el cual finaliza el retardo, indicando esto al asignar el valor *true* a la señal f_ret. Cuando la condición no se cumpla (retardo desactivado) la variable conteo se reinicia en 0 y la señal f_ret se hace igual a *false*.

El valor de umbral marca la duración del retardo. Dado que el contador aumenta en cada ciclo del reloj, cuando esto sucede, ha transcurrió un tiempo igual al periodo del reloj. El cristal de la tarjeta de desarrollo es de 5MHz, con lo cual el periodo es de 20ns, por lo tanto el valor de umbral en función de la duración del retardo viene dado por:

$$Umbral = \left| \frac{T(ns)}{20ns} \right| \qquad (2.22)$$

Donde *T* representa la duración del retardo expresada en nanosegundos. Se representa con valor absoluto, porque el contador funciona con valores enteros.

En el ejemplo ilustrado se tiene un valor de umbral de 1’250.000, lo que nos indica que el retardo es de 25ms.

2.4.3 Memoria de trayectoria recorrida

El sistema de referencia para el posicionamiento del robot es cartesiano *XY*. Ahora, una de las premisas del funcionamiento del robot indica que este debe ir memorizando la trayectoria que recorre. Pues bien la forma de realizar esto es discretizando la medida de la posición de la siguiente manera:

- Posición inicial

Tomando una vista superior del robot, el eje *y* va de sur a norte (atrás hacia delante) y el eje *x* de occidente a oriente (izquierda a derecha). Se parte del hecho que al encender el robot, el centro de giro se encuentra ubicado en el centro del sistema de referencia, esto es *x*=0 y *y*=0. El vector de orientación apunta desde el centro de giro hacia la parte delantera-centro del robot, entonces, la condición inicial indica que esta dirección coincide con el eje y, por lo tanto el ángulo de orientación es de 90º con respecto al eje *x*.

- Avance recto

Realizando este movimiento es la única forma en que el robot modifica su posición *XY*. Cada vez que el robot se mueva en línea recta y según la ecuación 2.15, avanza $1.68x10^{-2}$cm por cada pulso del encoder, en la dirección del vector de orientación. Entonces se realiza el conteo de pulsos y se calcula la distancia recorrida.

- Giro sobre el eje

Realizando este movimiento el robot modifica el ángulo de orientación, si el giro es de izquierda a derecha, el valor del ángulo disminuye, de lo contrario aumenta. La variación del ángulo se rige mediante la ecuación 2.3.

$$\theta = \frac{2S}{b} = \frac{2S}{22cm} \qquad (2.23)$$

Donde *S* representa la distancia recorrida por cada rueda y se mide del mismo modo que en el avance recto, según la relación y el número de pulsos del encoder.

- Posiciones de memorización

Hay dos momentos en los cuales el robot memoriza su posición, el primero es cada que avanza 5cm en línea recta, esto según la ecuación 2.15 es equivalente a 297 pulsos del encoder, el otro cuando llega a un punto de giro.

El dato que se guarda en memoria es el nuevo valor de las coordenadas *X* y *Y*, que se calcula como sigue:

$$\begin{aligned} X &= X + \Delta X \\ Y &= Y + \Delta Y \end{aligned} \tag{2.24}$$

ΔX y ΔY se calcula según la siguiente relación:

$$\begin{aligned} \Delta X &= \Delta S * \cos\theta \\ \Delta Y &= \Delta S * \sin\theta \end{aligned} \tag{2.25}$$

ΔS se obtiene según el momento en que se memoriza el punto, cuando es por avance en línea recta, su valor es de 5cm en. Cuando es por punto de giro es igual a la cantidad de pulsos del encoder medidos desde el último punto de memoria y convertidos a distancia en cm.

2.4.4 Sistema de búsqueda de trayectoria

Este es el punto más importante en cuanto al funcionamiento del robot, se trata del mecanismo que usa el FPGA para encontrar el camino que lleve al robot a su destino. Las partes que lo componen se explican a continuación.

- Posiciones de referencia alrededor del robot

Se sabe que el motor paso a paso cubre un ángulo de 180º centrado frente al robot, y este ángulo es representado por 48 pasos. Ahora bien, independiente del

hecho de que el sensor no cubre la circunferencia alrededor del robot, se define un sistema de referencia de 360º, representado por los 96 pasos que daría el motor paso a paso para completar un giro. Con base en lo anterior se crea un vector de 96 posiciones, donde cada posición representa un ángulo de dirección respecto al sistema cartesiano de posición del robot. Esta referencia es el pilar del funcionamiento del sistema de búsqueda.

- Ubicación de obstáculos

El vector presenta datos de tipo entero, aunque solo maneja tres valores (0, 1 o 2), y estos valores indican la presencia o no de obstáculos en el ángulo de dirección que representa cada posición del vector. El valor 0 indica que no hay obstáculo, el valor 1 que si hay obstáculo, y el valor 2 indica que se desconoce la presencia de obstáculos, esto es la parte trasera del robot, la cual el sensor no revisa.

- Ubicación del punto de llegada

Paralelo a lo anterior se ha calcular el ángulo de dirección del punto de destino. Ahora para determinar este ángulo basta realizar un sencillo cálculo geométrico teniendo las coordenadas *X* y *Y*, tanto actuales, como del punto de llegada (suministradas por el usuario al inicio del funcionamiento). Las coordenadas actuales, se refrescan cada vez se realiza memoria de trayectoria.

- Destino ubicado en línea recta

Esta es la primera y la más simple premisa de la trayectoria que debe seguir el robot, cuando en la dirección en la que se encuentra el punto de llegada no se encuentra ningún obstáculo, el robot se ha de mover siguiendo la línea recta que lleva a este punto.

Para detectar esta condición, se busca la posición del vector de ubicación de llegada en la cual esta el 1 (dirección del punto de llegada), teniendo esta posición se compara con la misma posición en el vector de obstáculos, si presenta 0 (no hay obstáculo), el robot se ubica hacia esta posición y avanza.

La detección de esta condición se realiza cada vez que el robot memoriza la posición y no esta haciendo uso del beneficio de dirigirse en línea recta hacia el obstáculo.

- Búsqueda de caminos alternos

Cuando no se presenta la condición anterior se recurre a la búsqueda o elección de caminos, para esto se incorpora una neurona artificial según el modelo de la neurona hebbiana visto en la sección 1.7.2, según el cual se calcula la entrada ponderada I, si esta es mayor a cero, el robot decidirá tomar camino bordeando el obstáculo por la izquierda, y si es menor que cero, lo hará bordeando por la derecha.

$$I = B + W.E \tag{2.26}$$

E, representa el vector de las entradas de la neurona, que en este caso son 4, la ecuación 2.27 muestra la manera de calcular el valor de cada una de estas entradas, y su significado es el siguiente:

- *E(1)* es una sumatoria que indica el nivel de obstáculos presentes a la derecha de la dirección del punto destino, calculado con base en los datos que posea el vector *POS* en las 15 posiciones inmediatamente a la derecha de la posición ubicada en la dirección del punto destino. Estos datos se guardan en un nuevo vector, *OBSD*, donde el primer dato representa la posición más cercana al punto destino y el dato 15, la más alejada.

Podemos ver como el dato 1 aporta al resultado en mayor proporción que el dato 15, ya que el dato 1 es el mas cercano al punto destino.

- *E(2)* es un factor que indica el nivel de éxito alcanzado al bordear el obstáculo por la derecha, donde nivel de éxito se refiere a estar más cerca o más lejos del punto destino. n_d es un contador que aumenta cada vez que el robot, bordeando el obstáculo por la derecha, memoriza una nueva posición *XY*; *f*(*i*) tendrá valor 0 si en la posición actual el robot se alejo del punto destino y 1 en caso contrario.

- *E(3)* es similar a *E(1)*, pero indica el nivel de obstáculos a la izquierda. Los 15 datos almacenados en *OBSI*.

- *E(4)* indica lo mismo que *E(2)*, pero esta vez el contador es n_i y, *f*(*i*) indica lo mismo, pero esta vez con referencia a bordear el obstáculo por la izquierda.

$$
\begin{aligned}
E(1) &= \sum_{i=1}^{15} \frac{OBSD(i)}{i} \\
E(2) &= \frac{1}{n_d} * \sum_{i=1}^{n_d} f(i) \\
E(3) &= \sum_{i=1}^{15} \frac{OBSI(i)}{i} \\
E(4) &= \frac{1}{n_I} * \sum_{i=1}^{n_i} f(i)
\end{aligned}
\qquad (2.27)
$$

W, es el vector de ponderaciones o pesos, sus datos sirven para modificar los valores que circulan por la neurona. *B* es una constante. Los datos de *W* y *B* se modifican en el proceso de entrenamiento de la neurona, y sus valores iniciales son: 0 para los datos de *W*, y 1 para *B*.

El proceso de entrenamiento de la neurona implementada en el FPGA, se hizo poniendo el robot en funcionamiento y mediante inspección visual, cada vez que el robot encontraba un obstáculo, se detenía y por medio de los switch de la tarjeta de desarrollo se indicaba la salida deseada, es decir, si debía bordear el obstáculo por la derecha o la izquierda. Hecho esto se calculaba los nuevos valores para *W* y *B*.

- Condición de punto de llegada inaccesible

Existen diferentes ocasiones en las cuales el punto de llegada puede ser inaccesible. Después de un análisis visual se determinó que si el robot pasa por tercera vez por un mismo punto está andando de un lado para otro sin encontrar camino al punto destino. Entonces si el robot pasa más de dos veces por un mismo punto se detiene la búsqueda.

Ahora, como se establece volver a pasar por un punto, lo primero es recordar que el robot tiene memorizada la trayectoria que ha recorrido y los puntos en los cuales ha girado. Entonces, cada vez que se memoriza un nuevo punto, este se compara con todos los puntos guardados hasta dos giros antes de la posición actual para ver si son puntos similares, la condición de dos giros antes es para evitar la comparación con los puntos de las últimas dos trayectorias rectas descritas por el robot. Para la comparación se establece que dos puntos son similares cuando al restar entre si los valores de X y de Y el resultado está en un rango de -2.5cm a +2.5cm.

- Punto destino alcanzado

El método de comparación descrito en el párrafo anterior es utilizado también para determinar si un punto memorizado corresponde al punto destino, y así terminar la búsqueda.

2.5 INTERFAZ DE USUARIO

En términos generales le manejo del robot es sencillo, primero se muestran los dispositivos de interacción del usuario y luego la secuencia normal de funcionamiento el robot.

2.5.1 Dispositivos

Los dispositivos son básicamente dos: botones de configuración y puesta en marcha, y pantalla LCD para visualizar variables del robot. Estos se encuentran ubicados en la tarjeta de desarrollo y se pueden ver en la Figura 49.

- Botones

Son 5, uno central y los otros 4 ubicados en los puntos cardinales, su funcionamiento es sencillo, normalmente envían 0 lógico al FPGA y cuando se oprimen, envían un 1 lógico.

El botón central puede girar, esta característica se usa al momento de seleccionar el punto de llegada del robot.

- Pantalla LCD

La pantalla es de 2 líneas, cada una con posibilidad de mostrar 16 caracteres. Se usa para mostrar mensajes de la operación del circuito, en otro punto de la operación muestra las coordenadas de posición actual del robot y el restante para llegar al sitio destino, esta información se actualiza cada vez que el robot memoriza una nueva posición.

2.5.2 Secuencia de operación del robot

La Figura 43 ilustra los mensajes que se visualizan en la pantalla LCD.

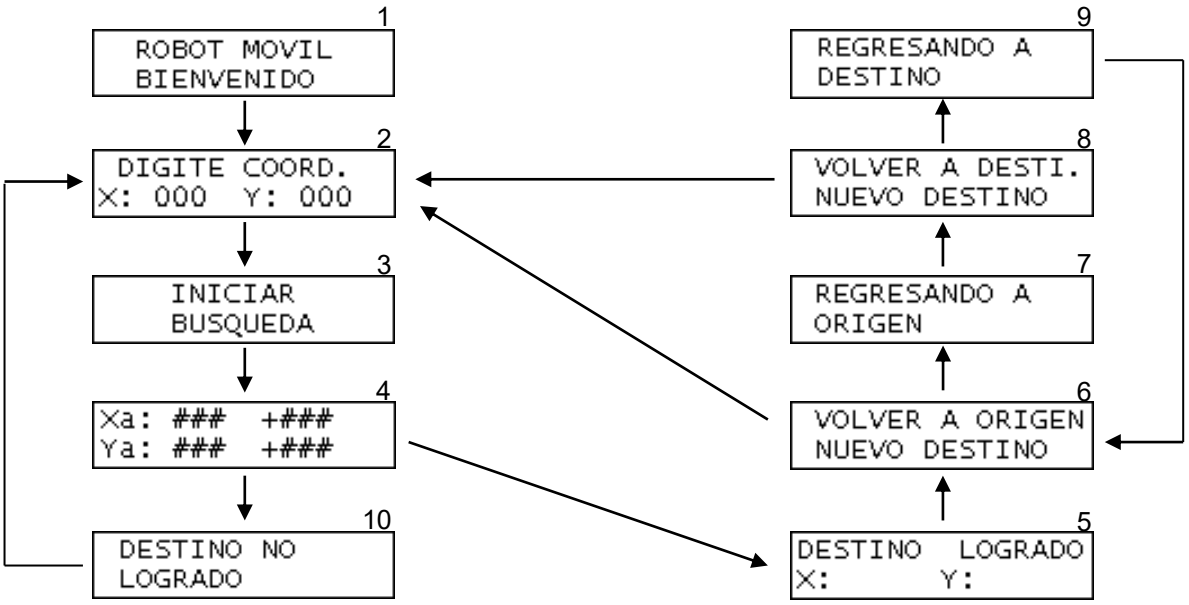
La secuencia de operación del robot tan pronto como se enciende es la siguiente:

1. Mueve el motor paso a paso de izquierda a derecha para referenciar su posición.
2. A la vez que realiza lo anterior en la pantalla se observa el mensaje de bienvenida. (Mensaje 1)
3. Presionando el botón central se visualiza el Mensaje 2 solicitando las coordenadas del punto destino.
4. Para indicar los valores de coordenadas se utilizan los botones arriba, abajo, izquierda y derecha.
5. Presionando el botón central se guardan las coordenadas del punto destino y se visualiza el mensaje de confirmación de inicio. (Mensaje 3)
6. Presionando nuevamente el botón central se inicia la búsqueda.
7. Durante la búsqueda se visualiza el Mensaje 4, que visualiza la posición del robot y la distancia que falta al punto destino. Estos valores se actualizan cada vez que el robot memoriza su posición.
8. Cuando el robot alcanza el punto destino se visualiza el Mensaje 5.
9. Presionando el botón central se visualiza el Mensaje 6.
10. Utilizando los botones arriba, abajo y centro; se escoge entre las opciones de volver al punto de inicio o buscar nuevo destino
11. Si se escoge NUEVO DESTINO se visualiza el Mensaje 2 nuevamente y la secuencia vuelve al paso 4.
12. Si se escoge VOLVER A ORIGEN, el robot ejecuta esta acción y en la pantalla se visualiza el Mensaje 7.
13. Cuando el robot retorna el punto origen, se visualiza el Mensaje 8 en pantalla.

14. Utilizando los botones arriba, abajo y centro; se escoge entre las opciones de regresar al punto destino o buscar nuevo destino.
15. Si se escoge NUEVO DESTINO se visualiza el Mensaje 2 nuevamente y la secuencia vuelve al paso 4
16. Si se escoge VOLVER A DESTI., el robot ejecuta esta opción y en la pantalla se visualiza el Mensaje 9.
17. Cuando el robot alcanza el punto destino nuevamente se visualiza el Mensaje 6 y la secuencia vuelve al paso 10.

En caso que el robot no pueda encontrar el punto destino en el paso 7, se informa esta condición visualizando el Mensaje 10. Luego de esto, y presionado el botón centro, se visualiza el Mensaje 2 y la secuencia vuelve al paso 4.

Figura 43. Mensajes en LCD



Autor.

3 CONSTRUCCIÓN

3.1 CHASIS

La construcción inició con los acoples (chumaceras) para las ruedas, teniendo en cuenta que el diámetro de estas es de 6.5cm, lógicamente los acoples deben tener una altura inferior, es por esto que la altura es de 4cm. Dentro del acople va el rodamiento y por medio de este pasa el buje que entra a presión sobre la rueda, la otra punta del buje lleva un orificio concéntrico, en el cual se ajusta el eje del motor.

Terminado el acople, comienza la construcción del chasis, en principio este era una lámina de acrílico de 20cm de frente por 25cm de fondo, que tenía un solo nivel y cuyo frente describía media circunferencia, debajo de esta, tenía pegada unas bases de forma perpendicular, a estas bases se ajustaban los motores por medio de tornillos. Igualmente, a la lámina se atornillaban los acoples de los motores mencionados en el párrafo anterior. Este conjunto lucía tal y como se observa en la siguiente figura.

Figura 44. Primera construcción del chasis del robot

Autor.

Al momento de acoplar la rueda “loca” o rueda libre se detectó, tal y como se mencionó en el diseño, que el chasis de un solo nivel era muy alto para acoplar esta rueda, además existía una perdida de espacio en la parte trasera del robot, por debajo del chasis y justo alrededor de la rueda libre.

Por esta razón el diseño y la construcción del chasis se modificó a dos niveles, es decir, la lámina se divide en dos partes, la parte que representa la zona de atrás del robot se ubica más cerca al piso y esta se una a la otra parte (delantera), que lleva acoplado los motores, por medio de una lámina perpendicular a ambas y de la misma anchura. Esta modificación no afecta la disposición de las ruedas y los motores, pero si representa una mejor disposición estética para la rueda libre y la recuperación de el espacio perdido. La Figura 45 muestra el chasis con la nueva modificación.

Figura 45. Chasis de dos niveles

Autor.

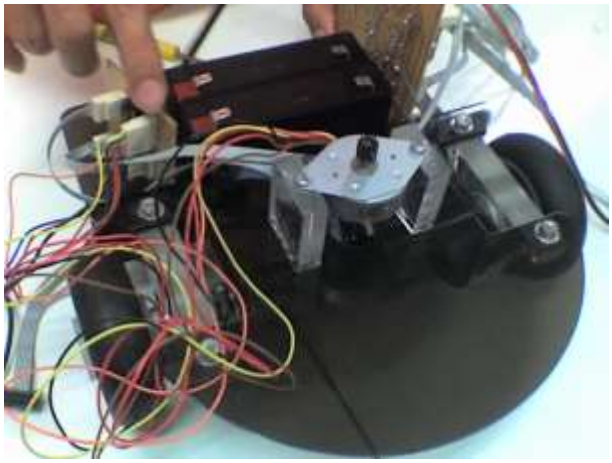
En el nuevo nivel creado, se instala el soporte para las baterías, unas pequeñas láminas de acrílico que lo sujetan, el cual queda centrado con respeto a los lados del robot y pegado a la lámina que une los dos niveles del chasis. A lado y lado de este soporte se ubican las bases en acrílico para los circuitos impresos (que se

describen en la siguiente sección), estas bases van paralelas a cada borde lateral de la lámina, de esta forma los circuitos dan la cara a los laterales del robot, esta disposición con el objeto de buscar la simetría, tanto por razones estéticas como por equilibrio de las cargas. Sobre estas bases se instala un soporte para la tarjeta de desarrollo que contiene el FPGA.

Ahora bien, en la parte delantera del robot, sobre el nivel más alto del chasis, es decir, por encima de los acoples y los motores de avance del robot, se instalan dos soportes para el motor paso a paso, de modo que el eje de este quede ubicado perpendicular al chasis y justo sobre el punto de giro del robot, a este eje se acopla el sensor ultrasónico junto con su soporte, de este modo el sensor describirá el movimiento ilustrado en la Figura 37.

Debajo de este nivel y frente a los motores se acopla el módulo del sensor ultrasónico, este y todos los acoples de dispositivos se hace por medio del conjunto tornillo-arandela-guasa-tuerca, que garantiza buena sujeción.

Figura 46. Construcción completa del chasis



Autor.

La construcción definitiva del chasis junto con los soportes y motores instalados se presenta en la Figura 46, cabe resaltar que aquí no se observan aún las tarjetas impresas, que se explicarán en la siguiente sección.

3.2 TARJETAS IMPRESAS

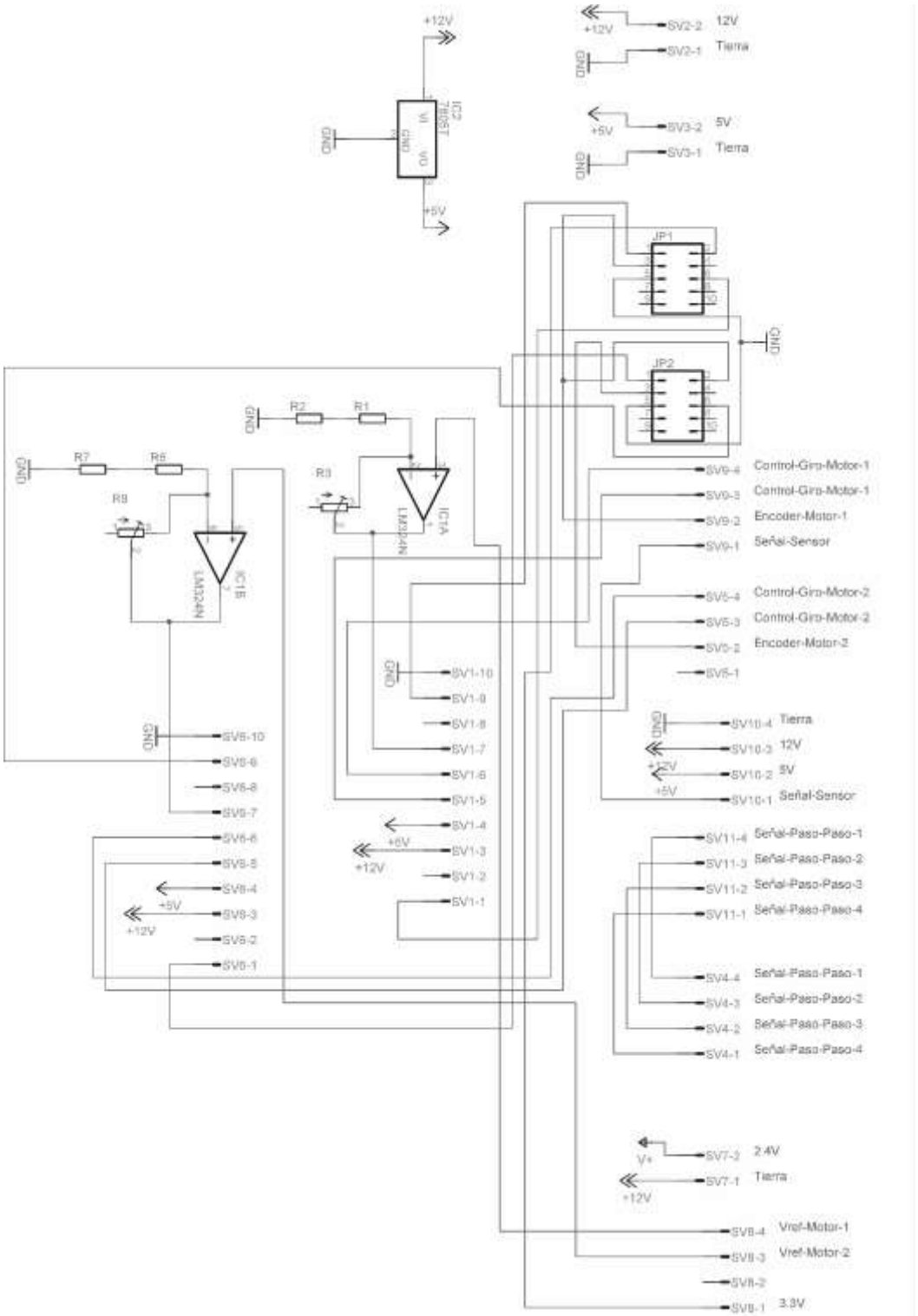
3.2.1 Tarjeta principal

A esta tarjeta llega la alimentación de 12VDC desde las baterías, contiene un regulador de 5V para la alimentación de la tarjeta de desarrollo que contiene el FPGA. En el robot se encuentra ubicada al lado izquierdo.

Del mismo modo esta tarjeta contiene:

- Conector para alimentación de la tarjeta de desarrollo.
- Los dos driver TA7291 con sus respectivas conexiones.
- Conectores de los dos motores que se unen con los driver, envían la alimentación para los encoder y reciben la señal de los mismos.
- Conector a tarjeta secundaria con alimentación a 12V y 5V, y recepción de la señal del sensor.
- Conector a FPGA para recibir las dos señales de giro para el motor de la izquierda, y enviar la señal del encoder de este motor y la señal del sensor.
- Conector a FPGA para recibir las dos señales de giro para el motor de la derecha y enviar la señal de encoder de este motor.
- Conector a tarjeta de desarrollo para recibir alimentación de 3.3V para los encoder y señal de conversor digital-análogo.
- Los dos circuitos de amplificación de la señal recibida del conversor digital-análogo de la tarjeta de desarrollo para el control de la velocidad de los motores, estos circuitos conectados al driver TA7291.

Figura 47. Diagrama esquemático de la tarjeta principal



Autor.

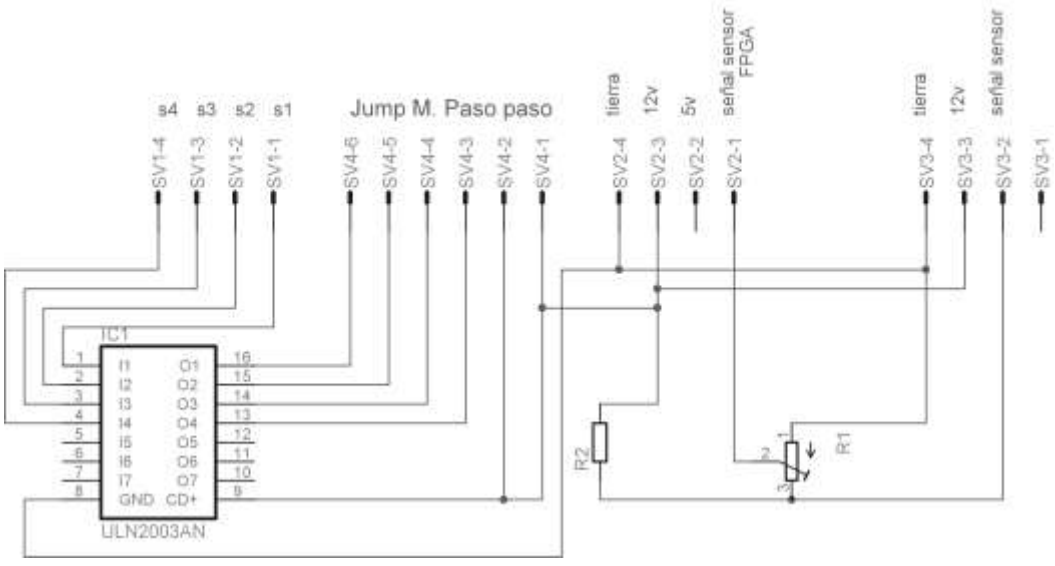
3.2.2 Tarjeta secundaria

Esta tarjeta se encuentra ubicada al lado derecho del robot y contiene los siguientes elementos:

- Conector a tarjeta principal para recibir alimentación a 12V y 5V, y enviar la señal de sensor a referencia de 3.3V.
- Conector a FPGA para recibir secuencia de operación de motor paso a paso.
- Driver ULN2003 para motor paso a paso.
- Conector a motor paso a paso que le envía 12V y lo une al driver.
- Conector a modulo de sensor ultrasónico que envía alimentación a 12V y recibe las señal de detección de obstáculos.
- Circuito divisor de voltaje para la señal del sensor de referencia 12V a 3.3V.

La siguiente figura ilustra el diagrama esquemático de la tarjeta secundaria.

Figura 48. Diagrama esquemático de la tarjeta secundaria.



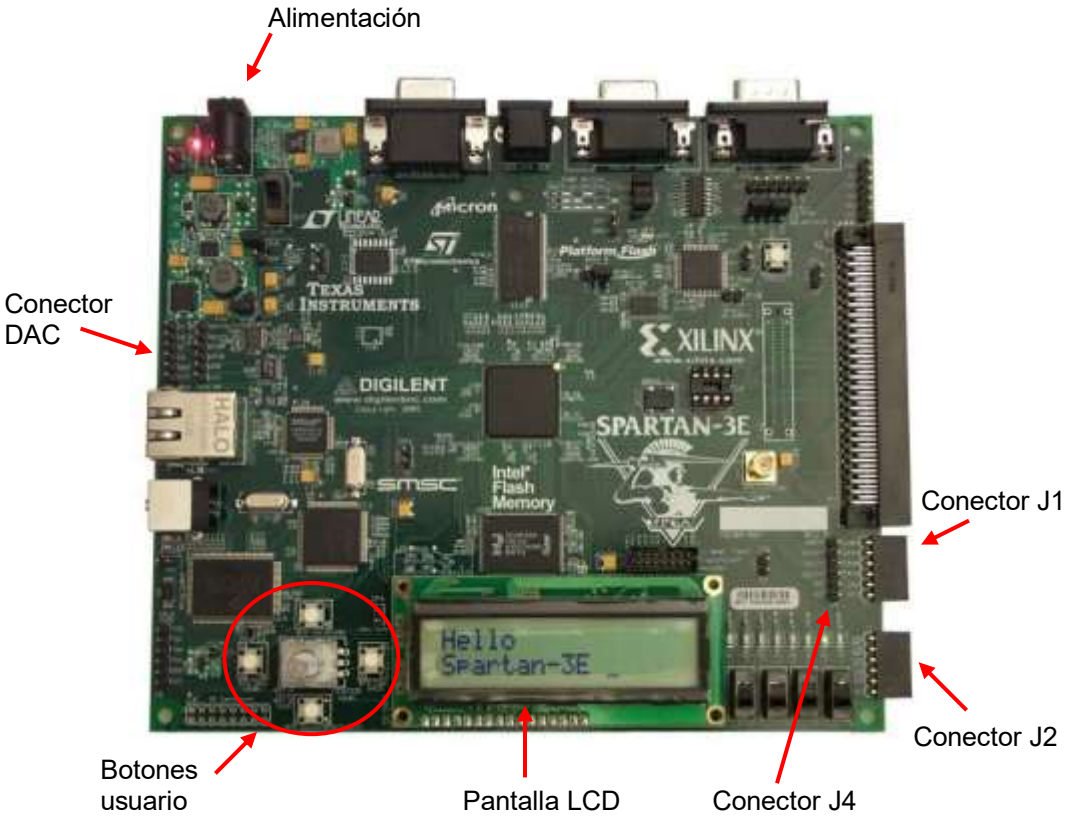
Autor.

3.3 TARJETA DE DESARROLLO

La Figura 49 muestra la tarjeta de desarrollo resaltando los dispositivos e interfaces que utiliza el robot, a continuación se describe la función de cada uno.

- Alimentación: Como su nombre lo indica, por medio de este conector se alimenta la tarjeta de desarrollo (5V), a través de la tarjeta impresa principal.
- Conector DAC: Dos pines para la salida de la señal de voltaje (0 a 2.5V) que indica la velocidad de los motores. Pin de 3.3V para alimentación de los encoder.
- Botones funcionamiento: Por medio de estos botones se opera el robot.

Figura 49. Conexiones a la tarjeta de desarrollo



Autor.

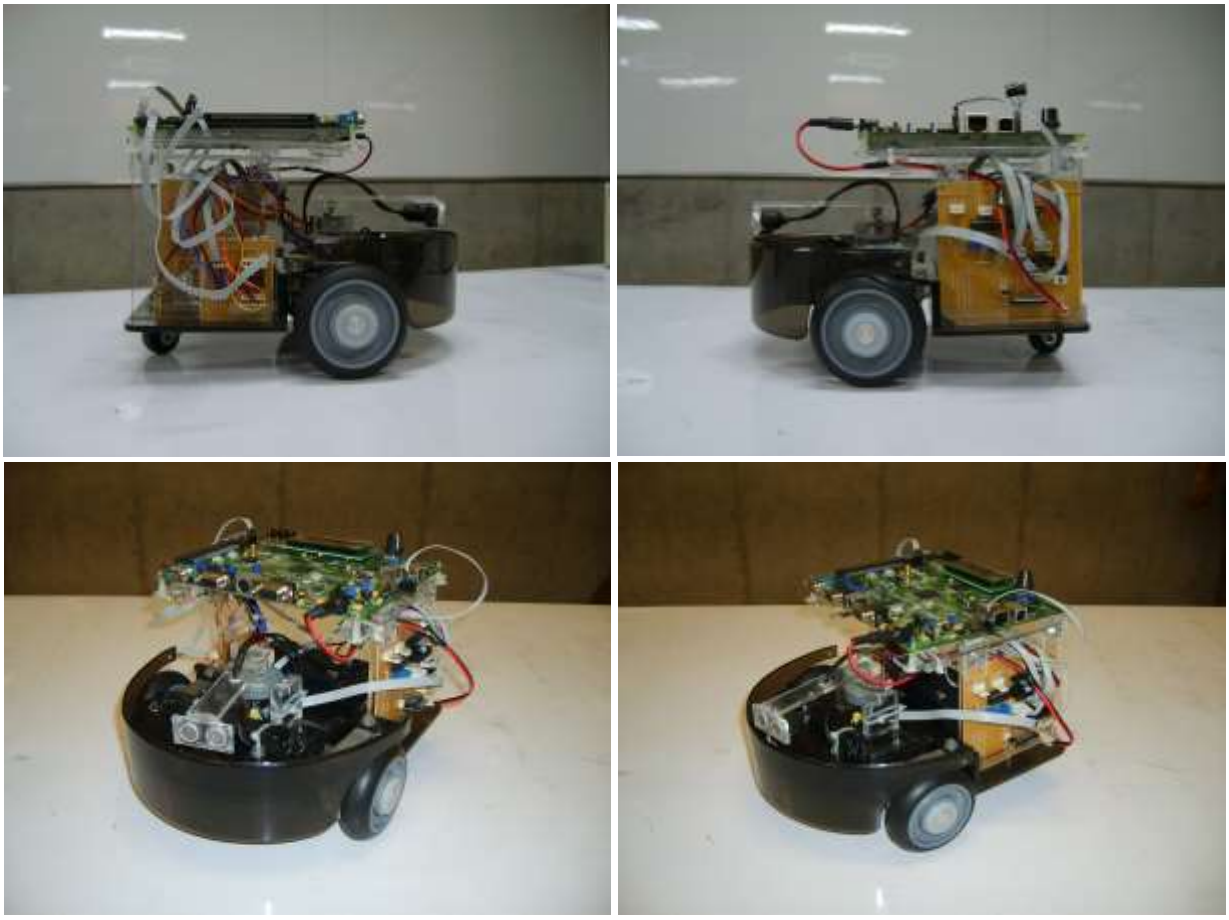
- Pantalla LCD: Muestra el estado del robot.
- Conector J1: Envía dos señales de giro para el motor de la izquierda y recibe señal del encoder del mismo motor, además de la señal del módulo de sensor ultrasónico acoplada a referencia 3.3V.
- Conector J2: Envía dos señales de giro para el motor de la derecha y recibe señal del encoder del mismo motor.
- Conector J4: Envía secuencia de operación de motor paso a paso.

3.4 ENSAMBLE FINAL

La Figura 50 muestra cuatro fotografías en las que se observa el ensamble entre de chasis final, los acoples y motores DC, el soporte y el motor paso a paso con el acople al sensor ultrasónico (parlante y micrófono), el módulo del sensor, las tarjetas impresas principal y secundaria, la tarjeta de desarrollo, las baterías, y el cableado de conexión entre las tarjetas y los dispositivos.

La Figura 51 muestra armado del robot completo, que incluye además de lo anterior, la carcasa de protección.

Figura 50. Ensamble del robot



Autor.

Figura 51. Armado completo del robot

Autor.

INTRODUCCION

En robótica móvil existe gran cantidad de trabajos enfocados a diferentes tareas, entre algunas de estas tenemos, la navegación entre laberintos, la recolección de objetos o el reconocimiento de entornos. En este caso la tarea consiste en desplazarse de un punto a otro en un plano, evadiendo los obstáculos, en caso de que estos existan.

Dos características diferencian este trabajo de otros, una de ellas, la ausencia de algún mapa o guía para lograr la trayectoria, ya que solo se tiene los dos puntos identificados por medio de coordenadas cartesianas. La otra característica es la forma de controlar el robot, utilizando un FPGA o arreglo de compuertas lógicas de la marca Xilinx, y perteneciente a la familia Spartan 3E.

Este FPGA viene montado en una tarjeta de desarrollo en la cual se conecta a diferentes interfaces y dispositivos, algunos de estos utilizados para el funcionamiento del robot, como lo son: botones, conectores, pantalla LCD, conversor digital-análogo, entre otros. El FPGA es programado utilizando el lenguaje de descripción hardware VHDL, mediante el cual se especifican las funciones que deben realizar los componentes del robot para luego ser sintetizadas en el formato que entiende el FPGA utilizando la herramienta de síntesis proveída con la tarjeta de desarrollo.

La implementación de FPGA ofrece la gran ventaja del procesamiento en paralelo. De esta forma se reciben las señales de los sensores e inmediatamente se efectúan las acciones de control sin necesidad de esperar a que terminen otras secuencias.

El desarrollo de este libro esta dividido en tres capítulos, en el primero se presenta el marco teórico, en el cual se compila de manera resumida los conceptos aplicados en el diseño del robot.

El segundo capítulo refiere al diseño del robot, empezando por una ilustración de la secuencia de diseño y continuando con la explicación de cada uno de los pasos de esta secuencia, que incluye: la selección el tipo de robot, diseño mecánico, interfaces con sensores y actuadores, programación del robot para terminar con la interfaz usuario.

En el tercer capítulo se presenta la construcción y acople de los elementos mecánicos y electrónicos del robot.

Finalmente, se presentan las conclusiones del desarrollo del presente proyecto.

PLANTEMIENTO DEL PROBLEMA

La situación que se desea afrontar se sustenta básicamente en la necesidad de implementar las nuevas tecnologías en lo que a dispositivos programables se refiere, en donde existe una variada oferta que va más allá de los microcontroladores y ofrece grandes ventajas. Bien es sabido que la tecnología avanza a pasos agigantados y en el desarrollo como ingenieros y profesionales es importante tener la capacidad de aplicar los avances tecnológicos de una forma adecuada y coherente, pues bien, con este proyecto se pretende dar el primer paso en la universidad e incluso en la región como un proyecto de investigación en la aplicación de dispositivos lógicos programables (PLD's) que junto con lenguajes de programación y descripción de hardware como el VHDL son las herramientas mas utilizadas a nivel industrial y en el proceso de diseño de sistemas electrónicos complejos. Un aspecto importante que vale la pena destacar radica en el hecho de que en la facultad ya se encuentran aprobados recursos para proyectos de investigación relacionados a la implementación de estos dispositivos.

El concepto de redes neuronales es otro de los aspectos a tratar en este proyecto, esencialmente porque es otro punto que hasta ahora no se ha trabajado, y de un modo u otro hacia allá se dirige la computación, la programación y los modelos de inteligencia artificial (IA), en este modelo también conocido como modelo de computación conexionista se basará la concepción del control y la toma de decisiones de la "inteligencia" del robot en su funcionamiento.

El problema a solucionar consiste básicamente en concebir una forma muy sencilla de inteligencia artificial utilizando el concepto de redes neuronales e implementando dispositivos lógicos programables y lenguaje VHDL, mediante el cual un pequeño robot móvil tenga la capacidad de identificar y memorizar una forma de ir de un punto a otro en un plano, es decir, una trayectoria. Posterior al

aprendizaje de esta trayectoria, el robot deberá estar en la capacidad de recorrerla de forma eficiente y continua hasta que se le indique el aprendizaje de una nueva trayectoria, punto el cual borrara la trayectoria anterior. Para el funcionamiento del prototipo se dispone de dos etapas, una primera de aprendizaje, en la que se le aplican como entradas las coordenadas de los dos puntos, aquí entonces se encargará de reconocer la trayectoria; en una segunda etapa o modo de operación el robot se desplazara por la trayectoria que se encuentre almacenada en su memoria.

La importancia de este proyecto radica, como ya se ha explicado anteriormente, en el hecho de dar un primer paso a la implementación de nuevas tecnologías y conceptos basándonos en la investigación y aprovechando los recursos que ofrece la universidad y la facultad en este aspecto, con un proyecto de investigación ya aprobado.

OBJETIVOS

OBJETIVO GENERAL

Diseñar, construir e implementar un prototipo de robot móvil que funcione en dos etapas, en la primera que tenga la capacidad de identificar y aprender una trayectoria entre dos puntos dados en una superficie plana, para posteriormente en la segunda etapa de su funcionamiento recorrer la trayectoria aprendida almacenada en su memoria.

OBJETIVOS ESPECIFICOS

Realizar el diseño mecánico del robot móvil, seleccionando la configuración adecuada según los movimientos y trayectorias que deberá seguir el robot

Construir el robot y dimensionarlo para que pueda operar en superficies pequeñas, a la vez tenga el espacio suficiente para albergar los elementos mecánicos, además de los dispositivos y el montaje electrónico propios de su funcionamiento.

Seleccionar e implementar uno o varios sensores para detectar y esquivar los obstáculos que se encuentren entre los dos puntos a recorrer por parte del robot.

Implementar el lenguaje VHDL en dispositivos lógicos programables, hardware en el cual se lleve a cabo el proceso de control del robot, según las directrices establecidas para tal fin.

Investigar y aplicar el concepto de redes neuronales para la concepción de la estrategia de control y el aprendizaje de la trayectoria.

ANTECEDENTES

Este es un proyecto que pretendía aplicar nuevas tecnologías y conceptos, contó además del apoyo de los autores con el de la Universidad Autónoma de Bucaramanga y la Facultad de Ingenierías físico mecánicas, especialmente con el proyecto de investigación que le fue aprobado a la Ing. Nayibe Chio Cho en su momento, y con el cual se contó con recursos para implementar estas tecnologías.

En cuanto a los robots móviles, se tiene ya una experiencia en la facultad, en la cual se han desarrollado diferentes prototipos a nivel de proyectos de grado y proyectos integradores, de aquí se destaca la importancia de aprovechar la experiencia que se tiene en este sentido en cuanto al diseño y construcción del prototipo.

VHDL EN LA ACTUALIDAD

La actividad que se ha generado en torno a VHDL es muy intensa. En muchos países como España se han creado grupos de trabajo alrededor de dicho lenguaje y se realizan periódicamente conferencias, reuniones, etc., donde se presentan trabajos tanto en Estados Unidos (en el VIUF, VHDL International User's Forum) como en Europa (VHDL Forum for CAD in Europe), así como en el congreso EuroVHDL celebrado desde 1992.

El proceso de estandarización del VHDL no se detuvo con la primera versión del lenguaje (VHDL '87), sino que ha continuado con la nueva versión (VHDL '93) y constantes actualizaciones, mejoras y metodologías de uso. Entre estas adiciones 0 actualizaciones se encuentra una muy importante: la extensión analógica, que permite la utilización de un lenguaje único en todas las tareas de especificación, simulación y síntesis de sistemas electrónicos digitales, analógicos o mixtos.

CAMPO DE APLICACIÓN DE LA LÓGICA PROGRAMABLE

La lógica programable es una herramienta de diseño muy poderosa, que se aplica en el mundo industrial y en proyectos universitarios en todo el mundo. En la actualidad se usan desde los PLD más sencillos (como el GAL, PAL, PLA) como reemplazos de circuitos LSI y MSI, hasta los potentes CPLD y FPGA, que tienen aplicaciones en áreas como telecomunicaciones, computación, redes, medicina, procesamiento digital de señales, multiprocesamiento de datos, microondas, sistemas digitales, telefonía celular, filtros digitales programables, entre otros.

En general, los CPLD son recomendables en aplicaciones donde se requieren muchos ciclos de sumas de productos, ya que pueden introducirse en el dispositivo para ejecutarse al mismo tiempo, lo que conduce a pocos retrasos. En la actualidad, los CPLD son muy utilizados a nivel industrial, ya que resulta fácil convertir diseños compuestos por múltiples PLD sencillos en un circuito CPLD. Por otro lado, los FPGA son recomendables en aplicaciones secuenciales que no suponen grandes cantidades de términos producto. Por ejemplo, los FPGA desarrollados por la compañía A TMEL ofrecen alta velocidad en cómputo intensivo, aplicaciones en procesadores digitales de señales (DSP) y en otras fases del diseño lógico, debido a la gran cantidad de registros con los que cuentan sus dispositivos (de 1024 a 6400). Esto los hace ideales para su uso en dichas áreas.