DISEÑO DE UNA INTERFAZ PARA CONTROLAR EL MOVIMIENTO DE UN ROBOT (3-DOF) UTILIZANDO COMUNICACIONES INALÁMBRICAS

Trabajo de Grado # 1046

Laura Jimena Mosquera Arrieta

Paola Reinemer Valencia

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ DC

2011

DISEÑO DE UNA INTERFAZ PARA CONTROLAR EL MOVIMIENTO DE UN ROBOT (3-DOF) UTILIZANDO COMUNICACIONES INALÁMBRICAS

Trabajo de Grado # 1046

Laura Jimena Mosquera Arrieta

Paola Reinemer Valencia

Directores

Ing. Ana Milena López López

Ing. Diego Alejandro Patiño Guevara Ph.D.

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

BOGOTÁ DC

2011

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

RECTOR MAGNIFICO:	P. JOAQUÍN EMILIO SÁNCHEZ GARCÍA, S.J.
DECANO ACADÉMICO:	Ing. FRANCISCO JAVIER REBOLLEDO MUÑOZ
DECANO DEL MEDIO UNIVERSITARIO:	P. SERGIO BERNAL RESTREPO, S.J.
DIRECTOR DE CARRERA:	Ing. JUAN MANUEL CRUZ BOHORQUEZ M.Ed.
DIRECTORES DEL PROYECTO:	Ing. ANA MILENA LÓPEZ LÓPEZ
	Ing. DIEGO ALEJANDRO PATIÑO GUEVARA Ph.D

NOTA DE ADVERTENCIA

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque los trabajos no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ellos el anhelo de buscar la verdad y la justicia".

Artículo 23 de la Resolución No 13, del 6 de julio de 1946. Por la cual se reglamenta lo concerniente a Tesis y Exámenes de Grado de la Pontificia Universidad Javeriana.

AGRADECIMIENTOS

Este proyecto no se habría podido realizar sin la colaboración de muchas personas que nos han brindado su disposición y sus conocimientos. En primera instancia, agradecemos a nuestros dos directores de trabajo de grado la Ing. Ana Milena López López y el Ing. Diego Alejandro Patiño Guevara Ph.D, que nos han acompañado en todo momento durante el proceso.

Al personal del laboratorio de la Facultad de Ingeniería Electrónica, y del grupo de investigación SIRP, en especial al Ing. Carlos Alberto Parra Rodríguez Ph.D y a la Ing. Flor Ángela Bravo, por su disponibilidad y servicio.

Agradecemos a Dios y nuestras familias por el apoyo, la fortaleza y la confianza que nos brindaron durante la realización del proyecto.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ANEXOS

1. INTRODUCCIÓN

La automatización y el desarrollo tecnológico en la actualidad han incrementado el diseño y uso de robots para diferentes aplicaciones. Hoy en día el número de robots comerciales e industriales han aumentado en el mundo, mejoran las actividades realizadas por los humanos en trabajos con mayores riesgos profesionales como plantas de manufactura, transporte, medicina (cirugías), exploraciones terrestres y espaciales, minería, rescate y búsqueda de personas, limpieza de residuos tóxicos entre otros y constituyen una solución eficiente y prácticas ,debido que pueden ser programados para realizar la misma tarea varias veces con la misma precisión. Es por esto que son idóneos para aplicaciones en la industria [1].

La visión artificial en el campo de la robótica ha sido utilizada en exploraciones en el espacio, como las misiones a Marte y recolección de datos sobre su superficie las cuales se logran gracias a la visión con la que el robot cuenta por medio de cámaras incorporadas en el sistema de navegación. Además los datos recolectados por el robot pueden procesarse a distancia por medio de un computador y no en el robot mismo, como se hace en las exploraciones espaciales. Con la aplicación de la visión artificial en la robótica se puede conseguir información del entorno en forma diferente [2].

La visión en la robótica es muy importante ya que permite obtener información del espacio donde se desenvuelve el robot logrando la adaptación al medio en el que se encuentra. Cuando adicionalmente se transmite la imagen de forma inalámbrica directamente en el computador, se pueden diseñar diversas leyes de control visual para el robot y observar la evolución de las variables ya sea para seguir o identificar un objeto mediante el análisis del video. En el momento en el que se involucra más de un robot en el sistema, resulta más práctico tener la información centralizada en un computador, y no de forma local en cada uno de los robots. El control visual (visual servoing) es una técnica que últimamente ha sido muy utilizada en los robots ya que permite una relación sensorial en el cual no es necesario el contacto físico pues el uso de sensores para la navegación logran obtener la información del entorno donde se encuentra el robot. El control visual, es un tipo de control en el cual se adquieren datos desde un dispositivo (cámara) montado físicamente sobre un robot, que son utilizados para extraer información acerca de la posición del robot y así controlar los movimientos del mismo [3].

Con este proyecto se realizará el montaje de una plataforma para el control visual en la cual estudiantes de pregrado y postgrado puedan implementar leyes de control basadas en visión, además de poder

ejecutar otros proyectos que han sido desarrollados de manera simulada, pero no han sido implementados por no tener una plataforma como la diseñada en este proyecto.

Este proyecto hace parte del trabajo de investigación titulado "Diseño de leyes de control aplicando técnicas de Desigualdades Lineales Matriciales para un lazo de retroalimentación visual." realizado por la estudiante de maestría de Ingeniería Electrónica de la Pontificia Universidad Javeriana Ana Milena López López. El proyecto busca básicamente diseñar leyes de control robusto para realizar el control de la navegación de un robot basado en visión [4]. Estos proyectos a su vez hacen parte de un trabajo de investigación que se presentó a Colciencias y se titula "Control para sistemas distribuidos en una red de comunicaciones".

Además de ser complemento y parte de otro trabajo, este proyecto puede ser utilizado para beneficios educativos, ya que puede mostrar el control del robot desde el computador utilizando realimentación visual y ver los aspectos prácticos del uso de la visión y sus aplicaciones en la robótica.

El objetivo de este trabajo es construir una interfaz para controlar el movimiento de un robot (3-DOF) utilizando comunicaciones inalámbricas, lo cual se logrará montando una plataforma robótica con su respectivo sistema de comunicaciones. Se desea adicionalmente obtener el modelo del robot y validarlo mediante pruebas, y adquirir, ordenar y visualizar los datos por medio de Matlab®.

2. MARCO TEÓRICO

2.1 Robótica

2.1.1 Definición de Robots

Desde hace mucho tiempo el hombre ha tenido atracción hacia máquinas que imitan el movimiento del mismo y es por esto que se han venido incorporando varios dispositivos mecánicos con el paso del tiempo. [5]. Gracias a los avances tecnológicos que se viven en la actualidad, la robótica es cada vez más común en la vida cotidiana de los hombres. Por definición, la robótica es "*una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas capaces de desempeñar tareas realizadas por el ser humano o requieren del uso de inteligencia. Las ciencias y tecnologías de las que deriva podrían ser: el álgebra, los autómatas programables, las máquinas de estados, la mecánica o la informática*" [6].

Una definición oficial de un robot viene del Instituto de Robot América (RIA): *Un robot es un manipulador multifuncional reprogramable diseñado para mover materiales, piezas, herramientas o dispositivos especializados a través de variables y movimientos programados para la realización de una variedad de tareas* [1]. Existen varios tipos de robots: los móviles, estáticos, androides todos ellos con un objetivo común: mejorar y facilitar las tareas diarias del hombre.

2.1.2 Subsistemas de un Robot

Un robot puede considerarse como una agrupación de diferentes subsistemas con características particulares que aportan de manera fundamental al objetivo final de la construcción del mismo. Éstos se clasifican en: estructura, movimiento, alimentación, sensores y control y lógica como se describen a continuación.

Subsistema de estructura

La estructura de un robot es la responsable del soporte físico del mismo y es tan importante como análogamente es el esqueleto para el cuerpo humano. Al hacer la estructura de un robot se deben contemplar varios aspectos importantes: primero se debe calcular el centro de gravedad del robot al momento de distribuir el peso de los elementos que tendrá, ya que los componentes más pesados deben ser ubicados en el centro y los más livianos pueden ir más alejados; en segunda instancia, el polígono de soporte se debe imaginar antes de realizar la estructura para tener una idea clara de los puntos con los cuales la estructura se conecta con el piso, normalmente las ruedas; tercero, la estabilidad del robot se logra ubicando el centro de gravedad en el centro del polígono de soporte; cuarto, la robustez del robot, que consiste en que cada parte pequeña que conforma la estructura del robot debe estar bien unida para evitar separación y movimiento entres ellas; y finalmente, la vulnerabilidad de las piezas ya que hay partes

más frágiles que otras y al diseñar el robot se deben tratar de proteger para así evitar daños o maltratos en ellas que puedan afectar el desempeño [7].

Subsistema de movimiento

El movimiento del robot se lleva a cabo gracias a este subsistema el cual incluye los motores que generan el movimiento, y los engranajes y ruedas que transfieren y trasforman el movimiento en la forma deseada. Así como la estructura es el esqueleto, el subsistema de movimiento puede considerarse los músculos del robot. Los motores DC y servomotores son partes que logran transformar energía eléctrica en energía mecánica. Un motor DC se diferencia de un servomotor en que su movimiento rotacional es continuo y cubre 360°, mientras que el servomotor rota a posiciones especificas dentro de su rango de movimiento. Un motor genera cierta cantidad de energía la cual se utiliza generalmente para mover la rueda. El torque del motor es la fuerza con que el motor hace girar la rueda y la velocidad del motor es la tasa a la cual el motor hace circular la llanta. Hay varios tipos de motores pero los más utilizados en robótica son los motores que funcionan con una señal de control PWM, con la que se controla su velocidad. Una señal PWM (Pulse Width Modulation) es una señal cuadrada periódica a la cual se le modifica el ciclo útil, es decir el ancho del pulso, para transmitir información a través de un canal de comunicaciones o controlar la cantidad de energía que se envía a una carga. En el caso de un motor se varía el ancho de los pulsos (modula) para controlar la velocidad y el sentido de giro [8].

Los engranajes son los elementos utilizados para transferir los movimientos del motor hacia las ruedas. Si se tiene una relación de engranajes de 2:1 se necesita el doble de torque y la mitad de la velocidad de si se tuviera una relación de 1:1. Es decir, los engranajes funcionan como multiplicadores de torque y divisores de velocidad. Cuando están dos engranajes unidos, se dice que uno es el engranaje conductor, el cual provee la fuerza hacia el engranaje conducido. En la figura 1 se muestra un engranaje conductor "driving gear" junto con un engranaje conducido "driven gear":

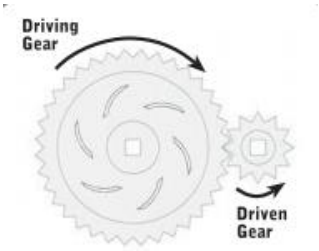


Figura 1 Engranajes conductor y conducido [7]

Entre los engranajes conductor y conducido pueden haber más engranajes (idler) como la figura 2 que se muestra a continuación:

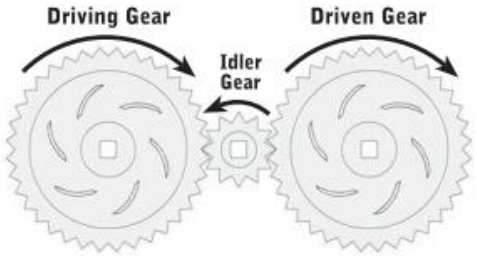


Figura 2 Engranajes conductor, conducido e idler [7]

Las ruedas son el último elemento después de los engranajes y motores, que permiten el movimiento del robot. El diámetro de las ruedas afecta dos características importantes, la velocidad y la aceleración del robot, mientras más grandes sean las ruedas, el robot se desplaza con menor aceleración y mayor velocidad. La fricción en las ruedas afecta el movimiento constante cuando hay dos superficies en contacto y es la que hace que la aceleración sea posible, de lo contrario sería como un carro girando en una superficie de hielo. La fricción también es responsable de bajar la velocidad del robot. Si el robot se mueve en una superficie pegajosa, tendrá un movimiento más lento que si estuviera sobre una superficie lisa [7].

Subsistema de alimentación

La alimentación de un robot es muy importante para la operación de todas las partes electrónicas incluyendo los controladores, motores y demás componentes integrados en el robot. Ésta puede ser suministrada por un adaptador de pared o una batería con el voltaje y la corriente necesaria para energizar los diferentes sistemas. El uso de las baterías evita la dependencia del adaptador de pared, logrando mayor libertad en los movimientos del robot.

Una batería tiene electrones en un área con alto potencial energético y en otra con bajo potencial, donde los electrones viajan de un lado hacia el otro. El voltaje es la diferencia del potencial eléctrico entre estas dos partes de la batería, es decir, si una batería es de 7.2 V es porque la diferencia entre la región positiva y la región negativa es de este valor. Existen varios tipos de baterías pero las más recomendadas para robots son las baterías recargables NiCd (Nickel-Cadmium) debido a la gran cantidad de corriente que suministran. Para escoger la batería que alimentara los diferentes elementos del robot se debe tener en cuenta el voltaje de funcionamiento y la corriente necesaria de los mismos [7].

Subsistema de sensores

Este subsistema le da al robot la habilidad de detectar varias cosas en el ambiente. Los sensores son los ojos y oídos del robot, y gracias a estos, el robot puede conocer el medio ambiente en el que se desenvuelve y ajustar su comportamiento para adaptarse. Existen dos tipos de sensores: análogos y digitales. Los sensores análogos se comunican con el microcontrolador enviando un voltaje eléctrico por medio de un cable que este digitaliza. Un sensor de iluminación por ejemplo, puede indicar la cantidad de luz detectada en un ambiente, enviando un voltaje cero cuando no hay luz y un voltaje máximo cuando la cantidad de luz es alta, y los valores entre estos dos voltajes indican cantidades intermedias de luz. El problema de estos sensores análogos es que es muy difícil lograr enviar y mantener con exactitud los valores específicos de voltaje en un cable de un circuito; los sensores digitales son más confiables pero solo tienen dos valores: alto (1) y bajo (0). Estos sensores envían valores de voltaje así como los análogos, pero solo envían valor máximo y valor mínimo y no valores intermedios. Los valores intermedios son redondeados y enviados como un alto (valor máximo) o un bajo (valor mínimo). Un ejemplo de un sensor digital es un sensor parachoques el cual indica si hay algo tocando el parachoques o no y solo necesita estos dos valores de alto o bajo. Los sensores de límite, como el que se muestra en la figura 3, son también otro ejemplo de sensores digitales, tienen un brazo delgado el cual indica cuando ha sido empujado hacia abajo, haciendo un corto circuito. Normalmente la señal de límite es una señal en alto [1], cuando hay un corto en el circuito, se sabe que se ha llegado al límite por que la señal ahora recibida es una señal en bajo (0) [7].

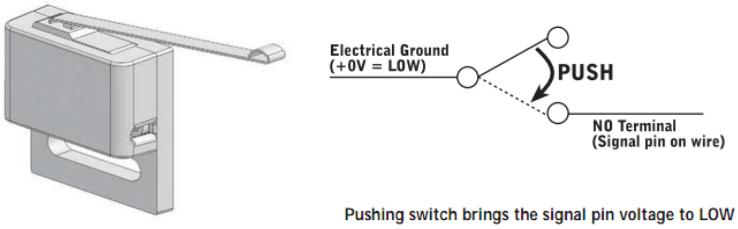


Figura 3 Sensor de límite y su funcionamiento [7].

Existen diferentes tipos de sensores a parte de los ya mencionados como los seguidores de línea, potenciómetros y encoders. Los encoders son sensores de cuadratura que tienen dos salidas donde se pueden medir la posición y la dirección de rotación, siendo esta última la que indica si el movimiento es hacia adelante o hacia atrás. Los potenciómetros son dispositivos cuyo valor de resistencia se relaciona con una posición angular específica.

Subsistema de control y lógica

El subsistema de control permite programar las tareas encomendadas al robot y para esto se usan microcontroladores, que son considerados el cerebro del mismo. Los microcontroladores son dispositivos que realizan operaciones siguiendo una serie de instrucciones ordenadas por el usuario, integrando una unidad central de procesamiento (CPU), una memoria, y unidades de entrada y salida. Las memorias ROM y RAM del microcontrolador, son las encargadas de guardar las instrucciones del programa que se desea ejecutar, y los datos que manipula. La CPU es el cerebro del microcontrolador y está encargada de descargar las instrucciones de la memoria, interpretarlas y ordenar su ejecución. Las unidades de entrada y salida permiten el ingreso de las variables utilizadas por la CPU para dar órdenes y obtener un resultado representado en una salida [9]. En la figura 4 se muestra un esquema de bloques general de un microcontrolador.

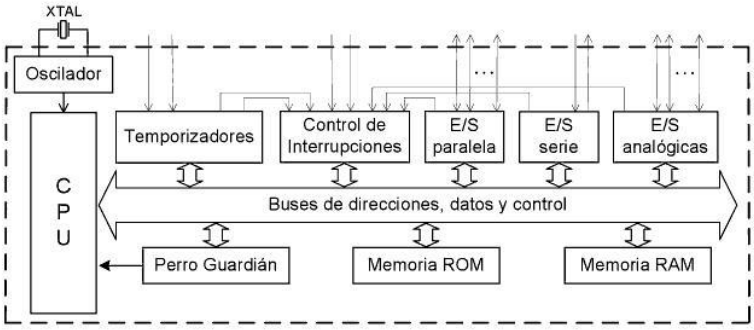


Figura 4 Esquema de bloques de un Micro controlador [9]

Para programar los microcontroladores se deben dar instrucciones para que los procesadores las ejecuten y estas instrucciones deben estar en un lenguaje denominado “lenguaje máquina” para que pueda ser decodificado. El lenguaje máquina se asocia con un código mnemotécnico más sencillo de utilizar que permite la comunicación entre el usuario y la máquina directamente. Al conjunto de los códigos mnemotécnicos que puede ejecutar un procesador se le llama “lenguaje ensamblador” y la función de un compilador es traducirlo en lenguaje máquina para que así pueda entenderlo el procesador [10].

2.2 Modelado dinámico de un robot monociclo

El control y dominio de un robot son aspectos importantes en robótica, pues se construyen generalmente con el fin de realizar tareas específicas asignadas por el hombre, como en una planta de producción de alguna empresa manufacturera. Para realizar el control sobre el robot se debe conocer su modelo dinámico en el cual se relacionan las entradas y las salidas del mismo [1].

El modelo dinámico de un robot es la relación matemática del movimiento en función de las fuerzas que se aplican, con el cual se puede obtener, por ejemplo, la posición y la velocidad final con valores de

entrada puntuales. Es un método que sirve para representar los aspectos geométricos y dinámicos de un sistema y de esta manera conocer matemáticamente el comportamiento del movimiento en el tiempo, información muy importante en el momento de diseñar un control para ese sistema [1].

El modelado dinámico de un robot se logra al conocer la relación existente entre sus diferentes partes y la caracterización de cada uno de sus componentes. La complejidad aumenta con los grados de libertad que tiene el robot, los cuales indican la dimensión de su espacio geométrico. Así, un robot de 2 grados de libertad opera en un plano (2 dimensiones) mientras que uno de 3 grados de libertad en un espacio (3 dimensiones) [1].

Un robot se puede representar como un conjunto de partes unidas por junturas, que pueden ser rotatorias o lineales, presentándose movimientos rotacionales o lineales, respectivamente, los cuales se tienen en cuenta en el modelo dinámico del sistema. Existen otros aspectos como el funcionamiento mecánico o características funcionales de operación que no se deben tener en cuenta [1].

2.2.1 Definición de un Robot Monociclo

Los robots móviles se pueden modelar de dos maneras; como monociclo o como un carro. Un robot tipo carro posee dos pares de llantas, un par conectadas a un motor y el otro a un sistema mecánico que controla la dirección del sistema, tal y como funciona un carro real. Un robot monociclo (figura 5) se refiere a un robot que posee dos motores independientes, moviendo cada uno una llanta, donde la velocidad angular y el sentido de giro de cada motor controla la dirección, el sentido y la posición final del robot [11].

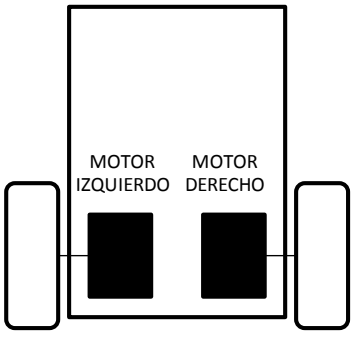


Figura 5 Representación de un Monociclo [autoridad propia]

2.2.2 Modelo de un Robot Monociclo

El modelo dinámico de un monociclo relaciona la salida del sistema con sus entradas, para simular el comportamiento del mismo, donde las entradas son las velocidades de cada uno de los motores, y la salida es la velocidad y la dirección de movimiento del carro. Para el desarrollo se debe ubicar el monociclo en un plano que permita inferir las relaciones matemáticas correspondientes.

El monociclo, por tener dos grados de libertad, se debe ubicar en un plano de 2 coordenadas, en este caso x_1 y x_2 como se ve en la figura 6. La dirección corresponde al ángulo θ que se forma entre la línea perpendicular al eje de las llantas y el eje x_1, siendo el eje de las llantas una línea que las une de centro a centro (T en la figura 6). Suponiendo el origen del plano como la posición inicial, las variables que definen la posición final del carro son la distancia recorrida en el eje x_1, la distancia recorrida en el eje x_2,

y la dirección representada con el ángulo θ. La ecuación 1 expresa las variables de posición en una matriz.

$$Variables\ de\ posición = \begin{bmatrix} x_1 \\ x_2 \\ \theta \end{bmatrix}$$

Ecuación 1 Variables de posición

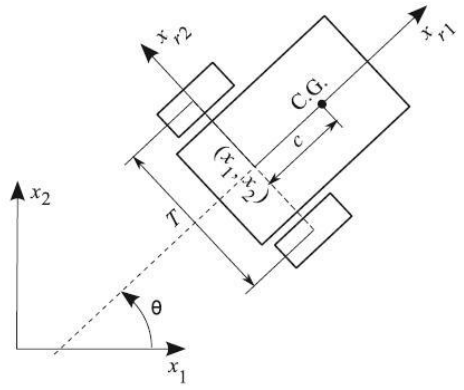


Figura 6 Representación del plano de un monociclo [autonomous]

El punto (x_1, x_2) definido en la figura 6, representa en coordenadas la posición global del monociclo, el cual no tendrá componentes de velocidad laterales por ir paralelo al plano de las llantas. En otras palabras, aunque el robot se mueve con velocidad angular y lineal, el punto (x_1,x_2) se mueve únicamente de forma lineal.

La relación de la velocidad lineal con la dirección se observa en la figura 7, de donde se deducen las ecuaciones 2 y 3.

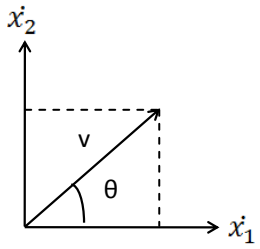


Figura 7 Relación de velocidad lineal con la dirección [autoridad propia]

$$\dot{x}_1 = v\cos\theta$$

Ecuación 2

$$\dot{x_2} = v \sin\theta$$

Ecuación 3

Donde $\dot{x_1}$ es la velocidad en el eje X_1, $\dot{x_2}$ es la velocidad en el eje X_2, y v es la velocidad lineal del robot en el punto (x_1,x_2).

Por otro lado, la velocidad angular del robot representa los cambios en θ en el tiempo, es decir, la derivada de la dirección es la velocidad angular, como se representa en la ecuación 4.

$$\dot{\theta} = \omega$$

Ecuación 4

Convirtiendo las ecuaciones 2, 3 y 4 en un sistema matricial, representado con la ecuación 5.

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Ecuación 5

La velocidad lineal y angular del robot están relacionadas con la velocidad angular de los motores por medio de las ecuaciones 6 y 7.

$$v = r\left(\frac{\omega_{motor\ izquierdo} + \omega_{motor\ derecho}}{2}\right)$$

Ecuación 6

$$\omega = r\left(\frac{\omega_{motor\ derecho} - \omega_{motor\ izquierdo}}{T}\right)$$

Ecuación 7

Donde v es la velocidad lineal del robot, ω es la velocidad angular, *r* el radio de las llantas y *T* la distancia entre ellas como está indicado en la figura 6. Este sistema de ecuaciones se plantea en la ecuación 8.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{T} & -\frac{r}{T} \end{bmatrix} \begin{bmatrix} \omega_{derecho} \\ \omega_{izquierdo} \end{bmatrix}$$

Ecuación 8

Utilizando las ecuaciones 5 y 8, se puede llegar al modelo dinámico final del monociclo, tomando como salida las velocidades en cada eje y la velocidad angular del robot, y como entradas las velocidades

angulares de cada uno de los motores. La posición final, referida a una posición inicial, está compuesta por una variación en x_1 , x_2 , y θ.

2.3 Comunicaciones

Los sistemas de comunicaciones en la actualidad se han convertido en una herramienta que involucra directamente a todas las personas y los procesos que se llevan a cabo cotidianamente. Un sistema de comunicaciones es un conjunto de elementos que permiten el envío y recepción de cualquier tipo de información de un lugar a otro. El diagrama de bloques de la figura 8 resume los componentes de un sistema de comunicaciones, en el cual se encuentra una fuente de información, la cual va conectada a un transmisor que modula y envía la información a través de un medio de transmisión que la transporta hasta el receptor, que está conectado al destino correspondiente.

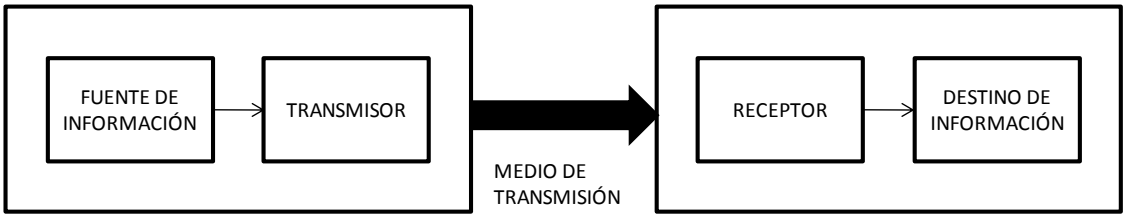


Figura 8 Diagrama de Bloques de un Sistema de Comunicaciones [12]

Los sistemas de comunicaciones tienen ciertas características generales que se definen para su diseño, dependiendo de la aplicación que se requiera. El primero es el tipo de señal que se va a transmitir; análoga o digital. La diferencia radica en que las señales análogas son señales continuas en el tiempo que pueden ocupar cualquier valor de amplitud en un intervalo, mientras que las señales digitales son señales construidas con un número finito de puntos en un intervalo de tiempo. Actualmente, las señales análogas se convierten a digitales para su transmisión, ya que las señales digitales son de mayor resolución y permiten ser procesadas en equipos digitales como un computador. Un ejemplo de esto es la transición de televisión análoga a digital, que convierte la televisión tradicional pasiva, en una televisión interactiva donde se ofrece al usuario opciones como cambio de idioma, subtítulos y grabación entre otros [12].

Existen tres parámetros importantes que se deben definir durante el diseño de un sistema de comunicaciones. En primera instancia, la potencia promedio de la señal a transmitir, definida como el producto del voltaje con la corriente promedio; por otro lado la relación señal a ruido, que indica la cantidad de ruido que puede estar afectando la señal de información, y finalmente, el ancho de banda o rango de frecuencias que contiene la señal. Con estos parámetros se define, por ejemplo, el medio de comunicación más conveniente o el tipo de transmisor más adecuado.

2.3.1 Fuentes de información

Una fuente de información es un sistema que tiene la capacidad de producir mensajes por medio de señales análogas o digitales, que pueden ser la representación de una imagen o un dato, como se detallará a continuación.

Imágenes

Las imágenes son fuentes bidimensionales que se obtienen generalmente por medio de una cámara, la cual está basada en el funcionamiento del ojo humano. Estas pueden ser tanto estáticas (fotos) como dinámicas (video), y se pueden transmitir de manera análoga o digital. El video se puede definir como la sucesión de varias imágenes estáticas a una velocidad tal que el ojo humano no alcanza a percibir.

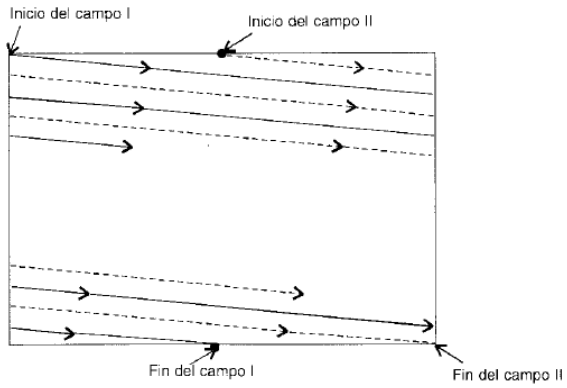


Figura 9 Campos del cuadro de una imagen [14]

Para transmitir una imagen, se debe hacer un escaneo de la misma, de izquierda a derecha, de arriba abajo, donde la secuencia horizontal se define como una línea, y la sucesión vertical corresponde a un cuadro. Como se observa en la figura 9, la imagen se divide de arriba abajo en varias líneas que conforman dos campos, uno contiene las líneas pares y el otro las impares. Al alcanzar el extremo de cada línea, el punto de escaneo se devuelve rápidamente a la izquierda para iniciar la siguiente línea [13].

La calidad de reproducción de una imagen depende tanto de su resolución horizontal, como de la vertical. La resolución vertical es el número de líneas disponibles en un escaneo, y la resolución horizontal es limitada por el ancho de banda del canal que transmite el video. Estos factores dependen del sistema de codificación que se utilice. El sistema PAL (Phase Alternating Line) es el sistema de codificación utilizado en la mayoría de países del continente africano, europeo, Australia y algunos países americanos. Escanea 625 líneas por cuadro. Por otro lado, el sistema NTSC (National Television System Comitte) que constituye el estándar de Estados Unidos, Japón, Francia y la mayoría de países de América, tiene una resolución de 525 líneas y un ancho de banda de 4.5 MHz. [13].

El procesamiento de imágenes es un tema que ha venido teniendo cabida en los centros de investigación ya que ha producido un fuerte impacto en temas de visión artificial. Este consiste en captar y manipular información espacial (imágenes) en forma vectorial para analizarla desde un equipo especializado que

generalmente es un computador [15]. Las imágenes transmitidas en forma análoga necesitan ser digitalizadas para ser manipuladas desde este dispositivo terminal.

Finalmente, los computadores se han convertido tanto en una fuente como en un receptor importante de información, que además de poder procesar, transmitir y recibir sonidos, imágenes y videos, es capaz de generar cualquier tipo de información y de intercambiarla con otros dispositivos por medio de algún protocolo de comunicación, ya sea por conexión física o inalámbrica. Esta información se genera digitalmente y comúnmente de codifica en ASCII (American Standard Code for Information Interchange), que consiste en un código de 128 caracteres que pueden ser números, letras o signos. Cada carácter está formado por 7 bits de datos, donde un bit es un digito binario que puede ser 1 o 0, y los caracteres van desde 0000000 hasta 1111111 pasando por todas las posibles combinaciones [14].

2.3.2 Transmisión de la Información

Para poder transmitir una señal desde una fuente de información, se deben realizar dos procesos; primero, convertir la información en una señal eléctrica, y después pasarla por el proceso de modulación. Este proceso consiste en llevar el mensaje a través de una portadora, que varía sus características como la amplitud, la frecuencia o la fase dependiendo del tipo de modulación utilizada, produciendo variaciones equivalentes a la señal de mensaje [14].

Se utiliza una portadora generalmente sinusoidal, de frecuencia mucho mayor que el mensaje que se desea transmitir. Esta portadora es sintonizada en el receptor de acuerdo a su frecuencia. Los tipos de modulación que se utilizan generalmente son modulación por amplitud (AM), modulación por frecuencia (FM) o modulación por fase (PM). Las señales digitales son discretas tanto en tiempo como en amplitud y los tipos de modulación digital que generalmente se utilizan son FSK (modulación por desplazamiento de frecuencia), PSK (modulación por desplazamiento de fase) y QAM (modulación de amplitud en cuadratura) [12].

2.3.3 Transmisión desde un Computador

Existen varias formas de transmitir un carácter desde un computador; entre ellas por puerto paralelo o por puerto serial. En el puerto paralelo se envía 1 carácter completo en un instante de tiempo, es decir, se envían 8 bits por el mismo cable, además de las señales de control correspondientes. Este puerto se utilizaba para conectar la impresora al computador antes de ser fabricadas con conexión por puerto USB como la mayoría de las actuales. En el puerto serial, se envía 1 bit a la vez, de forma secuencial, de manera que el receptor organiza los bits en orden de llegada para identificar el carácter que se está recibiendo. Para evitar errores, se envía primero un bit de inicio, después los 7 bits del carácter, en orden del bit más significativo al menos significativo, seguido de 1 bit de paridad y 1 bit de parada, tal como se

observa en la figura 10. De esta manera el receptor identifica el inicio y el fin de una trama, así como cualquier error que se pudo producir en la recepción. La salida USB o RS-232 del computador son ejemplos de puerto serial [14].

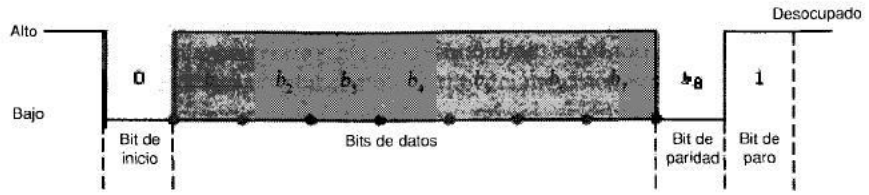


Figura 10 Secuencia de Datos en Comunicación Serial [14]

2.3.4 Medios de Transmisión

Las señales deben pasar por un medio de transmisión que conduce la información desde el transmisor hacia el receptor. Los medios se pueden clasificar en alámbricos, ópticos o electromagnéticos (inalámbricos) como se observa en la tabla 1. El medio se elige dependiendo de la aplicación y de algunos factores como la distancia entre receptor-transmisor o la potencia de la señal a enviar [16]. Para este proyecto se utilizó como medio de transmisión es espacio atmosférico.

TIPO DE MEDIO	EJEMPLO
ALÁMBRICOS	Par Trenzado
	Cable Coaxial
ÓPTICOS	Fibra Óptico
ELECTROMAGNÉTICOS	Espacio Atmosférico

Tabla 1 Tipos de Medios de Transmisión [16]

Los medios alámbricos son cables físicos conectados del transmisor al receptor, fabricados generalmente en cobre, lo que permiten el transporte de una señal de un lado a otro. Dentro de ellos los más comunes son el par trenzado y el cable coaxial. El primero consiste en uno o varios pares de cables trenzados (entrelazados entre sí) cuya torsión evita interferencias eléctricas e inducción electromagnética. Este tipo de cable se utiliza comúnmente en telefonía. El cable coaxial lo compone un conductor central cubierto por un aislante y una malla metálica que protege las señales transmitidas de señales parasitas que puedan afectar la información. Este presenta un ancho de banda mayor al par trenzado, menor radiación electromagnética y mayor dificultad de intercepción (protección a la información). Es utilizado, por ejemplo, en la distribución de televisión por cable [16].

La fibra óptica es un conductor cilíndrico hecho de vidrio el cual transmite un haz de luz que se refleja en las paredes internas con pérdidas casi nulas, logrando cubrir una mayor distancia sin necesidad de repetidoras. Su principio de funcionamiento es el envío de bits (1 o 0) representados por dos estados; luz apagada o luz encendida. Entre sus ventajas se puede destacar el ancho de banda, su pequeño diámetro, el material flexible que permite facilidad en su manipulación y las pequeñas pérdidas que se producen por tener un recubrimiento resistente. Este es uno de los medios más utilizados en todo el mundo para transmisión digital de información a grandes distancias, usado por ejemplo en la distribución de internet por cable [16].

El espacio libre constituye el medio de transmisión inalámbrico que permite comunicar dos puntos sin necesidad de conexiones físicas, convirtiendo la información en ondas que se propagan por el aire desde un transmisor hacia un receptor. Para este medio, los transmisores y receptores son denominados antenas, las cuales cumplen la función de convertir energía eléctrica en ondas electromagnéticas que tienen la propiedad de propagarse en el aire. El espacio atmosférico, utilizado como medio de transmisión, ha permitido acercar distancias y comunicar lugares remotos [16].

2.3.5 Comunicaciones Inalámbricas

Los sistemas de comunicación inalámbrica no utilizan medios de transmisión que conecten físicamente el transmisor con el receptor, por el contrario, usan el espacio libre para el envió de señales. Esta tecnología ha venido posicionándose en la vida cotidiana permitiendo en la actualidad el acceso a la información desde dispositivos personales como un celular o un reproductor musical independientemente de la ubicación en la que se encuentre.

Un protocolo de comunicación permite mediante el seguimiento de reglas, la comunicación entre dos dispositivos, siempre y cuando los dispositivos entiendan el protocolo. Existen varios estándares que han sido creados para regular las comunicaciones inalámbricas, los cuales buscan clasificar diferentes protocolos según las características que poseen. La tabla 2 presenta un comparativo de la distancia de cubrimiento entre algunos de los estándares que están aprobados por el comité de la IEEE 802. En una aplicación, el tipo de estándar que se utiliza debe ser acorde a las exigencias de la misma, tal como la potencia de la señal a transmitir, la distancia que se desee cubrir y el costo a invertir [17].

Nombre	Distancia Máxima	Tecnología Utilizada
WMAN (wireless metropolitan area network)	Kilómetros	WiMAX
WLAN (wireless local aerea network)	Cientos de Metros	Wi-Fi
WPAN (wireless personal area network)	Décadas de metros	ZigBee Bluetooth WiMedia

Tabla 2 Redes y Tecnologías Inalámbricas aprobadas por la IEEE 802 [17]

ZIGBEE

Zigbee es un protocolo estándar de comunicación diseñado por la empresa ZigBee Alliance que se creó con el objetivo de lograr comunicaciones inalámbricas de baja potencia y bajo costo, que implican un consumo mínimo de energía contribuyendo al ahorro energético [17].

El protocolo Zigbee soporta diferentes configuraciones entre terminales (puntos que envían o reciben información), tales como comunicación punto a punto, multipunto, estrella o árbol, que se logran definiendo ciertas características en cada uno de sus extremos. Las características que se deben definir son, la dirección propia y de destino, la dirección PAN y el canal a utilizar [18].

La dirección propia define la identidad del terminal. La dirección de destino indica hacia donde y desde donde se quiere compartir información. Las direcciones son únicas para cada terminal. La dirección PAN simula una pequeña red, donde varios terminales pueden pertenecer a la misma PAN y de esta manera comunicarse entre ellos. El canal puede contener varias direcciones PAN, y se pueden definir varios canales para separar comunicación entre terminales. La figura 11 muestra un diagrama de una posible configuración de dirección, PAN y canal [18].

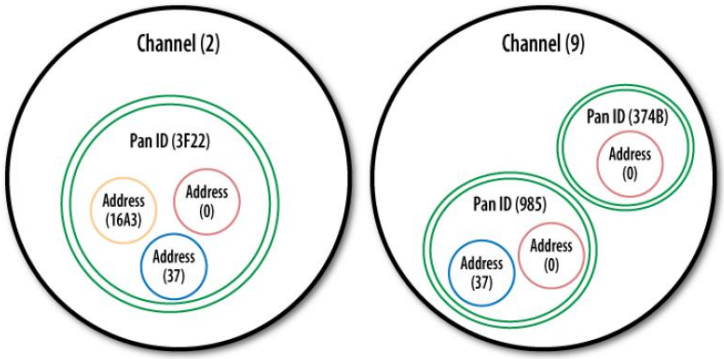


Figura 11 Diagrama del canal, Pan y Direccion en Zigbee [18]

Se diseñó a partir de este protocolo, el hardware correspondiente que soporta el estándar ZigBee, y funciona a un rango de frecuencias de 2.4 GHz utilizando un microcontrolador de radiofrecuencia. Esta tecnología aunque no cubre una gran distancia, es ideal para usar en el sector de automatización de edificios. Un ejemplo de este hardware son las tarjetas XBee [18].

3 ESPECIFICACIONES

El proyecto consiste en diseñar una interfaz que permita controlar los movimientos de un robot móvil de (3-DOF) y visualizar en el computador la imagen de una cámara que se encuentra sobre el robot, por medio de comunicaciones inalámbricas. El diseño del proyecto es posible dividirlo en dos partes, la construcción física (hardware) y la configuración de software, como se describe a continuación.

3.1 Hardware

3.1.1 Estructura

La plataforma robótica consiste en un robot móvil tipo monociclo de 4 ruedas, 2 ubicadas al costado derecho del robot y las 2 restantes al costado izquierdo, cada par con su respectivo motor. Sostiene una cámara en su parte delantera, teniendo libertad de movimientos verticales con limitaciones angulares de giro. La estructura general del robot es un carro con dos niveles que facilita la ubicación de las tarjetas de desarrollo y las baterías que se utilizan para el funcionamiento del mismo. Las dimensiones donde se encuentran distribuidos todos los componentes son de 33 cm de alto por 33 cm de largo y 20 cm de ancho.

Los componentes del robot son 4 llantas, dos motores DC que permiten el desplazamiento del carro por un plano, 2 encoders conectados a cada uno de los motores para medir la velocidad, un servo motor que realiza el movimiento de posición de la cámara, dos sensores de límites que indican la posición superior e inferior máxima de la cámara, y un potenciómetro que se usa como sensor de posición angular. El peso total del robot con todos sus componentes es de 3.5 kg. En el anexo A se encuentran las especificaciones de cada uno de los componentes del robot.

La estructura se puede dividir en tres secciones; dos niveles horizontales, y el soporte vertical de la cámara. El nivel horizontal inferior sostiene los dos motores, cada uno conectado a un costado del robot, la tarjeta de desarrollo de Arduino UNO, la batería de la cámara y un circuito que permite las conexiones entre los componentes del robot y el microcontrolador. La conexión entre las llantas laterales y su respectivo motor está dada por un sistema mecánico de engranajes. El nivel superior horizontal es una estructura plana en forma de H, que sostiene únicamente la batería que alimenta el Arduino UNO. Finalmente, el soporte vertical de la cámara consiste en dos columnas que están unidas a dos engranajes directamente conectados a un servo motor, que permite el movimiento de la cámara, como se observa en la figura 12.

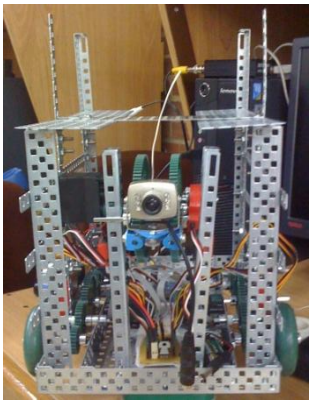


Figura 12 Vista frontal del Robot

El chasis del carro fue construido utilizando la estructura de aluminio del kit de Vex Robotics, la cual provee flexibilidad para el diseño porque permite conectar sensores o motores en toda la superficie mediante tornillos o varas cuadradas de metal. De esta manera, es posible tener mayor estabilidad y rigidez en el montaje del robot. El Anexo B presenta las imágenes de la estructura final del carro.

3.1.2 Imagen

El tercer grado de libertad del robot es el movimiento vertical de la cámara, el cual se obtiene utilizando un servomotor conectado a un sistema rotacional unido al soporte de la misma.

La cámara utilizada es una mini cámara inalámbrica infrarroja Radio AV que tiene un transmisor incorporado y la imagen llega hasta un receptor de video que está incluido en el paquete de la misma. En la Figura 13 se muestra la cámara utilizada junto a su receptor.

Figura 13 Cámara y receptor [26]

Figura 14 Batería de 9 V [24]

Ésta cámara se alimenta con una batería de 9 V para que los movimientos del robot tengan mayor independencia. Se escogió esta batería por sus pequeñas dimensiones de 20.2x17.5x49.2 mm. La Figura 14 muestra la imagen de la batería utilizada.

El receptor se conecta a una capturadora de video que convierte la imagen analógica recibida en imagen digital. La capturadora utilizada es la Sensoray modelo 2250s. Este dispositivo tiene una salida de puerto AB y utilizando un cable AB/USB se conecta con el computador por puerto USB, la capturadora también posee una entrada BNC donde se conecta el receptor de video (con un adaptador y un cable de video RCA), y además no necesita alimentación externa ya que se alimenta directamente del computador utilizando la conexión USB. En la Figura 15 se muestra la imagen de la capturadora utilizada.

Figura 15 Capturadora Sensoray 2250s [20]

Para el uso de la capturadora es necesario instalar un software y seguir los pasos de instalación de los drivers que se encuentran en el manual y se pueden descargar de la página de internet del fabricante [20]. Una vez instalado el software la capturadora se conecta directamente al computador y la imagen digitalizada es visualizada.

3.1.3 Control del robot

El control de los movimientos del robot se realiza desde el microcontrolador Arduino UNO, al cual van conectados todos los pines de los componentes. Los datos de control se envían inalámbricamente desde el computador, utilizando el protocolo ZigBee y usando como hardware dos tarjetas XBee.

Arduino UNO

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexible y sencillo de utilizar. Dentro de su portafolio se encuentra la tarjeta Arduino UNO (Figura 16) que está basada en el microcontrolador ATmega 328, caracterizado por ser un dispositivo de alto rendimiento y bastante rápido. El Anexo C presenta las características principales de este microcontrolador [21].

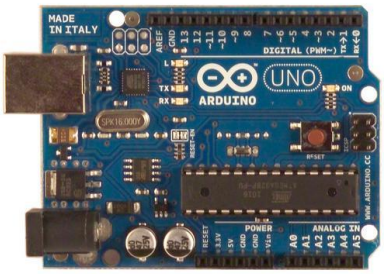


Figura 16 Tarjeta Arduino UNO [21]

La tarjeta Arduino UNO es utilizada en este proyecto para el control del robot, conectando cada uno de los componentes a los diferentes pines entradas/salida que ofrece dependiendo del tipo de información que se requiera, ya sea digital, análogo o PWM. A parte de los pines de entrada/salida, Arduino UNO ofrece salidas de alimentación que regulan la entrada a valores de 5V y 3.3V y Vin (Voltaje de alimentación de la tarjeta). El Anexo D presenta las especificaciones y el esquemático de Arduino UNO. La salida de 5 V es utilizada para alimentar los sensores de límite, el potenciómetro, los encoders y el servomotor. Los motores DC son alimentados directamente de la batería de 9.6 V y 1000mA con la cual se energiza la tarjeta.

XBEE

Las tarjetas Xbee (figura 17) son módulos de radiofrecuencia diseñados para permitir la conectividad inalámbrica entre dos puntos. Utilizan el protocolo de comunicación digital inalámbrica IEEE 802.15.4 para asegurar una rápida conexión de punto a punto. Estos dispositivos trabajan con una comunicación serial para el ingreso de datos y configuración [22].

Figura 17 Tarjetas XBee [22]

Las Xbee tienen una tasa de transmisión de 250kbps y operan dentro de un rango de frecuencias cercana a 2.4GHz. Pueden llegar a cubrir entre 40m y 90m en espacio cerrado, y entre 120m y 320m en espacio abierto, dependiendo de la tarjeta utilizada. Poseen 10 puertos de entrada/salida digital, y entradas ADC de 10-bit, y requieren de una alimentación entre 2.1 y 3.6 VDC, tal como se presenta en el Anexo E.

El proyecto utiliza dos tarjetas XBee configuradas para transmisión punto a punto, asegurando la comunicación bilateral al definir las direcciones propias y de destino de cada una y establecer el canal y la dirección PAN. La configuración de las tarjetas se hizo utilizando el programa X-CTU que permite la modificación de estos parámetros.

Para configurar las XBee, se requiere un módulo que las reciba y se comunique al computador mediante puerto serial. Para este proyecto se utilizo un módulo MaxStream que se comunica con el computador por puerto RS-232.

3.2 Software

3.2.1 Matlab®

Matlab® es un software que maneja lenguaje de alto nivel y ambientes interactivos que permiten desarrollar diferentes programas en lenguajes como C, C++ y Fortran. Matlab® se puede usar para diferentes aplicaciones como procesamiento de señales e imágenes, comunicaciones, diseño de control, pruebas y medidas, gráficas en 2D o 3D y el análisis y modelamiento financiero entre muchas otras. Matlab® además tiene funciones que permiten la transmisión de datos a otros programas dentro del computador, o fuera de éste por puerto paralelo o serial. Este programa se basa en el uso de funciones ya diseñadas por los fabricantes [23].

Dentro de las aplicaciones que ofrece Matlab, la opción GUIDE permite el diseño de interfaces gráficas por parte del usuario, utilizando botones, menús, y entradas de texto entre otros. Éste proyecto utiliza el Guide y la comunicación por puerto serial para la interfaz. La interfaz diseñada contiene una ventana donde se observa la imagen transmitida por la cámara, unos botones que permiten seleccionar hacia donde se desea mover la cámara; arriba o abajo, y además despliega la posición en grados en la que se encuentra la cámara, y si se encuentra en alguno de sus límites (superior o inferior). Por otro lado, permite seleccionar el sentido de movimiento del robot, aumentar o disminuir la velocidad de cada motor de manera independiente y visualizar su valor.

3.2.2 Arduino

Para el control de los movimientos del robot y de la cámara así como toda la información proporcionada por los sensores, se utilizo el lenguaje Arduino, el cual está basado en el lenguaje de programación C++ y su software puede ser descargado de forma gratuita desde el sitio web http://arduino.cc/es/Main/Software.

La estructura de un programa del lenguaje Arduino se compone de dos partes fundamentales las cuales contienen los bloques que tienen las declaraciones e instrucciones. Estas funciones son *void setup()* y *void loop().* Las declaraciones e instrucciones van entre llaves para indicar el principio y final del bloque y cada instrucción debe terminar en punto y coma para indicar el final de la instrucción. El *setup()* es la parte encargada de recoger la configuración como inicialización de pines y puertos de comunicación y el *loop()* es la parte que contiene el programa que se realizará constantemente como la lectura de entradas y activación de salidas.

Una función es un bloque de código que se realiza dentro del programa. Las fundamentales son las de *setup()* y *loop(),* pero también pueden ser creadas por el usuario, las cuales deben ir acompañadas del tipo que son. El tipo de función con el que se especifica es el tipo de dato que va a devolver, por ejemplo, una función *int* devolverá un dato de tipo entero.

Una variable es una manera de nombrar y almacenar un valor numérico para poder utilizarlo más adelante en el código. Las variables deben ser declaradas antes de poder ser utilizadas, se debe definir el tipo de variable, asignar un nombre y opcionalmente un valor inicial. El tipo de variable puede ser *int* (entero), *float* (flotante), *long* (largo), entre otros. Se puede declarar al inicio del programa antes del *setup()* o a nivel local dentro de las funciones en las cuales se van a utilizar. Para hacer operaciones matemáticas entre distintas variables, los operadores de suma, resta, multiplicación y división se incluyen en el entorno de programación y devuelven la operación requerida. También pueden haber asignaciones compuestas que combinan una operación matemática con una variable como las mostradas en la Tabla 3.

x ++	*// igual que x = x + 1,*	*o incrementar x en + 1*
x --	*// igual que x = x - 1,*	*o decrementar x en -1*
x += y	*// igual que x = x + y,*	*o incrementra x en +y*
x -= y	*// igual que x = x - y,*	*o decrementar x en -y*
*x *= y*	*// igual que x = x * y,*	*o multiplicar x por y*
x /= y	*// igual que x = x / y,*	*o dividir x por y*

Tabla 3 Asignaciones compuestas en Arduino (basado en C)

Las comparaciones entre variables se utilizan con frecuencia en estructuras condicionales como el *if* y la Tabla 4 muestra los tipos de operadores de comparación.

x == y	*// x es igual a y*
x != y	*// x no es igual a y*
x < y	*// x es menor que y*
x > y	*// x es mayor que y*
x <= y	*// x es menor o igual que y*
x >= y	*// x es mayor o igual que y*

Tabla 4 Operadores de comparación

Las constantes de HIGH y LOW indican los niveles de salida de alto (1) o bajo (0) respectivamente para las lecturas o escrituras digitales. Las constantes de INPUT y OUTPUT definen el funcionamiento de los pines como entrada o salida. Las funciones de *digitalRead* y *digitalWrite* son funciones para leer y enviar respectivamente valores de alto o bajo de un pin especifico.

```
valor = digitalRead(Pin);        // hace que valor sea igual al estado leído en Pin
digitalWrite(pin, HIGH);         // deposita en el 'pin' un valor HIGH (alto o 1)
```

Las funciones de *analogRead* y *analigWrite* leen y escriben respectivamente valores de pines específicos utilizados para lectura y escritura análoga. Para los pines de lectura, se utilizan los pines análogos de la tarjeta que van desde A0 hasta A5. Para escribir un valor en el pin, se utilizan los pines marcados como pines de PWM en la tarjeta.

```
valor = analogRead(pin);         // asigna a valor lo que lee en la entrada ´pin'
analogWrite(pin, valor);         // escribe 'valor' en el 'pin' definido como analógico
```

Arduino también cuenta con funciones de tiempo como la función *delay()*, la cual detiene el programa durante el tiempo especificado dentro de los paréntesis y *millis()*, la cual devuelve el número de milisegundos transcurridos desde el inicio del programa.

Para la transmisión de datos en serie la función que se debe utilizar es *Serial.begin()* y dentro del paréntesis se especifica el valor de la velocidad para la comunicación, la cual normalmente es 9600. Para imprimir el valor del puerto serial, se utiliza la función *Serial.print()* o *Serial.println()* para que imprima el dato y luego haga un salto de línea. Para leer o capturar un dato desde el punto serial se utiliza la función *Serial.available()*, si hay un dato disponible, *Serial.available()* será mayor que cero y el dato es leído utilizando la función *Serial.Read()* [27].

La figura 18 presenta el diagrama de bloques que resume las fases del proyecto.

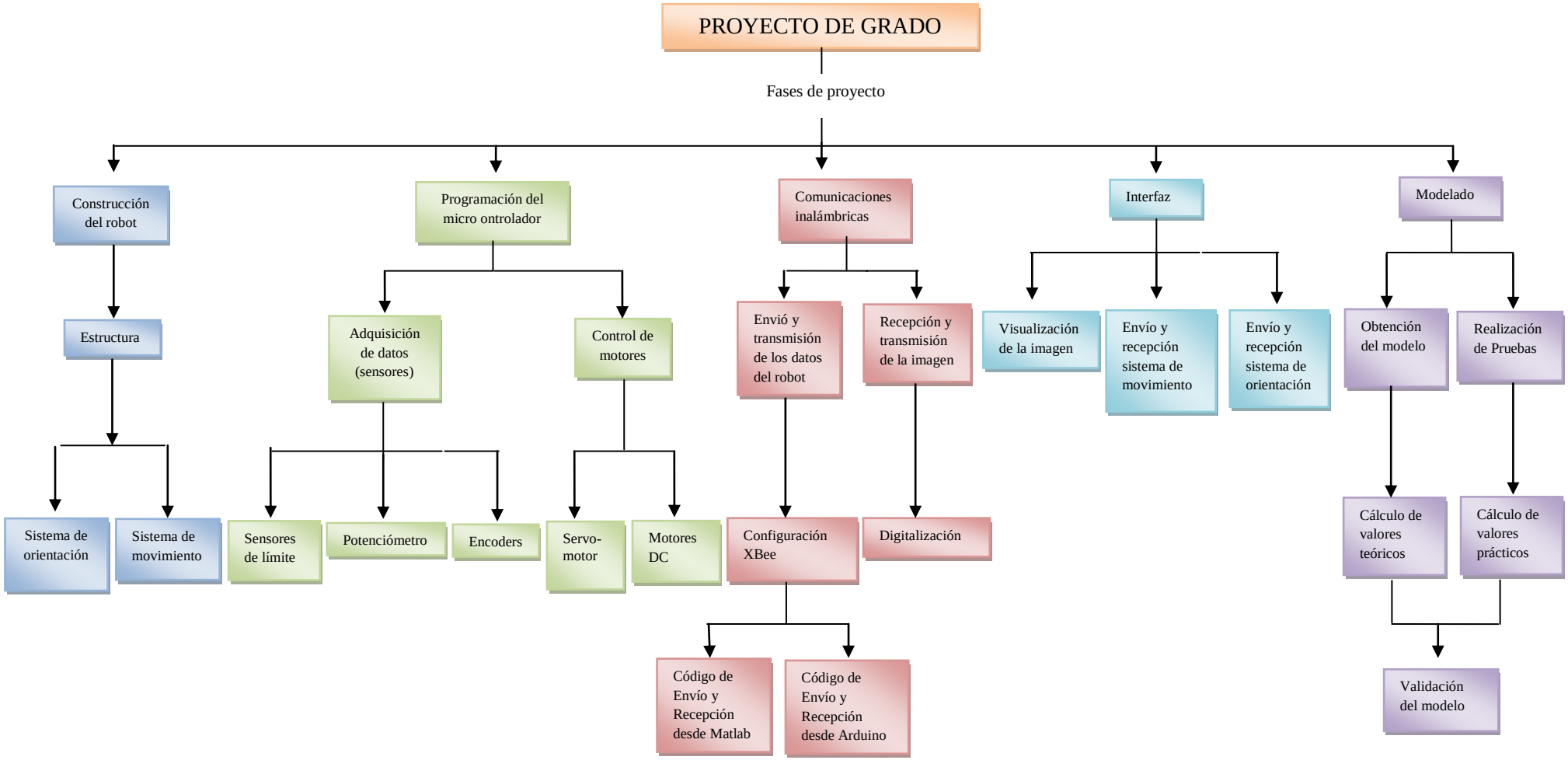


Figura 18 Diagrama de bloques del proyecto

4 DESARROLLO

Para el desarrollo del proyecto se tuvieron en cuenta 5 etapas, la construcción del robot, la programación, el establecimiento de la comunicación entre el computador y el robot, el diseño de la interfaz en Matlab® y el modelado del robot. A continuación se describen detalladamente cada una de ellas.

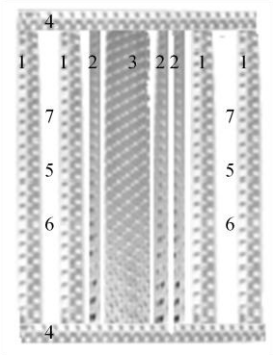
4.1 Construcción del robot

La estructura del robot tiene los siguientes componentes:

- 3 placas largas de aluminio de 6.5 cm x 31.5 cm.
- 1 placa corta de aluminio de 19 cm x 6.5 cm.
- 11 varas largas de aluminio de 31.5 cm.
- 6 rieles de 31.5 cm de largo por 1 cm de ancho y 2 cm de alto.
- 2 rieles de 22 cm de largo por 1 cm de ancho por 2 cm de alto.
- 2 rieles de 20.5 cm de largo por 2.5 cm de ancho y 1 cm de alto.
- 4 engranajes de 60 dientes.
- 4 engranajes de 36 dientes.
- 4 engranajes de 24 dientes.
- 4 llantas de baja fricción.
- 2 encoders ópticos.
- 2 sensores de límite.
- 1 potenciómetro.
- 2 motores DC.
- 1 servomotor.

La estructura del robot puede dividirse en tres secciones: dos niveles horizontales y el soporte vertical de la cámara. El nivel horizontal inferior es la base del robot. Esta sección está conformada por 4 rieles de aluminio, 2 de 20 cm de largo por 1 cm de ancho y 2 cm de alto para las partes delanteras y traseras del robot, y otros 2 rieles de 31.5 cm de largo por 1 cm de ancho y 2 cm de alto para las partes laterales del mismo formando así un rectángulo. En el centro, unido a la placa delantera y trasera del robot se encuentra una placa larga de aluminio de 6.5 cm x 31.5 cm que conforma la base inferior. Al lado derecho de la placa se encuentran 2 varas largas de aluminio unidas también a los canales delantero y trasero, y en el lado izquierdo una sola vara. Junto a las dos varas extremas se unen dos rieles de aluminio de 31.5 cm de largo por 1 cm de ancho y 2 cm de alto, uno a cada lado y con la parte alta de 2 cm colocada hacia la parte externa del robot a una distancia de 3.5 cm de los canales laterales dejando este espacio para los

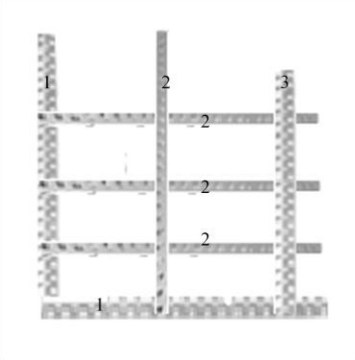
engranajes. En el centro de cada uno de los rieles laterales (aproximadamente a una distancia de 16 cm) se colocan los engranajes de 60 dientes que van conectados a los motores DC y a los encoders. A cada lado de este engranaje van unidos los engranajes de 36 dientes y a estos últimos los de 24 dientes, que están conectados a las llantas. En la figura 19 se muestra la parte inferior del robot.



1. Rieles de 31.5 cm de largo por 1 cm de ancho y 2 cm de alto.
2. Varas largas de aluminio de 31.5 cm.
3. Placa larga de aluminio de 6.5 cm x 31.5 cm.
4. Rieles de 20 cm de largo por 1 cm de ancho por 2 cm de alto.
5. Engranaje de 60 dientes. (unido al motor DC y encoder)
6. Engranaje de 36 dientes.
7. Engranaje de 24 dientes. (unido a una llanta)

Figura 19 Parte inferior del robot

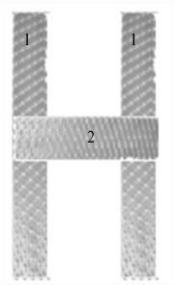
En las esquinas traseras del robot se unen dos rieles de 31.5 cm de largo por 1 cm de ancho y 2 cm de alto y un poco más atrás de las esquinas delanteras (2 cm) se unen 2 rieles de 22 cm de largo por 1 cm de ancho por 2 cm de alto. Se unen varas largas de aluminio a cada una de las esquinas laterales para dar más estabilidad conectando una vara cada 5 cm de abajo hasta arriba hasta completar 3 varas largas horizontales en cada lado. Se coloca un vara larga vertical a una distancia de 10 cm de las esquinas delanteras y 13.5 cm de las esquinas traseras, una a cada lado. La figura 20 muestra la parte lateral del robot.



1. Rieles de 31.5 cm de largo por 1 cm de ancho y 2 cm de alto.
2. Varas largas de aluminio de 31.5 cm.
3. Rieles de 22 cm de largo por 1 cm de ancho y 2 cm de alto.

Figura 20 Parte lateral del robot

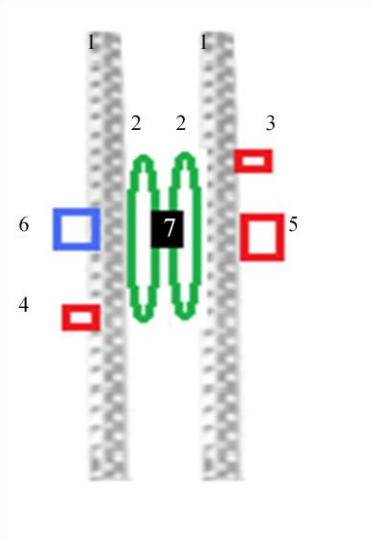
La sección superior del robot tiene 2 placas largas de aluminio de de 6.5 cm x 31.5 cm puestas en forma vertical y una placa corta de aluminio de 19 cm x 6.5 cm en forma horizontal, en el medio de las dos placas verticales. La figura 21 muestra la parte superior del robot.



1. Placa larga de aluminio de de 6.5 cm x 31.5 cm
2. Placa corta de aluminio de 19 cm x 6.5 cm

Figura 21 Parte superior del robot

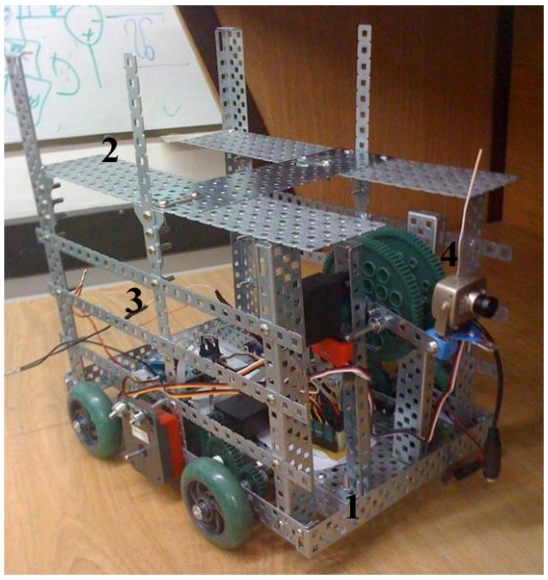
El soporte vertical de la cámara consta de 2 rieles de 20.5 cm de largo por 2.5 cm de ancho y 1 cm de alto colocados verticalmente en la parte delantera del robot a una distancia de 6.5 cm de cada esquina, dejando un espacio de 5 cm entre cada riel. Para el movimiento de orientación de la cámara se utilizan dos ruedas de diámetro de 2.75 pulgadas las cuales se colocan en medio de los dos rieles a una distancia de 15 cm desde la base hasta el centro del sistema. Al sistema rotacional se le conecta a un lado el servomotor y al otro lado el potenciómetro. La cámara se sujeta de este sistema y los sensores de límites se colocan en los rieles, el límite inferior sobre riel izquierdo a una distancia de 14 cm de la base y el límite superior sobre el riel derecho a una distancia de 18 cm de la base. La figura 22 muestra el soporte vertical de la cámara del robot.



1. Rieles de 20.5 cm de largo por 2.5 cm de ancho y 1 cm de alto.
2. Sistema de orientación.
3. Límite superior.
4. Límite inferior.
5. Potenciómetro.
6. Servomotor.
7. Cámara.

Figura 22 Soporte vertical de la cámara

La figura 23 muestra el robot construido en su totalidad.



1. Parte inferior (base) del robot.
2. Parte superior del robot.
3. Parte lateral del robot.
4. Sistema de orientación.

Figura 23 Robot final

4.2 Programación del robot

Para programar la tarjeta Arduino UNO, primero se debe descargar el software en la página del fabricante: *http://arduino.cc/es/Main/Software* y escoger el tipo de sistema operativo en el cual se va a instalar el programa. Al terminar la instalación se prueba la tarjeta entrando a la carpeta Arduino en la cual se instalo el software y abriendo un esquemático de programación, como lo ilustra la figura 24. Para probar por primera vez la tarjeta, se programa el ejemplo llamado *Blink* que consiste en hacer parpadear el led que se encuentra en el pin 13 de la tarjeta.

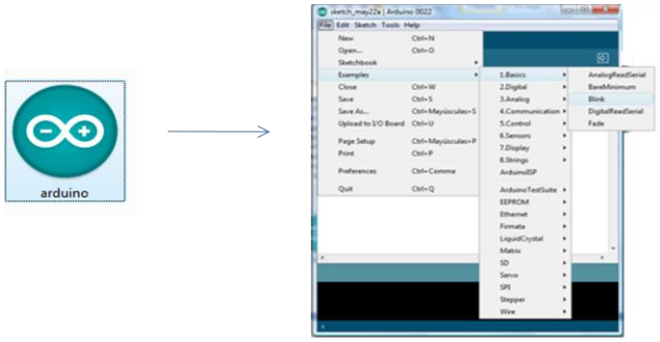


Figura 24 Iniciando en Arduino

4.2.1 Motores DC y servomotor

Los motores DC y el servomotor poseen 3 conexiones, un cable rojo para la alimentación (para los motores DC 9.6 V y para el servomotor 5V), uno negro para tierra y uno blanco para la señal PWM que cambia la velocidad del motor DC o la posición del servomotor.

Por medio del programa Arduino y modificando el ejemplo Servo- Sweep, se diseñó un programa que variara el ciclo útil de una señal cuadrada de 50 Hz entre 0.5 ms y 2 ms. Esta variación se logra cambiando el valor de una variable dentro del programa, la cual se denomina ´pos´ a lo largo de este proyecto, permitiendo fijar la posición final del servomotor que se desea. La variable ´pos´ se variará entre 0 y 120, ya que es el rango de movimiento en grados del servomotor.

Para la programación del servomotor se utiliza el siguiente código que contiene los comentarios respectivos para su explicación.

```
#include <Servo.h>
Servo myservo;  // Se crea un objeto servo
int pos = 120;  // Variable para guardar la posición del servo
void setup()
{
  myservo.attach(9);  // La señal de control del servo se conecta al pin digital 9 de  PWM.

}
 void loop()
 {
    myservo.write(pos);    // El servo toma el valor de la variable 'pos' y se repite el ciclo.
 }
```

Para la programación de los motores DC se utiliza un código similar al anterior pero con la diferencia que en este caso, al variar la variable ´pos´, el motor no interpreta su valor como una posición sino un sentido y una velocidad específicos. El código implementado fue el siguiente:

```
#include <Servo.h>
 Servo myservo_2;    // Se crea el motor derecho
Servo myservo_3;   // Se crea el motor izquierdo
int pos_2 = 50;   // Variable para la velocidad del motor derecho
int pos_3 = 120;  // Variable para la velocidad del motor izquierdo
```

```
void setup()
{
  Myservo_2.attach(6);     // Motor derecho
  myservo_3.attach(11);    // Motor izquierdo
}

 void loop()
 {
    Myservo_2.write(pos_2);    // El motor derecho varia su velocidad con respecto al valor ´pos_2´
    myservo_3.write(pos_3);    // El motor izquierdo varia su velocidad con respecto al valor ´pos_3´
}
```

4.2.2 Sensores

El robot utiliza 3 tipos de sensores, 2 sensores de límite para indicar la posición límite inferior y superior de la cámara, un potenciómetro para obtener la posición exacta de la cámara y 2 encoders para tener la velocidad de los motores.

Sensor de Límite

Los sensores de límite tienen 3 conexiones, un cable rojo que no se conecta, uno negro el cual se conecta a tierra y uno blanco conectado a una resistencia de 100 kΩ a fuente de 5 V, y al pin digital correspondiente en la tarjeta Arduino UNO. Para los límites superior e inferior se utiliza el siguiente código:

```
#include <Servo.h>
 Servo myservo;   // Se crea un objeto servo
int pos = 50;     // Variable para guardar la posición del servo
int digitalPin=12;         // Se conecta el límite inferior al pin digital 12
int digitalPin2=7;          // Se conecta el límite superior al pin digital 12
int val=0;      // Se inicializa la variable val la cual entrega el valor del límite inferior
int val2=0;  // Se inicializa la variable val2 la cual entrega el valor del límite superior

void setup()
{
  myservo.attach(9);    // La señal de control del servo se conecta al pin digital 9 de  PWM.
```

```
  Serial.begin(9600);    // Se especifica la velocidad de la transmisión de datos en serie.
}
 void loop()
 {
   myservo.write(pos);    // El servo toma el valor de la variable 'pos'.
   val=digitalRead (digitalPin);     // Se lee el valor entregado por el pin
   Serial.println (val,2);     // Se muestra el valor leído en la ventana llamada ´serial monitor'
   val2=digitalRead (digitalPin2);   // Se lee el valor entregado por el pin
   Serial.print (val2,2);   // Se muestra el valor leído  en el ´serial monitor' .
}
```

Cuando se ha llegado a alguno de los límites, el valor de ´val´ pasa de ser 1 a ser 0 y se puede observar en la ventana denominada *serial monitor.*

Sensor de Posición de la cámara

El potenciómetro cumple la función de determinar la posición en la que se encuentra la cámara. Tiene 3 conexiones, un cable rojo que va conectado a una fuente de 5V, uno negro que se conecta a tierra y uno blanco conectado al pin análogo A1 de la tarjeta Arduino UNO. El código que lee el valor que entrega el potenciómetro y es el siguiente:

```
#include <Servo.h>
 Servo myservo;   // Se crea un objeto servo
int pos = 90;    // Variable para guardar la posición del servo
int analogPin=1;  // Se conecta el potenciómetro al pin análogo A1
int val=0;   // Se crea la variable la cual guarda el valor entregado por el potenciómetro
void setup()
{
  myservo.attach(9);   // La señal de control del servo se conecta al pin digital 9 de  PWM.
  Serial.begin(9600);     // Se especifica la velocidad de la transmisión de datos en serie.
}
  void loop()
 {
   myservo.write(pos);    // El servo toma el valor de la variable 'pos'.
   val=analogRead (analogPin);  // La variable ´val´ toma el valor leído en el pin analogo
```

Serial.println (val); // El valor de ´val´ se imprime en el serial monitor

}

Según la posición del servomotor, el potenciómetro toma un valor determinado el cual se puede visualizar en el *serial monitor*. Este valor se relaciona en grados con el rango de visión que cuenta la cámara, ya que el servomotor mueve la cámara y como el potenciómetro está conectado a él, mide la posición de la cámara.

Sensor de Velocidad

El encoder es un sensor que posee una rueda con 90 ranuras separadas equidistantemente, y a medida que la rueda gira, las ranuras permiten que un haz de luz llegue a un sensor que indica luminosidad y entre las ranuras, el rayo de luz es interrumpido, produciendo a la salida una señal cuadrada. Cada vez que pasa la luz por la ranura el sensor produce un 0, y cuando se encuentra en los intermedios es 1. Cada vuelta del encoder son 90 ranuras, es decir que se obtienen 90 pulsos a la salida por cada revolución.

Los encoders tienen 3 conexiones, un cable rojo que se conecta a una fuente de 5V, uno negro que se conecta a tierra y uno amarillo que se conecta a la señal de control correspondiente al pin digital de Arduino UNO que se especifique en el código, el cual se presenta en el Anexo F.

Para medir la velocidad se implementó un código que cuenta el número de pulsos que hay durante un segundo por medio de la función *millis* y establece el intervalo de tiempo de 1000 milisegundos (1 segundo) al inicio del código. Cada encoder se encuentra unido a un motor (motor derecho o motor izquierdo) y según la variable ´pos´ y ´pos_2´ los motores giran y los encoders calculan la velocidad de giro. Para el cálculo de la velocidad se necesita conocer el número de pulsos que hay durante un tiempo determinado (1 segundo), multiplicar este valor por 2π y luego dividirlo por 90, que es el número de pulsos que completan una vuelta del encoder, tal como se expresa en la ecuación 9.

$$Velocidad\ Encoder = \frac{\#\ Pulsos\ en\ 1\ segundo * 2 * \pi}{90}$$

Ecuación 9 Ecuación para hallar la velocidad del encoder.

La cantidad de pulsos se guarda en una variable llamada ´valor´ y para encontrar la velocidad angular de los encoders, se hace la operación matemática mencionada anteriormente y se almacena en la variable ´velocidad´ que es posible visualizar en el *serial monitor*.

4.2.3 Conexiones

Cada uno de los componentes tiene 3 cables que deben ser conectados ya sea a la alimentación, a tierra, o a la tarjeta Arduino UNO; este último recibiendo o enviando señales de control dependiendo del tipo de componente que sea (motor o sensor). En las especificaciones de cada componente (Anexo G) se encuentra la asignación de pines de cada uno por color.

Alimentación y Tierra

La tarjeta ArduinoUNO y los motores DC se alimentan de una batería de 9.6 V, mientras que los sensores y el servomotor se alimentan de una salida del microcontrolador que provee 5V. Para los límites, la alimentación se da por el Pin de control, con una resistencia entre el Pin y los 5V. En el caso de los motores DC, ya que el movimiento no va a ser constante, se pueden prsentar cambios bruscos que producen picos de corriente inversa que pueden afectar la batería. Con el objetivo de evitar este fenómeno, se utilizó un regulador de voltaje LM317 con los respectivos condensadores a la entrada y a la salida, asegurando el voltaje de salida cercano al de la entrada (voltaje de la batería) para el buen funcionamiento de los motores.

Entradas /Salidas de Control

Los motores DC y el servomotor reciben señales de control desde las salidas digitales PWM del Arduino UNO. Los limites y enconders envían señales de control por medio de entradas digitales, mientras que el potenciómetro envía sus datos por una entrada análoga. Al inicio del programa de Arduino UNO, se asignan cada una de las señales de control siguiendo la tabla 5.

COMPONENTE	SEÑAL DE ENTRADA O SALIDA	TIPO DE DATO	PIN SELECCIONADO
Servo Motor	Salida	PWM	Digital Pin 9
Motor DC Derecho	Salida	PWM	Digital Pin 6
Motor DC Izquierdo	Salida	PWM	Digital Pin 11
Limite Inferior	Entrada	Digital	Digital Pin 12
Limite Superior	Entrada	Digital	Digital Pin 7
Encoder Derecho	Entrada	Digital	Digital Pin 4
Encoder Izquierdo	Entrada	Digital	Digital Pin 2
Potenciómetro	Entrada	Análogo	Analog Pin 1

Tabla 5 Asignación de Pines por Componente

Se diseñó un circuito que permitiera de manera práctica las conexiones de los componentes con la fuente de alimentación y con la tarjeta de desarrollo, que consiste en 4 regletas tipo hembra, dos de 20 pines y dos de 10, ubicadas paralelamente, con las más largas hacia los extremos y las más cortas en el centro. Los componentes se conectan directamente en las regletas largas, donde en una se ubican los componentes del costado derecho del robot, y en la otra los del costado izquierdo. De cada una de las regletas del centro

sale un cable tipo ribbon que conectará el impreso con la tarjeta Arduino. El anexo G contiene el esquemático por componente, el circuito impreso y las respectivas conexiones.

4.3 Establecimiento de la Comunicación

La comunicación inalámbrica entre el computador y el robot consiste en dos partes; en el envío y recepción de la imagen, y en el intercambio del control del robot, el cual incluye tanto los datos para los motores (de la interfaz al robot) como la información que capturan los sensores (del robot a la interfaz), tal como lo ilustra la Figura 25.

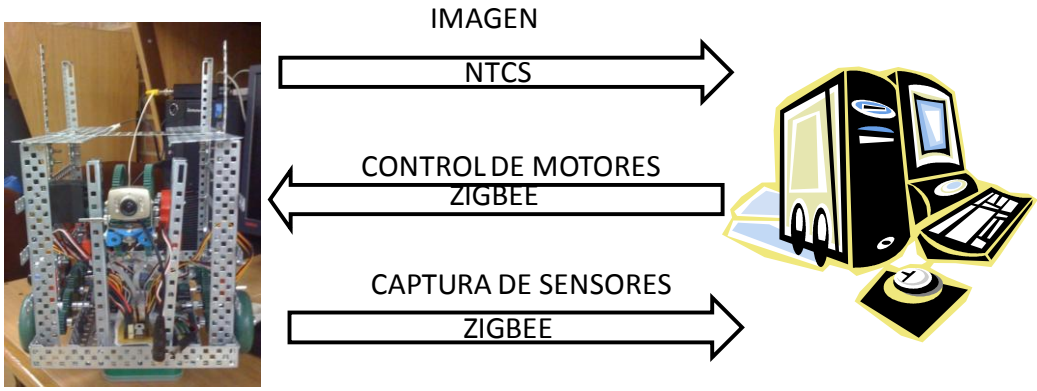


Figura 25 Comunicación entre el Robot y el Computador

4.3.1 Transmisión- Recepción de la Imagen

La transmisión- recepción de la imagen se logra gracias a la cámara inalámbrica que se utiliza, la cual contiene dentro de ella un transmisor a 2.4 GHz que envía la imagen análoga en formato NTSC directamente a un receptor. El receptor es alimentado con un adaptador de 12V, recibe la imagen por medio de una antena y su salida es por cable de video RCA. La cámara y el receptor están diseñados también para audio, pero para este proyecto no se utiliza.

La imagen pasa por un proceso de digitalización para poder ser manipulada en el computador, función que cumple la capturadora de video. La imagen entra por conexión BNC y sale por puerto tipo A/B, el cual lleva la imagen hasta el computador utilizando un puerto USB, que a su vez provee la alimentación necesaria al dispositivo. La Tabla 6 resume las características principales de la capturadora utilizada.

ESPECIFICACIONES CAPTURADORA SENSORAY 2250	
Entradas	Video: compuesto (BNC) - S- Video (DIN), 75 Ohm
	Audio: estéreo (2xRCA) o micrófono mono (3.5 mm Jack 3 Posiciones), 10kOhm
Formato de Video Entrante	NTSC, PAL
Formato de Video Saliente	MPEG1, MPEG2, MPEG4
Resolución de Salida	320X240, 720X480 (NTSC, 30 frames/seg) 320x288, 720x576 (PAL, 25 frames/sec)
Bitrates (numero de bits procesados por unidad de tiempo)	Constantes y Variables, hasta 6Mbs
Corriente por puerto serial	470mA
Comunicación hacia el receptor	USB 1.1 o 2.0

Tabla 6 Especificaciones de la Capturadora Sensoray 2250 [20]

Para obtener la imagen en el computador es necesario descargar el software de la pagina Web del producto (www.sensoray.com) e ingresar a Products/Video/ Frame Grabbers/USB Model 2250, donde se encuentra la descripción general, software y manuales, especificaciones, imágenes y precio de la capturadora. En la pestaña software & manuals se encuentra un link para la descarga del software, Model 2250/2251 Windows SDK o Model 2250/2251 Linux SDK que se selecciona dependiendo del sistema operativo utilizado.

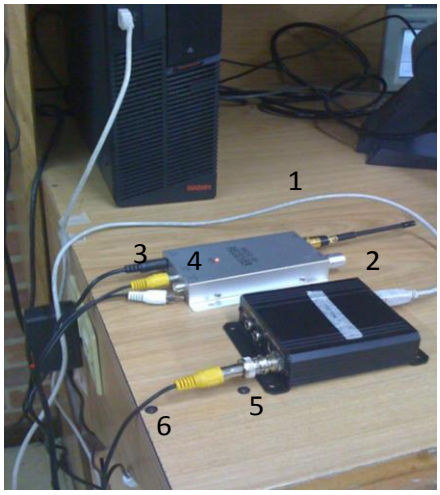
Automáticamente se descarga un archivo en WinZip que contiene un archivo ejecutable (2250setup.exe) que se abre para iniciar la instalación de los componentes del software y de una aplicación DEMO que viene incluida. Después se conecta la capturadora por USB, y en la ventana de "*Nuevo Hardware encontrado…*" que se abre, se selecciona la opción de instalar el Software Automáticamente. Adicionalmente, se debe instalar el ejecutable ffdshow-20051551.exe que está en la misma carpeta.

El DEMO que se incluye permite hacer una prueba para comprobar que efectivamente la imagen se tiene en el computador, simplemente oprimiendo el botón de PLAY azul. Si la capturadora no está correctamente conectada, aparece una ventana con el mensaje "*2250 not present. Please plug in device and restart application*". Si está bien conectada, se verá la imagen que captura la cámara.

En resumen, procedimiento para obtener la imagen con la mejor resolución es el siguiente:

1. Alimentar el receptor con un adaptador de 12V y asegurarse que el indicador de encendido (POWER) se encuentre iluminado. Conectar un extremo de un cable RCA al conector video out (figura 26).

2. El otro extremo del cable RCA conectarlo a la entrada VIDEO IN de la capturadora, utilizando un conector RCA-BNC. Por otro lado, con un cable tipo A/B- USB conectar la capturadora al computador (figura 26).



1. Cable tipo A/B- USB
2. Perilla TUNE del Receptor
3. Entrada DC 9-12 V del Receptor
4. Salida Video Out del Receptor
5. Entrada Video In de la Capturadora
6. Cable RCA-RCA

Figura 26 Conexiones entre capturadora y el receptor de video

3. Alimentar la cámara con una batería de 9V.
4. Abrir el DEMO de la capturadora, oprimir el triangulo azul de la parte superior, y sintonizar la imagen con la perilla TUNE del receptor. Si la imagen está borrosa, se calibra girando el lente hasta que la imagen se observe completamente nítida (figura 27).

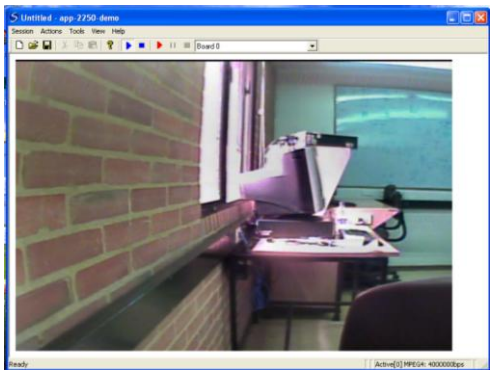


Figura 28 Imagen de la cámara desde el DEMO de Sensoray

Figura 27 Imagen de la cámara desde Matlab

5. Como la imagen se necesita en Matlab®, se corre el siguiente código que identifica la capturadora como un dispositivo de video, y despliega el video que captura la cámara en el monitor, como se observa en la figura 28.

```
Imaqreset
imaqhwinfo('winvideo')
camara=videoinput('winvideo');
preview(cámara)
```

4.3.2 Transmisión- Recepción de los Controles

Los datos que permiten la interacción del usuario con el robot son los referentes a los valores de control de los componentes, tales como los valores de velocidad de los dos motores DC, el valor de posición del servomotor, y la información que captura cada uno de los sensores. La transmisión- recepción de estos datos se realiza inalámbricamente utilizando las tarjetas XBee que transmiten la información de manera digital, y de forma serial.

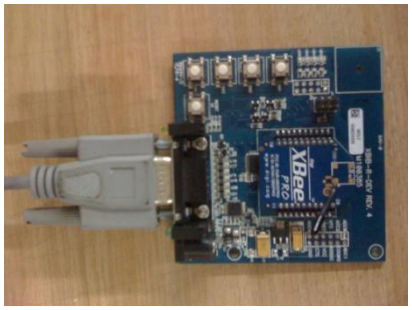


Figura 29 Tarjeta Maxstream con XBee

Los dispositivos que permiten este flujo de información desde el computador es una interfaz Maxstream (figura 29) lo que conecta directamente la XBee por el puerto de entrada serial RS-232. Desde el robot se usa una tarjeta XBee Shield (figura 30) que permite las conexiones entre la XBee y la tarjeta Arduino UNO. Con la XBee shield conectada sobre la Arduino, se puede trabajar en el microcontrolador ya sea por USB (conexión directa con el computador) o en modo XBee. Esto se controla manipulando unos jumpers que se encuentran sobre la Shield.

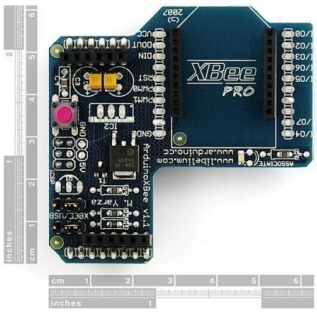


Figura 30 Tarjeta XBee Shield de Arduino

Para lograr comunicación punto a punto entre dos Tarjetas XBee, primero se deben configurar en cada una la dirección propia (MY) y la dirección de destino (DL), que en este caso será la dirección propia de la otra tarjeta. Esta configuración se realiza, para las dos XBee, desde la tarjeta Maxstream utilizando el programa X-CTU.

X-CTU es un software de configuración y pruebas diseñado para interactuar con los archivos de los productos DiGi de RF y proveer una interfaz gráfica para los usuarios. El programa se encuentra gratis en la página de Digi Key [28].

La interfaz de X-CTU consiste en 4 pestañas ubicadas en la parte superior, tal como se puede observar en la figura 31.

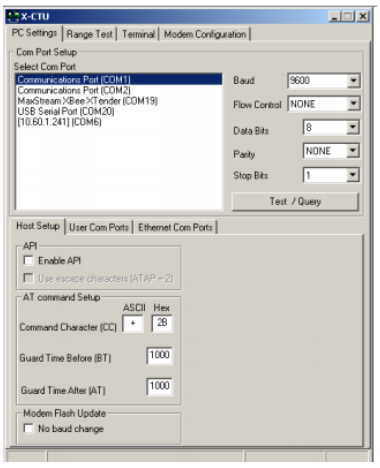


Figura 31 Interfaz Inicial del Programa X-CTU

En la pestaña PC Settings se configuran las características de acceso de los puertos existentes o puertos creados por el usuario, como la velocidad de transmisión, el control, el tamaño de datos, la paridad y bits de salida. Al conectar la tarjeta Maxstream, automáticamente es identificada por el software, y oprimiendo Test/Query se debe obtener un mensaje confirmando la comunicación.

Teniendo la comunicación establecida, en la pestaña Modem Configuration se definieron las características de las tarjetas XBee de la siguiente manera:

a) Primero se deben leer los parámetros que trae la XBee, oprimiendo "Read" con la casilla de "Always Update Firmware".
b) Si no es posible leerlos, se deben descargar las últimas versiones desde la opción "Download New Versions…" y seleccionar la opción de Restore.
c) Cuando aparezcan los parámetros de la XBee, se editan como se requiera, se desactiva la casilla de "Always Update Firmware" y se guardan oprimiendo WRITE. La figura 32 muestra la ventana donde se editan los parámetros de la XBee.

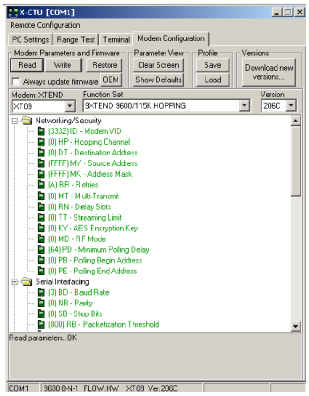


Figura 32 MODEM CONFIGURATION-Ventana de Actualización de Parámetros

Los valores que se utilizaron para la configuración de las XBee se resumen en la tabla 7.

DEFINICIÓN DE PARÁMETROS DE LAS XBEE		
Parámetros	Xbee 1 (computador)	Xbee 2 (arduino)
CHANNEL	C	
PAN ID	3332	
MY ADDRESS	3BA2	CC11
DT DESTINATION ADDRESS	CC11	3BA2

Tabla 7 Definición de Parámetros de la XBee

Con la configuración completa de los parámetros de las XBee, se programa el envío y recepción de datos seriales en el microcontrolador y en Matlab®. Para lograr una comunicación exitosa entre estos, es importante conocer el orden de los datos que se envían y se reciben, ya que estos no tienen ningún nombre en el momento del envío o recepción, por lo cual va a depender siempre del orden en el que se encuentren en el programa. La tabla 8 resume los datos que se envían y se reciben en la tarjeta Arduino UNO y en Matlab®.

DATOS	**ARDUINO**	**MATLAB®**
DATOS QUE SE ENVÍAN	Velocidad de Encoder Derecho	Control servomotor
	Velocidad de Encoder Izquierdo	Control Motor DC Derecho
	Valor Potenciómetro	Control Motor DC Izquierdo
	Valor Limite Inferior	
	Valor Limite Superior	
DATOS QUE SE RECIBEN	Control servomotor	Velocidad de Encoder Derecho
	Control Motor DC Derecho	Velocidad de Encoder Izquierdo
	Control Motor DC Izquierdo	Valor Potenciómetro
		Valor Limite Inferior
		Valor Limite Superior

Tabla 8 Comparativo de Datos de Envío y Recepción de Arduino y Matlab®

Arduino

Para recibir o enviar datos desde Arduino UNO se abre el puerto serial con la función Serial.Begin() especificando la velocidad de transmisión dentro del paréntesis, y comprobando su disponibilidad con la función serial.available(). Si lo está, se leen o se envían los datos utilizando las funciones Serial.read() o

Serial.print (). El código presentado en el Anexo H recibe los valores de control de los 3 motores del robot por medio del puerto serial, que provienen de la XBee. Adicionalmente, presenta en el Serial Monitor los valores que le están llegando.

Para enviar datos por el puerto serial se utilizan las funciones para abrir el puerto y comprobar su disponibilidad, pero para enviar los datos se iguala la función Serial.print() a la variable que contenga el valor a transmitir. El anexo I contiene el programa que se diseño para enviar la información de los limites y del potenciómetro de manera serial, donde se lee el valor de los pines respectivos, y con la función Serial.println() se van enviando uno a uno de forma separada.

Matlab®

Matlab® tiene la capacidad de enviar y recibir información por puerto serial siguiendo un procedimiento parecido al que se plantea en la tarjeta de desarolllo, donde existe una primera etapa para abrir el puerto y fijar parámetros como la velocidad de transmisión, bits de parada, paridad, y tamaño de datos de entrada y de salida entre otros, y utilizando funciones como fwrite (para transmitir) o fscanf (para recibir), se almacenan en variables que después se pueden manipular en el programa.

En el envío de datos, la función fwrite especifica el puerto por el que se envía el dato y el tipo de dato, que en el caso del proyecto corresponde a caracter tipo uchar. Además, se debe tener en cuenta que es necesario enviar un primer byte que indique que la comunicación ha iniciado, para poder enviar uno a uno los datos que se deseen transmitir. Este orden debe coincidir con el orden de valores de recepción que se tiene en Arduino UNO.

La función fscanf cumple con la tarea de almacenar en una variable un tipo de dato proveniente de un puerto específico, información que se detalla dentro del paréntesis de la siguiente manera:

Ejemplo= fscanf(‘nombre puerto’, ‘tipo de dato’)

El Anexo I contiene el programa que utiliza este proyecto, donde se envían los valores de los motores, y se almacena la información de los sensores en una matriz, por razones de facilidad a la hora de la administración de la información.

4.4 Diseño de la Interfaz

La interfaz está diseñada con la función en Matlab® llamada GUIDE, la cual facilita la realización de plataformas que permitan la interacción del usuario con el programa de manera dinámica. Por medio de botones, texto, menús y planos, es posible realizar cualquier tipo de programa que intercambie información con el usuario, creando un código para cada componente de la interfaz que contiene su función.

Al utilizar Guide, se abre una ventana que permite la integración física de los componentes que se requieran, arrastrándolos uno a uno hasta la posición deseada. Al guardar, Matlab® crea dos archivos; el primero es tipo .fig que representa la interfaz como tal, y el segundo un archivo .m que contiene el código de cada una de las partes de la interfaz.

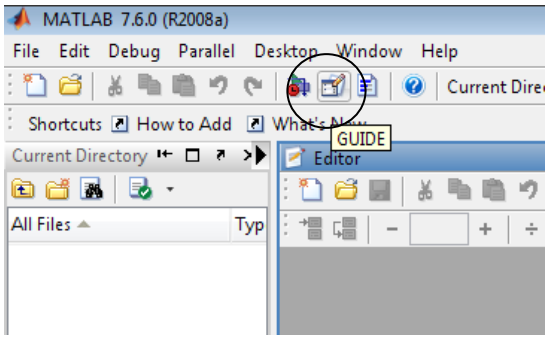


Figura 34 Selección de Función GUIDE en MatLab

La interfaz de este proyecto consiste en 2 partes; la primera es la adquisición y presentación de la imagen capturada por la cámara y la segunda el envió y recepción de los controles del robot así como la visualización de los mismos.

Para iniciar el diseño de la interfaz, se selecciona la función de GUIDE, tal como lo presenta la figura 33. Se crea un proyecto en blanco y en la ventana de la figura 34, se van seleccionando las partes que se requieran, que se encuentran en el lado izquierdo de la ventana a la cuadricula. En el archivo .m se crea una función general para toda la interfaz, y una función para algunos de sus componentes. Una característica importante que se debe tener en cuenta en el momento de la programación, es que las variables en GUIDE se declaran precedidas de un handles.variable, y a continuación se debe escribir la línea guidata (hObjects, handles) que almacena el valor de la variables.

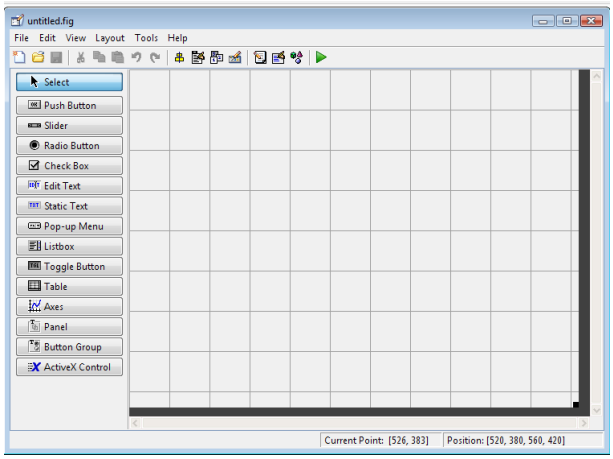


Figura 33 Ventana de Guide Matlab en blanco

4.4.1 Adquisición y Presentación de la Imagen

La ventana donde se observa el video de la cámara se realiza con el componente Axis de Guide, donde el cuadrado que representa se configurará de tal manera que mostrara la imagen del video. Como el axis no presenta función en el archivo .m, en la función general de la interfaz se asigna la ventana como entrada de video.

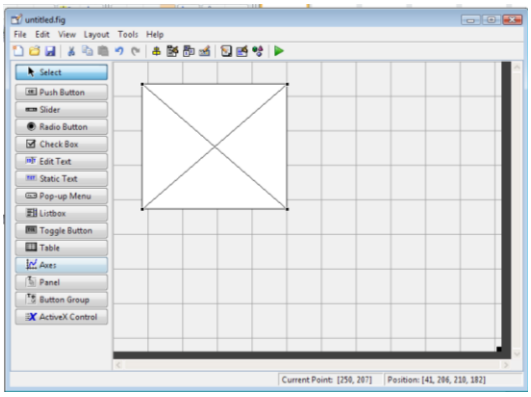


Figura 35 Axis de Matlab Guide

4.4.2 Envío y recepción de los Controles

El envío de controles se maneja gradualmente utilizando botones que aumentan o disminuyen esta señal de control. En el caso de la cámara, la señal de control hace que el movimiento sea hacia arriba o hacia abajo, mientras que en los motores DC, la señal de control aumenta o disminuye la velocidad de estos, hacia un sentido específico. Para ello se utilizan 3 pares de botones, un par para cada motor, programados de tal forma que la variable que se envía aumenta o disminuye a medida que se oprime el botón. Adicionalmente, dentro de la función de cada uno de estos componentes, se incluye el programa de envío y recepción de datos que se presenta en el Anexo J.

Figura 36 Botones de Control de Posición de la Cámara

El código de los botones es un *if* que varía la señal de control mientras se encuentre entre los límites permitidos, aumentando o disminuyendo el valor inicial del control. Las líneas que se exponen a continuación son el código de botón Abajo que se observa en la Figura 36 que mueve la cámara en esa dirección, limitado por el valor que representa la posición límite inferior. En el caso del botón Arriba, el sumando se resta y la función se limita con el valor límite superior, como se describirá más detalladamente en el análisis de resultados.

```
sumando=5;
handles.posicion1=handles.posicion + sumando;
if (handles.posicion1 < 135)
  handles.posicion = handles.posicion1;
else
    handles.posicion1=135;
end
guidata(hObject, handles);
```

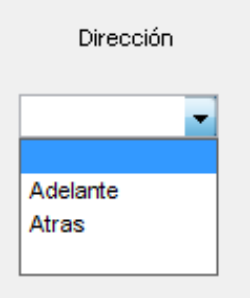


Figura 37 Menú de Sentido de Movimiento

En el caso de la velocidad, la señal de control se limita en un máximo y en un mínimo, ya que dentro del rango de valores de control del motor, la mitad corresponden a un sentido y la otra mitad al otro. Los valores límites dependen del sentido que se esté moviendo, el cual se fija en un menú como el que se observa en la Figura 37. El código que se plantea a continuación, corresponde a la función del botón "Aumentar Velocidad del Motor Derecho" (Figura 38) hacia adelante, donde handles.motorderecho1 es la señal de control que se envía, y handles.adelante es una variable que indica la decisión del usuario en el menú.

```
if (handles.adelante==1)
sumando=3;
handles.motorderecho1=handles.motorderecho + sumando;
 if (handles.motorderecho1 < 140) % Limita Valor Máximo
  handles.motorderecho =
handles.motorderecho1;
else
    handles.motorderecho1=140;
end
 if (handles.motorderecho1 > 90) %Limita Valor
Mínimo
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=90;
end
```

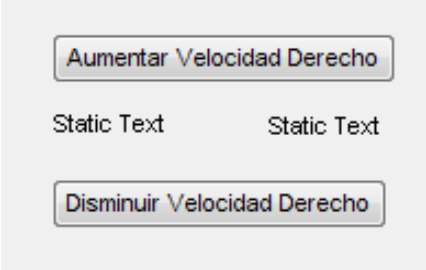


Figura 38 Botones de Velocidad de Motor Derecho

Los valores de control que se deben enviar se establecieron después de realizar las caracterizaciones respectivas de cada uno de los componentes, como se detallará más adelante.

4.5 Modelado del robot

El modelo dinámico del robot está basado en el modelo de un monociclo que se rige por las ecuaciones 5 y 8 del marco teórico, que se exponen a continuación:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Ecuación 5

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{T} & -\frac{r}{T} \end{bmatrix} \begin{bmatrix} \omega_{derecho} \\ \omega_{izquierdo} \end{bmatrix}$$

Ecuación 8

Donde $\dot{x}_1$ es la velocidad en el eje X_1, $\dot{x}_2$ es la velocidad en el X_2, $\dot{\theta} = \omega$ es la velocidad angular global del robot, v es la velocidad lineal, θ el ángulo de desviación durante la trayectoria, r es el radio de las llantas, T la distancia entre ellas, $\omega_{derecho}$ la velocidad angular de las llantas del costado derecho y $\omega_{izquierdo}$ la velocidad angular de las llantas del costado izquierdo.

La validación del modelo consiste en fijar una velocidad angular a cada costado y comparar los valores teóricos de la salida del sistema matricial con los valores prácticos que se obtengan después de una serie de pruebas. El protocolo de pruebas utilizado para llegar a esto fue el siguiente:

a) Como primera instancia, se debe relacionar la velocidad angular de las llantas con la velocidad angular del motor, pues al estar unidos por medio de un sistema de engranajes, se produce una relación lineal entre ellas multiplicada por un factor que depende del número de dientes de los mismos. Ese factor k se incluye en la ecuación 8 de la siguiente manera.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{T} & -\frac{r}{T} \end{bmatrix} \begin{bmatrix} k * \omega_{encoderderecho} \\ k * \omega_{encoderizquierdo} \end{bmatrix}$$

Ecuación 10 Matriz del modelo 1: Velocidad Lineal y Velocidad Angular como salidas.

Reemplazando los valores del radio de las llantas r=0.0349 metros y la distancia entre ellas T = 0.23 metros del robot, se reduce a un sistema de dos incógnitas que representan las velocidades angulares de cada costado.

b) Después, se asigna un valor aleatorio de control a los motores DC que produzca un desplazamiento y se registra el valor de la velocidad angular de cada uno, medido por el encoder respectivo. El valor teórico de v y ω se obtiene ingresando el valor de estas velocidades angulares en el sistema matricial.

El valor práctico de v se obtiene midiendo la distancia lineal recorrida por el carro en un tiempo determinado. El valor de ω se obtiene haciendo girar el robot sobre su propio eje, y midiendo el ángulo de giro durante el tiempo que dure la prueba. Si el modelo es correcto, este valor debe ser cercano.

c) Reemplazando los valores teóricos de v y ω en la ecuación 5, se obtiene el valor teórico de $\dot{x}_1$, $\dot{x}_2$, $\dot{\theta}$. Para encontrar $\dot{x}_1$ y $\dot{x}_2$ en la práctica, se debe medir la distancia tanto en el eje X_1 (horizontal) como en el eje X_2 (vertical) que recorrió el robot durante un tiempo determinado, ya

que son valores de velocidad. Por otro lado, $\dot{\theta}$ es igual a la velocidad angular global hallada en el inciso b.

Las pruebas fueron realizadas colocando el robot en el origen de un plano cartesiano como lo ilustra la figura 39, y enviando una velocidad fija a ambos motores durante un tiempo. La velocidad lineal es calculada dividiendo la distancia del segmento 1 de la figura durante el tiempo de la trayectoria. El ángulo de desviación se mide en la posición final, midiendo el giro respecto al plano inicial, tal como se observa en el punto 4. Finalmente las velocidades en los ejes se hallan de la misma manera, pero tomando para $\dot{x}_1$ la distancia 2 y para $\dot{x}_2$ la distancia 3.

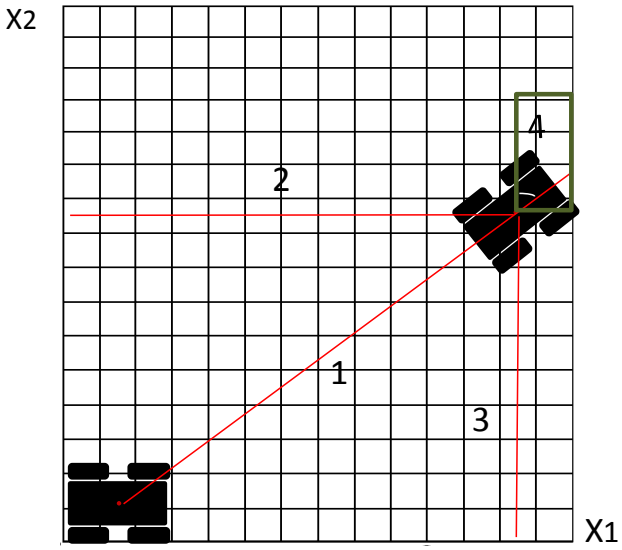


Figura 39 Representación gráfica de las pruebas

5 ANÁLISIS DE RESULTADOS

La función utilizada en Arduino UNO a lo largo del desarrollo del proyecto, es la función *myservo*, la cual produce a la salida una señal cuadrada de periodo fijo y ciclo útil variable, llamada señal PWM, que depende de un valor asignado en el programa. Aunque está función fue diseñada por Arduino para controlar el movimiento de un servomotor, también se utilizó para los motores DC. Para entender su funcionamiento, se desarrolló un código de prueba para un motor sin carga y se observó en el osciloscopio lo que entrega la señal de control al ir modificando el valor de una variable llamada 'pos'. El código de prueba es el siguiente:

```
#include <Servo.h>
 Servo myservo;   // Se crea el objeto servomotor
int pos = 120;    // Variable para guardar la posición del servo

void setup()
{
  myservo.attach(9);    // La señal de control del servo se conecta al pin digital 9 de  PWM.
  }

 void loop()
 {
    myservo.write(pos);     // El servo toma el valor de la variable 'pos'.
}
```

Lo observado en el osciloscopio fue que al variar la variable ´pos´, el ciclo útil de la señal cambiaba, la tabla con los valores obtenidos y su gráfica se pueden ver en el Anexo K.

5.1 Servomotor, límites y potenciómetro

Para el movimiento de la cámara se diseñó un sistema de orientación, que consta de un servomotor encargado de mover la cámara, un potenciómetro que mide los valores de la posición y 2 sensores de límite para indicar que la cámara llegó a su máximo movimiento hacia arriba o hacia abajo.

La caracterización del servomotor se realizó junto con el potenciómetro implementando el código anterior, y modificando la variable ´pos´ de forma discreta fijándole un valor grande seguido de un valor pequeño. Se realizaron 10 pruebas y de estas se calculó el promedio y los resultados se muestran en el Anexo L. Como complemento se realizó otra prueba en la cual la variable ´pos´ se cambiaba de forma gradual de a

5 unidades. Analizando los valores de la tabla M1 del Anexo M se noto que los valores eran diferentes cuando el cambio era creciente (desde 0 hasta 130) a cuando la variación era decreciente. Esto se debe al fenómeno de histéresis que presentan los instrumentos de medida como se observa en la grafica M1 del mismo anexo. Se realizó una linealización en cada uno de los casos obteniendo su respectiva ecuación y utilizándola para calcular la posición de la cámara en la interfaz.

5.2 Motores DC y encoders

Para el movimiento del robot se utilizaron dos motores DC cada uno conectado a un encoder para obtener su velocidad. La prueba que se realizó para caracterizar los motores consistió en modificar la variable ´pos´ para cada uno de ellos (derecho e izquierdo) y con el código realizado en Arduino, se obtiene el valor del conteo de pulsos en un segundo. La velocidad angular del encoder se encuentra por medio de la ecuación 9. Se realizaron varias pruebas para así tener un valor promedio para cada dato, estas pruebas se encuentran en el Anexo N.

La tabla 9 muestra los valores promedio de las velocidades de cada motor entregadas por el encoder respectivo, según el valor de la variable ´pos´. Estas pruebas fueron hechas sin carga. Se puede ver que al tener el mismo valor de ´pos´, la velocidad no es la misma para los dos motores.

Las pruebas con carga del motor no pudieron ser realizadas ya que la velocidad de un motor afecta la velocidad del otro. Si la velocidad es menor en un lado, este afectará el otro lado y viceversa. Lo anterior puede ser observado en la tabla 10 en la cual se muestra que fijando un valor de ´pos´ para un motor y variando los valores para el otro, las velocidades cambian.

Se realizó también la prueba de corrientes para saber cuánta corriente pedía cada motor, pero se realizó esta prueba sin carga ya que con carga no fue posible. Los resultados obtenidos se muestran en el Anexo O, y de estos se observó que la corriente entregada por las baterías utilizadas es suficiente para los motores.

POS	ω ENCODER DERECHO (rad/s)	ω ENCODER IZQUIERDO (rad/s)
45	8,288	11,022
48	8,232	10,806
51	7,884	10,582
54	8,064	10,4
57	8,148	10,05
60	7,064	9,686
63	6,728	9,252
66	6,448	8,776
69	6,154	8,158
72	5,79	7,4
75	4,676	6,378
78	3,882	5,146
81	2,748	3,854
84	1,316	2,034
87	quieto	Quieto
90	quieto	Quieto
93	quieto	Quieto
96	quieto	Quieto
99	quieto	Quieto
102	quieto	Quieto
105	0,728	1,314
108	2,174	3,252
111	3,518	4,716
114	4,162	6,056
117	5,106	7,15
120	5,65	7,898
123	6,112	8,636
126	7,216	9,154
129	7,218	9,572
132	7,648	9,924
135	7,644	10,162
138	7,51	10,372
141	8,204	10,806
144	8,314	11,202
147	8,052	11,244
150	8,356	11,356

Tabla 9 Valores de velocidad con respecto a la variable pos

Prueba 1			Prueba 2			Prueba 3		
Motor	Derecho	Izquierdo	Motor	Derecho	Izquierdo	Motor	Derecho	Izquierdo
Pos	111	81	Pos	123	81	Pos	130	81
Velocidad	1,54	1,54	Velocidad	2,37	2,51	Velocidad	2,58	3,7
	1,33	2,37		2,44	3,63		3,14	5,09
	1,95	2,3		3,28	3,91		4,05	5,09
	1,47	2,37		3,14	3,98		4,19	5,3
	1,4	2,58		3,07	4,19		4,26	5,3
	1,54	2,37		2,3	3,07		4,26	5,3
Promedio	1,538333333	2,255	Promedio	2,76666667	3,54833333	Promedio	3,74666667	4,96333333

Tabla 10 Valores de velocidad con carga

Además los motores tienen diferente sentido de giro con respecto a la variable ´pos´ ya que el motor derecho gira hacia adelante con un valor determinado y el motor izquierdo gira hacia atrás con este mismo valor, lo cual se muestra en la tabla 11. El anexo P presenta el código final de Arduino utilizado en el proyecto.

Valores de Pos Motor Derecha		Valores de Pos Motor izquierda
127	Adelante	55
122	Adelante	60
117	Adelante	65
112	Adelante	70
107	Adelante	75
90	QUIETO	90
85	Atrás	110
81	Atrás	115
77	Atrás	120
73	Atrás	125
69	Atrás	130

Tabla 11 Sentido de giro de cada motor según el valor de ´pos´

5.3 Caracterización del Robot

La caracterización del robot completo consiste en analizar su comportamiento mediante relaciones de fuerza, torque y aceleración; valores dependientes de las especificaciones de los componentes que permiten el desplazamiento del mismo.

5.3.1 Razón de Transmisión

El sistema de movimiento del robot, en cada costado, consiste en 5 engranajes, tal como se observa en la Figura 40, donde el central es el más grande (60 dientes) y va conectado directamente al motor DC del costado respectivo. A su lado, hay dos engranajes seguidos, donde el más cercano es de 36 dientes y el siguiente (engranajes de los extremos) de 24 dientes, estos últimos directamente conectados a las llantas.

La velocidad angular de los engranajes cumple la relación $\frac{\omega_1}{\omega_2} = -\frac{Z_2}{Z_1}$ donde ω es la velocidad angular y z el número de dientes de los engranajes. El signo indica que el sentido es contrario al del engranaje continuo. Según lo anterior, se obtiene que la razón de la velocidad del motor cumple la ecuación 11.

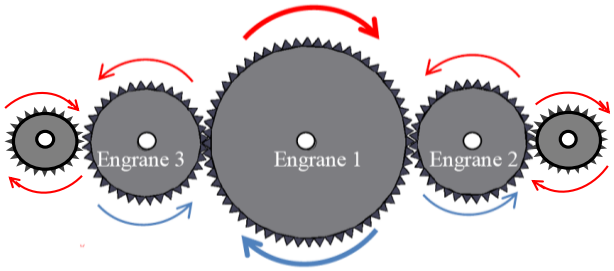


Figura 40 Engranajes del robot

$$\omega_{llantas} = 2.5 * \omega_{motordc}$$

Ecuación 11 Ecuación de razón de transmisión

5.3.2 Torque

El torque de las llantas viene dado por la ecuación 12, y se utiliza para calcular la fuerza del motor.

$$Torque\ Llantas = \frac{Torque\ del\ Motor * Relación\ de\ Transmisión}{2}$$

Ecuación 12 Calculo de la torque de las llantas

El torque del motor se obtiene de las especificaciones del mismo (Anexo A) y la razón de transmisión para este proyecto es de 2.5.

$$Torque\ Llantas = \frac{0.7345\ Nm * 2.5}{2} = 0.918\ Nm$$

5.3.3 Fuerza del Motor

La fuerza del motor se calcula utilizando la ecuación 10, donde el radio de las llantas se halla dentro de las especificaciones del mismo.

$$Fuerza\ del\ Motor = \frac{Torque\ Llantas}{Radio\ Llantas} = \frac{1.836\ Nm}{0.0349\ m} = 52.60\ N$$

Ecuación 13 Ecuación para hallar la fuerza del motor

5.3.4 Aceleración

La aceleración del robot se calcula con la ecuación 14 donde se duplica la fuerza porque el robot se mueve debido a los dos motores.

$$2 * F = m * a; a = \frac{2 * F}{m}$$

Ecuación 14 Ecuación para calcular la aceleración

$$a = \frac{2 * 52.60}{3.5\ kg} = 30.05\ m/s^2$$

5.4 Modelado del Robot

El modelado del robot consiste en utilizar las ecuaciones matriciales del modelo de un monociclo, y comparar los valores teóricos y prácticos bajo las mismas condiciones. Las velocidades angulares de cada costado constituyen entradas tanto para el sistema matricial teórico como para las pruebas, y se obtienen gracias a la lectura de los sensores de velocidad. Para obtener una medida más aproximada se calculó el promedio de varios datos de la velocidad angular de los motores enviada por el encoder.

5.4.1 Valores Teóricos

Los valores teóricos se obtuvieron con un programa diseñado en Matlab® (Anexo Q), que recibe las velocidades angulares de los motores de cada costado, el tiempo de la prueba, y el ángulo de desviación, y calcula la velocidad lineal, la velocidad en x_1 $(\dot{x}_1)$, la velocidad en x_2 $(\dot{x}_2)$, y la velocidad angular ω.

5.4.2 Valores Prácticos

Cada una de las pruebas consiste en realizar el siguiente procedimiento, registrando las mediciones indicadas.

- Ubicar el robot en el origen de un plano cartesiano y tomar una medida de la ubicación del centro del carro en la posición inicial. El plano cartesiano utilizado consiste en 3 pliegos de papel periódico unidos, con los ejes marcados cada 10 cm.

- Fijar una velocidad angular a cada motor y mantenerla constante durante un tiempo, valores que se deben registrar.
- Tomar una medida de la ubicación del centro del robot en la posición final. Con esta medida es posible hallar la distancia recorrida en el eje X_1, la distancia recorrida en el eje X_2, y la distancia total, respecto a la inicial.
- Finalmente, medir el ángulo de desviación de la posición final respecto al plano que se encontraba en la posición inicial.

Al final por cada prueba se deben obtener los siguientes parámetros.

PARÁMETROS MEDIDOS DURANTE LA PRUEBA
Distancia Recorrida
Distancia Recorrida en X_1
Distancia Recorrida en X_2
Ángulo de Desviación
Tiempo

Tabla 12 Parámetros obtenidos durante cada prueba

Con los valores de la tabla 12 se calculan las variables de salida del sistema de la siguiente manera:

VARIABLES DE SALIDA	
Velocidad Lineal	$\frac{Distancia\ Real\ Recorrida}{Tiempo}$
Velocidad en X_1	$\frac{Distancia\ Recorrida\ en\ X_1}{Tiempo}$
Velocidad en X_2	$\frac{Distancia\ Recorrida\ en\ X_2}{Tiempo}$

Tabla 13 Cálculo de las variables de salida en base a los parámetros medidos

5.4.3 Validación del Modelo

La tabla 14 compara los resultados teóricos y prácticos de las pruebas realizadas, que consistían en hacer las mediciones de distancia, tiempo y ángulo sobre el plano para 4 casos; hacia adelante en línea recta, hacia atrás en línea recta, hacia adelante con desviación a la derecha y hacia adelante con desviación hacia la izquierda. Para cada uno de los casos, se realizaron dos pruebas y se compararon los resultados con los valores del modelo, como se plantea en la tabla 14. El registro completo de las dos pruebas para cada caso se encuentra en el anexo R.

CASOS		CASO 1	CASO 2	CASO 3	CASO 4
Descripción		Desplazamiento hacia adelante en línea recta.	Desplazamiento hacia atrás en línea recta.	Desplazamiento hacia adelante con desviación a la derecha	Desplazamiento hacia adelante con desviación a la izquierda
Velocidad Lineal (v [m/s])	Teóricos	0,1143	0,0825	0,5032	0,2391
	Prácticos	0,1164	0,0978	0,5084	0,2683
Velocidad en X1 (x1 [m/s])	Teóricos	0,1143	0,0825	0,4771	0,2374
	Prácticos	0,1164	0,0978	0,5042	0,268
Veloxidad en X2 (x2 [m/s])	Teóricos	0	0	0,16	0,0279
	Prácticos	0	0	0,068	0,015

Tabla 14 Registro de los valores teóricos y prácticos para cada caso.

La tabla 15 muestra los errores porcentuales de los parámetros de salida para cada caso, donde se puede observar que la velocidad en X_2 es la que presenta mayor discrepancia entre los cálculos teóricos y los prácticos, mientras que la velocidad lineal es la que presenta menores porcentajes de error. Por otro lado se observa que el robot presenta mayor semejanza al modelo cuando se mueve hacia adelante comparado con el movimiento hacia atrás, mientras que respecto a los movimientos laterales, este se cumple con desviaciones hacia la derecha.

CASOS	CASO 1	CASO 2	CASO 3	CASO 4
Descripción	Desplazamiento hacia adelante en línea recta.	Desplazamiento hacia atrás en línea recta.	Desplazamiento hacia adelante con desviación a la derecha	Desplazamiento hacia adelante con desviación a la izquierda
Velocidad Lineal (v [m/s])	1,80%	18.50%	1%	12,20%
Velocidad en X_1 (X_1 [m/s])	1,80%	18.50%	6%	12,88%
Veloxidad en X_2 (X_2 [m/s])	0%	0%	57,50%	46,23%

Tabla 15 Errores porcentuales entre valores teóricos y prácticos para cada caso.

El parámetro ω que también representa una salida del modelo no es posible validarlo ya que por el tipo de llantas y la forma de construcción del robot, su máximo ángulo de giro es muy pequeño. Se hicieron pruebas ajustando la velocidad de un motor en 0 y el otro en su máximo, y el carro adquiría un movimiento lineal, ya que el motor con velocidad impulsa las llantas del otro el motor así éste se encuentre quieto. La forma de conexión del sistema rotacional puede ser la explicación de este comportamiento, lo que se puede solucionar conectando los motores directamente a las llantas traseras y las llantas delanteras dejarlas libres. Por lo anterior, fue imposible medir el ángulo para calcular la velocidad angular total.

5.5 Diseño de la interfaz

La interfaz diseñada está dividida en tres partes, en la primera se adquiere la imagen transmitida por la cámara y se visualiza, en la segunda se manipula el robot en general, y en la tercera se controlan las partes independientes del carro. Esta última tiene a su vez tiene 3 divisiones, que son cámara, costado derecho y costado izquierdo, y consiste en enviar las señales de control a cada motor, recibir y visualizar la información captada por los sensores. La Figura 41 muestra la interfaz.

Figura 41 Interfaz

5.5.1 Parte I: Visualización de la imagen

La visualización de la imagen no requiere análisis pues mientras ésta se encuentre de forma digital en el computador, la función de Matlab® de adquisición de video funciona perfectamente y no es necesario hacer modificaciones. La clave para que la visualización sea buena es la sintonización correcta del receptor de video y las conexiones entre el receptor, la capturadora y el computador.

5.5.2 Parte II: Manipulación General del Robot

Esta parte de la interfaz contiene un menú donde es posible seleccionar si se desea el movimiento hacia adelante o hacia atrás, un botón de parada, uno para actualizar y uno para salir. Estos afectan el movimiento total del robot.

Cuando se selecciona alguna de las opciones del menú, ya sea adelante o atrás, el programa cambia el rango de los valores de control que se envían al aumentar o disminuir la velocidad, en base a la información suministrada por la tabla 11.

El botón de parar, envía a los motores DC una señal que indica que deben permanecer estáticos, bloqueando cualquier otra orden que esté cumpliendo, funcionando como un parado de emergencia. Por otro lado, cada vez que se oprime el botón actualizar, los valores de los sensores que se visualizan en la interfaz se actualizan. Finalmente, el botón salir cierra la interfaz.

5.5.3 Parte III: Control Independiente de las Partes

Esta última parte contiene la información referente a la cámara, el movimiento del servomotor y la lectura e interpretación de los valores de los sensores de límite y del potenciómetro. En esta sección hay dos botones, uno que mueve la cámara hacia arriba y el otro hacia abajo, la ampliación de esta información se puede apreciar en el Anexo M. Los sensores de límite envían un 0 cuando llega al límite y un 1 cuando la cámara está dentro de su rango permitido de movimiento. Se utilizó dentro de la función de los botones de movimiento, un código que al recibir 0 indique en la interfaz que la cámara llegó al límite en forma de mensaje. El potenciómetro por su parte, envía un valor de resistencia que según la tabla del Anexo M se debe interpretar y visualizar como los grados entre la posición más baja y la posición actual. El rango de movimiento es de 120°, siendo 0° la posición en el límite inferior. Esto se logra dentro del código utilizando las ecuaciones de linealización de la gráfica de esta caracterización particular.

Las secciones de costado derecho e izquierdo contienen dos botones que aumentan o disminuyen la velocidad angular de cada motor, y un espacio donde se visualiza la velocidad angular de las llantas. El código de los botones de cambio de velocidad, primero verifica el sentido al que se desea mover el robot, y dependiendo de esto aumenta o disminuye la señal de control basados en la tabla 9. Cada vez que se oprimen los botones, se recibe la información de los encoders, que multiplicado por la razón de transmisión, se puede visualizar la velocidad angular de las llantas.

El anexo S presenta el código completo de la Interfaz diseñada en Matlab®.

5.6 Costos del Proyecto

El proyecto se desarrolló utilizando algunos dispositivos prestados por el Laboratorio de Electrónica y por el grupo de investigación SIRP, y otros adquiridos personalmente. En el anexo T se encuentra la tabla de costos donde están registrados todas las partes físicas del proyecto, con su respectivo precio en pesos, en dólares y el sitio web de adquisición.

6. CONCLUSIONES

El uso de las comunicaciones inalámbricas en la robótica ha promovido temas de investigación en áreas de ciencia e ingeniería como es el caso del presente proyecto, que logró involucrar al usuario con un robot móvil de manera inalámbrica por medio de una interfaz amigable, cuya principal ventaja es que facilita su uso a cualquier tipo de personas, incluso sin tener conocimiento en robótica ya que cada elemento tiene un nombre claro que describe su función.

La interfaz desarrollada en este proyecto se realizó utilizando MatLab®, la cual es una herramienta muy versátil pues en ella se pueden realizar diferentes funciones y utilizarlas en diversas aplicaciones como controles de velocidad y de posición, control visual, procesamiento de imágenes, ejecución de simulaciones, adquisición, ordenamiento y visualización de datos, comunicaciones, gráficas, diseños de pruebas y medidas, entre otros. La interfaz diseñada en este proyecto es una plataforma que tiene el control de los movimientos del robot y visualiza los valores obtenidos por los sensores. La principal ventaja de MatLab®, es que permite hacer modificaciones en el código donde se pueden implementar leyes de control o realizar procesamiento de imágenes con ayuda de las herramientas que provee en base al robot construido y a las caracterizaciones realizadas.

Un sistema de comunicaciones inalámbrico hace posible la interacción del usuario con un dispositivo a distancia permitiendo el acceso desde cualquier terminal así como el control del mismo. El uso de este tipo de comunicaciones en el proyecto evidencia la integración entre la robótica y las comunicaciones inalámbricas así como la utilidad que generan, pues se pueden lograr envío y recepción de datos evitando el uso de largos cables y la dependencia del robot a un sitio fijo. Ésta combinación se logró gracias a las interfaces entre XBee y Arduino que se obtienen en el mercado, donde se promueve su unión generando facilidad tanto en la integración de hardware como de software.

Respecto a la plataforma robótica diseñada, es importante resaltar que la estructura es parte fundamental del robot, ya que su construcción influye tanto en la ubicación de los componentes, como en los movimientos que puede lograr. Es necesario desde un inicio tener una idea general de lo que se desea construir y conocer detalladamente el funcionamiento de cada una de sus partes, en especial si se quieren varios grados de libertad, para evitar de esta manera posibles interferencias mecánicas entre las piezas.

El kit de Vex Robotics es una excelente opción para el diseño y construcción de robots, pues además de contener una variedad de partes, están diseñadas para permitir fácilmente la integración entre ellas, como es el caso de motores y sensores con la estructura. Por otro lado, el fabricante presenta información completa sobre las especificaciones de cada parte y las posibles configuraciones que se pueden lograr, enseñando a sus usuarios desde los conceptos más básicos de la robótica hasta los más complejos.

El modelo dinámico del robot describe matemáticamente su comportamiento en el tiempo obteniendo una aproximación del movimiento en función de las velocidades de cada llanta. La importancia de la obtención del modelo radica en que es un requerimiento fundamental para diseñar leyes de control sobre él. De los resultados obtenidos en este proyecto se llega a la conclusión que el modelo del robot es semejante al modelo de un monociclo a excepción del valor de su velocidad angular, pues la configuración del sistema de movimiento impide que pueda girar sobre su propio eje y por ende obtener este valor en base a pruebas físicas. Este inconveniente se podría solucionar conectando los motores a las llantas traseras y dejando las llantas delanteras libres.

Se espera que este proyecto despierte el interés en el diseño e implementación de controladores para robots de forma inalámbrica y que la interfaz diseñada facilite la implementación de controles de velocidad, de posición y visual. Asimismo este proyecto puede lograr la interacción de diferentes campos de la electrónica como las comunicaciones y el control.

7. BIBLIOGRAFÍA

1. **Spong, Mark W., Hutchinson, Seth y Vidyasagar, M.** *Robot Modeling and Control.* s.l. : Editorial JOHN WILEY & SONS, INC.

2. **Garcia, G, Pomares, J y Torres, F.** *Control Visual de Robots Manipuladores. Herramienta para su diseño y aprendizaje.* Ciudad Real : Universidad de Alicante, 2004. Articulo.

3. **Moreno, Marco.** *Visión Artificial Estero con Aplicación al Control de un Brazo de Robot.* México D.F : Instituto Politécnico Internacional, 2003. Tesis de Doctorado.

4. **Lopez, Ana Milena.** *Diseño de leyes de control aplicando técnicas de Desigualdades Lineales Matriciales para un lazo de retroalimentación visual.* Bogotá : Pontificia Universidad Javeriana, 2010. Anteproyecto de Grado.

5. **Ollero Baturone, Anibal.** *Robótica: manipuladores y robots móviles.* Barcelona : Marcombo S.A, 2001. 84-267-1313-0.

6. Robótica, Ingeniería Informática de Sistemas. [En línea] [Citado el: 27 de Abril de 2011.] http://robotica.wordpress.com/about/.

7. **Guide, Inventors.** Vex Robotics. [En línea] 2004. [Citado el: 13 de 05 de 2011.] www.vexrobotics.com/27-2178.html.

8. **Bravo, Flor Angela.** *Robótica, una herramienta para la educación básica.* Bogotá : Pontificia Universidad Javeriana, 2009. Tesis de Grado.

9. **Ramón, Pallas y Fernando, Valdéz.** *Microcontroladores: Fundamentos y Aplicaciones con PIC.* España : Marcombo S.A, 2007.

10. **Orduña, Juan Manuel y Arnau, Vicente.** *Arquitectura y Programación de Microcontroladores.* Valencia : LLORENS, 1996. 84-370-2346-5.

11. **Fahimi, Farbod.** *Autonomous Robots: Modeling, Path Planning and Control.* Edmonton : Springer, 2009. 978-0-387-09537-0.

12. **Wayne, Taylor.** *Sistemas de Comunicaciones Electrónicas.* Mexico : Editorial Pearson Educación, 2003.

13. **Llorents, Vincent.** *Fundamentos Tecnológicos de Video y Televisión.* Barcelona : Editorial Paidós Iberica S.A, 2004. 84-493-0168.

14. **Haykin, Simon.** *Sistemas de Comunicación.* México D.F : Editorial Limusa, 2001.

15. **Esqueda, Jose y Palafox, Luis.** *Fundamentos de Procesamiento de Imagenes.* Mexico : Universitaria Baja California, 2005. 970-735-016-4.

16. **Herrera.** *Tecnologías y Redes de Transmisión de Datos.* Mexico D.F : Limusa. 968-18-6383-6.

17. **Caro, Dick.** *Wireless Network for Industrial Automation.* s.l. : Editorial ISA, 2008.

18. **Faludi, Robert.** *Wireless Sensor Networks.* United States of America : O'Reilly, 2011. 978-0-596-80773-3.

19. **Bernal, Isidorro.** *Operaciones Auxiliares de Montaje de Componentes Informáticos.* Madrid : Ediciones Paraninfo S.A, 2010. 978-84-9732-777-0.

20. Manual de capturadora Sensoray 2250s. [En línea] [Citado el: 5 de Mayo de 2011.] http://www.sensoray.com/downloads/Manual2250revA-10-10-06.pdf.

21. **Corporation, ATMel.** [En línea] [Citado el: 5 de Mayo de 2011.] <http://www.atmel.com/dyn/products/product_card.asp?part_id=4720>.

22. **MaxStream.** *XBee/XBee Pro OEM RF Modules.* Lindon : s.n., 2007.

23. **Works, MatLab.** MathWoks. [En línea] [Citado el: 15 de Mayo de 2011.] http://www.mathworks.com/products/matlab/description6.html.

24. [En línea] [Citado el: 15 de Mayo de 2011.] www.electron-bazar.com

25. **Arduino.** Arduino. [En línea] www.arduino.cc.

26. [En línea] [Citado el: 14 de Mayo de 2011.] http://es.made-in-china.com/co_jppbattery/product_9v-Battery-6f22-Carbon-Battery_hnryosorg.html.

27. Manual de programación de Arduino. [En línea] [Citado el: 14 de Mayo de 2011.] http://www.cep-cr.es/~fnavarro/robotica/CD%20I/documentos/Manual%20Programacion%20Arduino.pdf.

28.[En línea][Citado el: 13 de Mayo de 2011] (www.digi.com)

LISTA DE ANEXOS

ANEXO A- ESPECIFICACIONES DE LOS COMPONENTES

El robot diseñado en este proyecto está compuesto por los siguientes componentes del kit de Vex, cuyas especificaciones se resumen a continuación.

SERVO MOTOR	
Características	Valor
Rotación	100°
Torque	6.5 pulgadas/libra
Voltaje Máximo y Mínimo	4.4-9.1 V
Corriente	20 mA – 1.5 A
Entrada PWM	1 ms a 2ms donde 1.5 ms es neutro
Conexiones	Negro- Tierra; Naranja- Voltaje de Alimentación; Blanco- Señal PWM
Peso	0,096 Libras

Tabla A 1 Especificaciones del Servomotor

[http://www.vexrobotics.com/products/accessories/motion/276-2162.html]

MOTOR DC	
Características	Valor
Velocidad	100 rpm a 7.5 Voltios
Torque	6.5 pulgadas/libra
Voltaje Máximo y Mínimo	4.4 – 15 Voltios
Corriente	5 mA a 1 A
Entrada PWM	1 ms a 2ms donde 1.5 ms es neutro
Conexiones	Negro- Tierra; Naranja- Voltaje de Alimentación; Blanco- Señal PWM
Peso	0.21 Libras

Tabla A 2 Especificaciones del Motor DC

[http://www.vexrobotics.com/products/accessories/motion/276-2181.html]

LLANTAS	
Diámetro	2,75 pulgadas
Peso	0,12 Libras

Tabla A 3 Especificaciones de las llantas

[http://www.vexrobotics.com/products/accessories/motion/276-1496.html]

ENCODERS	
Tipo	Rotación Mecánica
Sensor que usa	Sensor de Luz Infrarrojo
Resolución	90 pulsos por revolución
Peso	0,078 Libras
Conexiones	Negro- Tierra; Rojo- Voltaje de Alimentación; Blanco- Señal de Salida

Tabla A 4 Especificaciones del encoder

[http://www.vexrobotics.com/products/accessories/sensors/276-2156.html]

SENSOR DE LIMITE	
Tipo	Interruptor de un polo
Conexión Switch	Normalmente en fuente
Peso	0,03 Libras
Conexiones	Negro- Tierra; Rojo- Voltaje de Alimentación; Blanco- Señal de Salida

Tabla A 5 Especificaciones del sensor de límite

[http://www.vexrobotics.com/products/accessories/sensors/276-2174.html]

POTENCIÓMETRO	
Tipo	Rotación Mecánica
Medida	250º
Peso	0,04 Libras
Conexiones	Negro- Tierra; Rojo- Voltaje de Alimentación; Blanco- Señal de Salida

Tabla A 6 Especificaciones del Potenciómetro

[http://www.vexrobotics.com/products/accessories/sensors/276-2216.html]

ANEXO B- FOTOS DE LA ESTRUCTURA

Para diseñar el soporte del robot, se utilizó la estructura del kit de Vex Robotics, utilizando placas, tornillos y varas cuadradas de metal, de manera que se pueden integrar todos los componentes del robot con el chasis. A continuación se presentan las imágenes de la estructura.

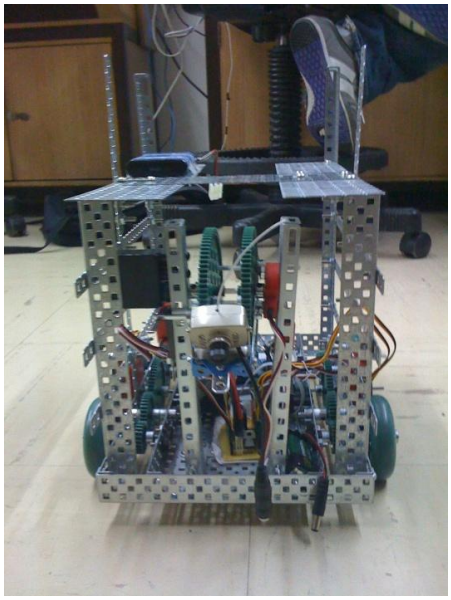


Figura B 1 Vista Frontal

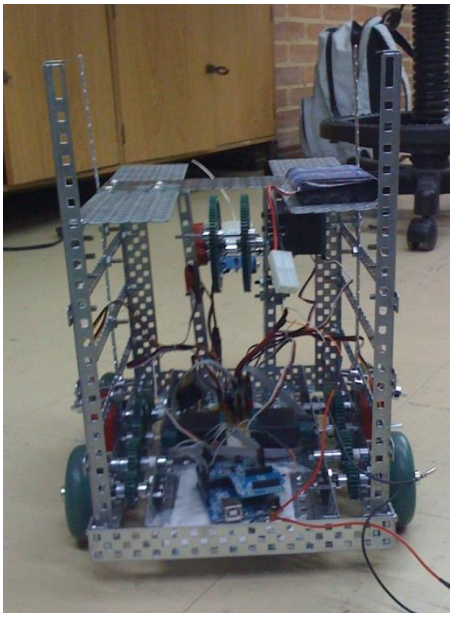


Figura B 2 Vista Trasera

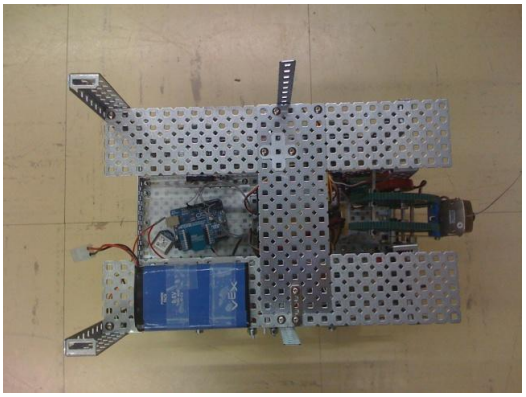


Figura B 3 Vista Superior

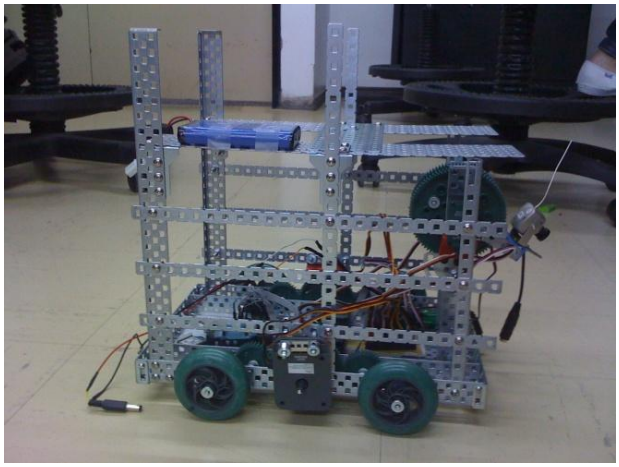


Figura B 4 Vista Costado Derecho

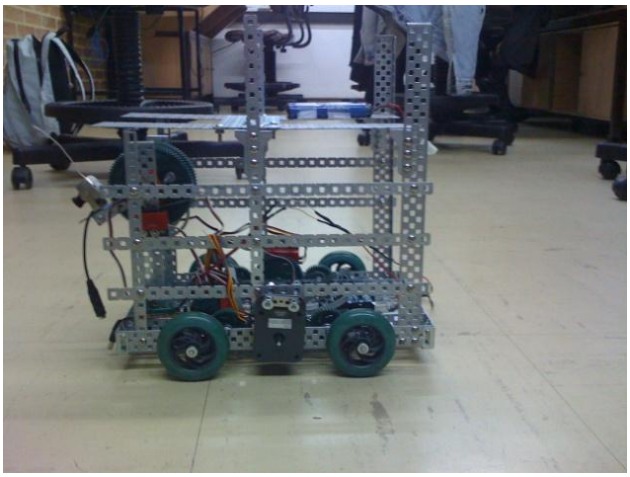


Figura B 5 Vista Costado Izquierdo

ANEXO C - CARACTERÍSTICAS DEL MICROCONTROLADOR ATMEGA 328

Parámetros claves	valor
Kbytes	32 Kbytes
Frecuencia Máxima de operación	20 Hz
Máximo de pines de entrada/salida	23
Sensor de temperatura	si
EEPROM	1024 Bytes
Rangos de temperatura	-40 to 85 °C
Timers	3
Canales PWM	6

Tabla C 16 Características ATMega 328

[http://www.atmel.com/dyn/products/product_parameters.asp?category_id=163&family_id=607&subfamily_id=760&part_id=4720&ListAllAttributes=1.]

ANEXO D-ESPECIFICACIONES DE LA TARJETA ARDUINO UNO

Se presentan las especificaciones de la tarjeta Arduino utilizada, así como el esquemático del mismo.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Tabla D 1 Especificaciones de la tarjeta Arduino

[www.arduino.cc]

Esquemático Arduino

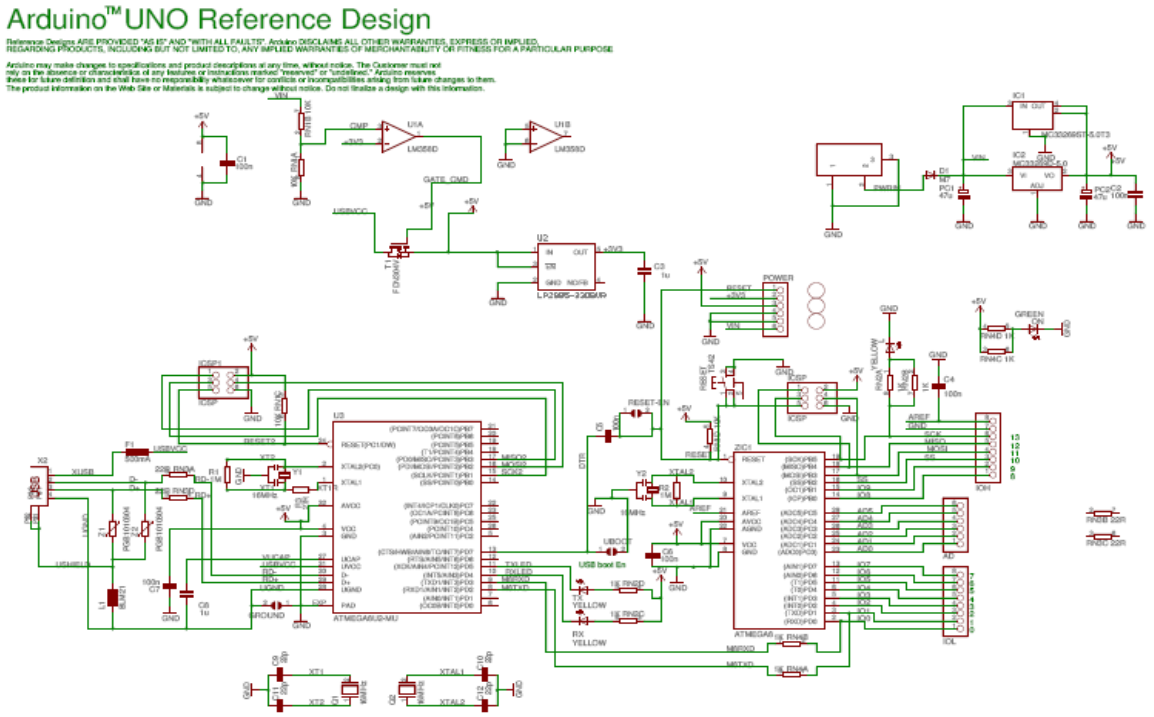


Figura D- 1 Esquemático de Arduino UNO

[www.arduino.cc]

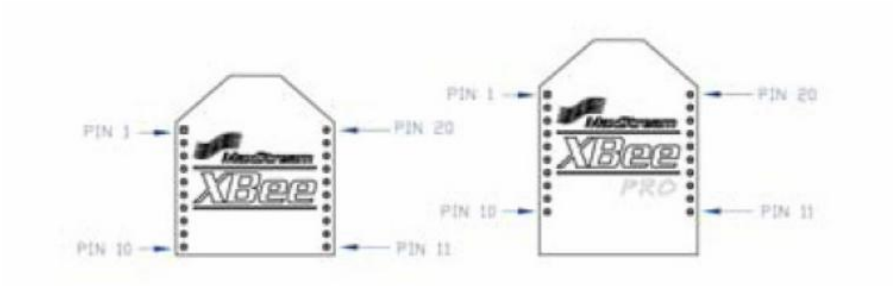
ANEXO E- ESPECIFICACIONES DE LA XBEE

Platform	XBee® ZB	XBee-PRO® ZB
Performance		
RF Data Rate	250 Kbps	
Indoor/Urban Range	133 ft (40 m)	300 ft. (90 m) / Int'l 200 ft (60 m)
Outdoor/RF Line-of-Sight Range	400 ft (120 m)	2 miles (3200 m)/ Int'l 5000 ft (1500 m)
Transmit Power	1.25 mW (+1 dBm) / 2 mW (+3 dBm) boost mode	63 mW (+18 dBm) / Int'l 10 mW (+10 dBm)
Receiver Sensitivity (1% PER)	-96 dBm in boost mode	-102 dBm
Features		
Adjustable Power	Yes	
Serial Data Interface	3.3V CMOS UART	
Configuration Method	API or AT commands, local or over-the-air	
Frequency Band	2.4 GHz	
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)	
Serial Data Rate	1200 bps - 1 Mbps	
ADC Inputs	(4) 10-bit ADC inputs	
Digital I/O	10	
Antenna Options	Chip, Wire Whip, U.FL RPSMA	PCB Embedded Antenna, Wire Whip, U.FL, RPSMA
Programmability	No	Yes
Networking & Security		
Encryption	128-bit AES	
Reliable Packet Delivery	Retries/Acknowledgments	
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 channels	PAN ID, 64-bit IEEE MAC, 15 channels
Power Requirements		
Supply Voltage	2.1 - 3.6VDC	2.7 - 3.6VDC
Transmit Current	35 mA / 45 mA boost mode @ 3.3VDC	205 mA, up to 220 mA with programmable variant
Receive Current	38 mA / 40 mA boost mode @ 3.3VDC	47 mA, up to 62 mA with programmable variant
Power-Down Current	<1 uA @ 25° C	3.5 uA, 4 uA @ 25° C with programmable variant
Regulatory Approvals		
FCC (USA)	Yes	
IC (Canada)	Yes	
ETSI (Europe)	Yes (10 mW or less only)	
C-TICK (Australia)	Yes	
TELEC (Japan)	Yes	Yes (int'l unit only)

Tabla E 1 Especificaciones de la XBee

[XBEE Manual- http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Manual.pdf]

La asignación de pines de la XBee se resume en la siguiente tabla, obtenida del manual realizado por sus fabricantes.



Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DIO12	Either	Digital I/O 12
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI / DIO10	Either	PWM Output 0 / RX Signal Strength Indicator / Digital IO
7	PWM / DIO11	Either	Digital I/O 11
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ/ DIO8	Either	Pin Sleep Control Line or Digital IO 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP** / **DIO9**	Output	Module Status Indicator or Digital I/O 9
14	[reserved]	-	Do not connect
15	Associate / DIO5	Either	Associated Indicator, Digital I/O 5
16	**RTS** / DIO6	Either	Request-to-Send Flow Control, Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0 / Commissioning Button	Either	Analog Input 0, Digital IO 0, or Commissioning Button

Tabla E 2 Asignación de pines la XBee

[XBEE Manual- http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Manual.pdf]

Esquemático Tarjeta XBEE Pro.

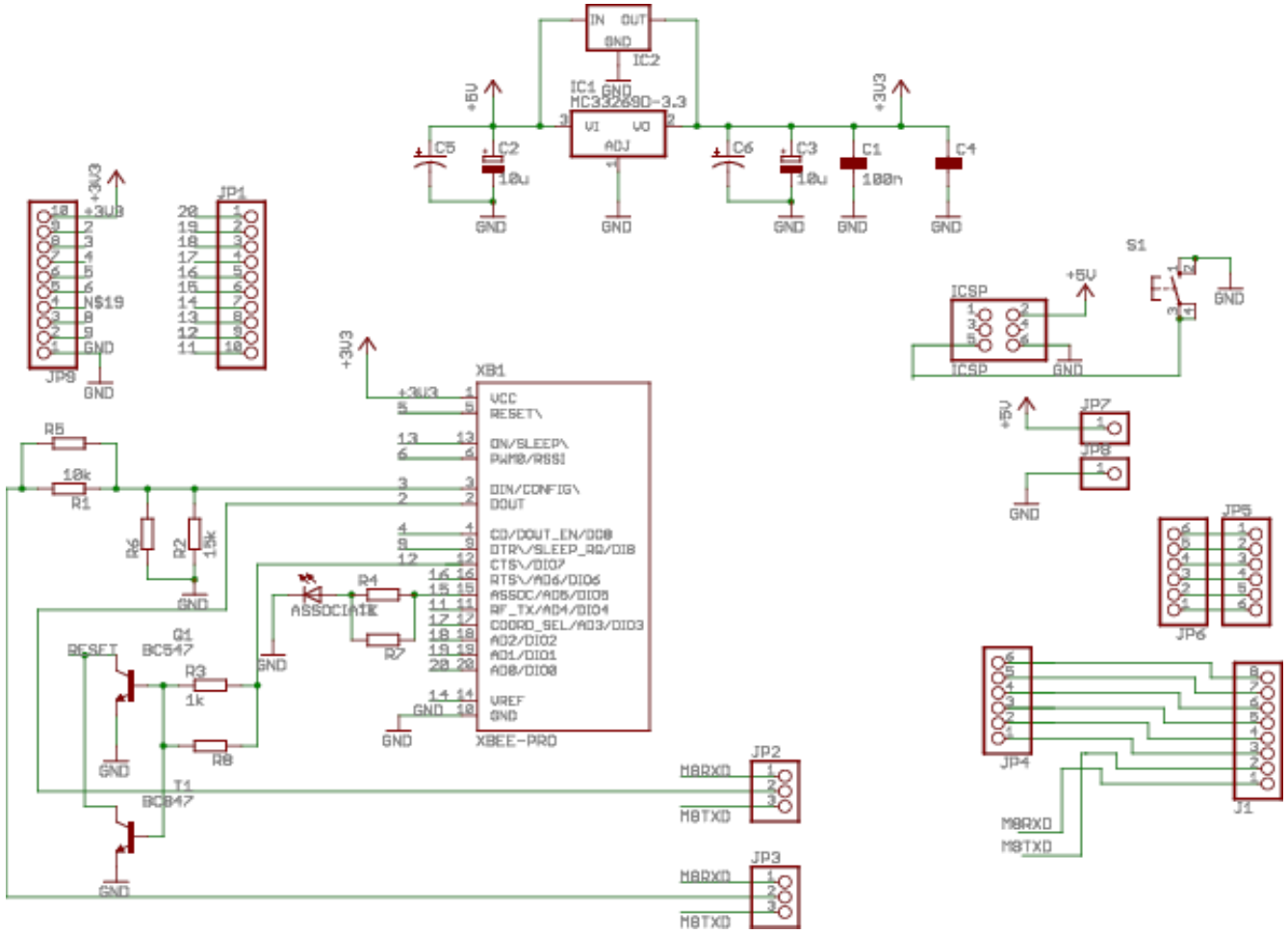


Figura E- 1 Esquemático Tarjeta Xbee

[XBEE Manual- http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Manual.pdf]

ANEXO F- CÓDIGO DEL SENSOR DE VELOCIDAD (ENCODER) EN ARDUINO UNO

```
long previousMillis = 0;    // Se inicializa en 0
long interval = 1000;   // Se define el intervalo de tiempo de 1000 milisegundos (1 segundo)
long previousMillis1 = 0;    // Se inicializa en 0
long interval1 = 1000;  // Se define el intervalo de tiempo de 1000 milisegundos (1 segundo)

#include <Servo.h>
Servo myservo;   //Se crea el motor derecho
int pos = 90;      // // Variable para la velocidad del motor derecho
int valor;    // Variable que guardará el número de pulsos del encoder derecho
Servo myservo_2;    // Se crea el motor izquierdo
int pos_2 = 90;     // Variable para la velocidad del motor izquierdo
int valor_2;      // Variable que guardará el número de pulsos del encoder izquierdo
int encoder0PinA = 4;   // El encoder del motor derecho se conecta al pin digital 4
int encoder0Pos = 0;   // Se inicializa el valor de los pulsos que se van a contar
int encoder0PinALast = LOW;  // Comienza en bajo la señal
int n = LOW;
float velocidad=0;   // Se inicializa el valor de la velocidad en 0
int encoder1PinB = 2;    // El encoder del motor izquierdo se conecta al pin digital 2
int encoder1Pos1 = 0;   // Se inicializa el valor de los pulsos que se van a contar
int encoder1PinBLast = LOW;    // Comienza en bajo la señal
int m = LOW;
float velocidad_2=0;   // Se inicializa el valor de la velocidad en 0

void setup() {
  myservo.attach(6);                // Motor derecho
  myservo_2.attach(11);             // Motor izquierdo

  pinMode(encoder0PinA, INPUT);   // Se inicializa como entrada al pin del encoder derecho
  digitalWrite(encoder0PinA, HIGH);     // Se toma el valor cuando la señal este en alto
  pinMode(encoder1PinB, INPUT);    // Se inicializa como entrada al pin del encoder izquierdo
  digitalWrite(encoder1PinB, HIGH);   // Se toma el valor cuando la señal este en alto

  attachInterrupt(0, doEncoder, CHANGE);   // Función que se realiza al haber un cambio de pulso
  attachInterrupt(1, doEncoder_2, CHANGE); // Función que se realiza al haber un cambio
  Serial.begin (9600);   // Se especifica la velocidad de la transmisión de datos en serie.
  Serial.println("start");   // Se visualiza en el serial monitor ´start´ al empezar el programa
}

void loop() {

   myservo.write(pos);   // El motor derecho varia su velocidad con respecto al valor ´pos´
   myservo_2.write(pos_2);   // El motor izquierdo varia su velocidad con respecto al valor ´pos_2´
}

void doEncoder() {
  unsigned long currentMillis = millis();   // Comienza a contar el tiempo
    n = digitalRead(encoder0PinA);   // Se guarda en la variable ´n´ la lectura del encoder derecho
      if ((encoder0PinALast == LOW) && (n == HIGH)) {  // cuando hay pulso, la variable cuenta
```

```
            encoder0Pos++;  // Conteo de la variable

             if(currentMillis - previousMillis > interval) {   // Después del segundo transcurrido,
               int valor = encoder0Pos;  // ´valor´ guarda la cantidad de pulsos contados
               encoder0Pos=0;   // Se reinicia para volver a contar

               velocidad = (valor*2*3.14)/90;    // Se hace la fórmula para determinar la velocidad
               previousMillis = currentMillis;    // Se reinicia el tiempo
              Serial.print ("Velocidad: ");
             Serial.println (velocidad);   // Se imprime el valor de la velocidad
             }
  }
  encoder0PinALast = n;   // Se reinicia la variable ´n´
}
void doEncoder_2() {
  unsigned long currentMillis1 = millis();   // Comienza a contar el tiempo
    m = digitalRead(encoder1PinB);  // Se guarda en ´m´ la lectura del encoder izquierdo
      if ((encoder1PinBLast == LOW) && (m == HIGH)) {  // cuando hay pulso, la variable cuenta
            encoder1Pos1++;   // Conteo de la variable

             if(currentMillis1 - previousMillis1 > interval1) {   // Después del segundo transcurrido,
               int valor_2 = encoder1Pos1;  // ´valor_2´ guarda la cantidad de pulsos contados
               encoder1Pos1=0;    // Se reinicia para volver a contar

               velocidad_2 = (valor_2*2*3.14)/90;  // Se hace la fórmula para determinar la velocidad
               previousMillis1 = currentMillis1;  // Se reinicia el tiempo
              Serial.print ("Velocidad_2: ");
             Serial.println (velocidad_2);  // Se imprime el valor de la velocidad
             }
  }
  encoder1PinBLast = m;   // Se reinicia la variable ´m´

}
```

ANEXO G- CONEXIONES DE LOS COMPONENTES CON LA TARJETA ARDUINO UNO

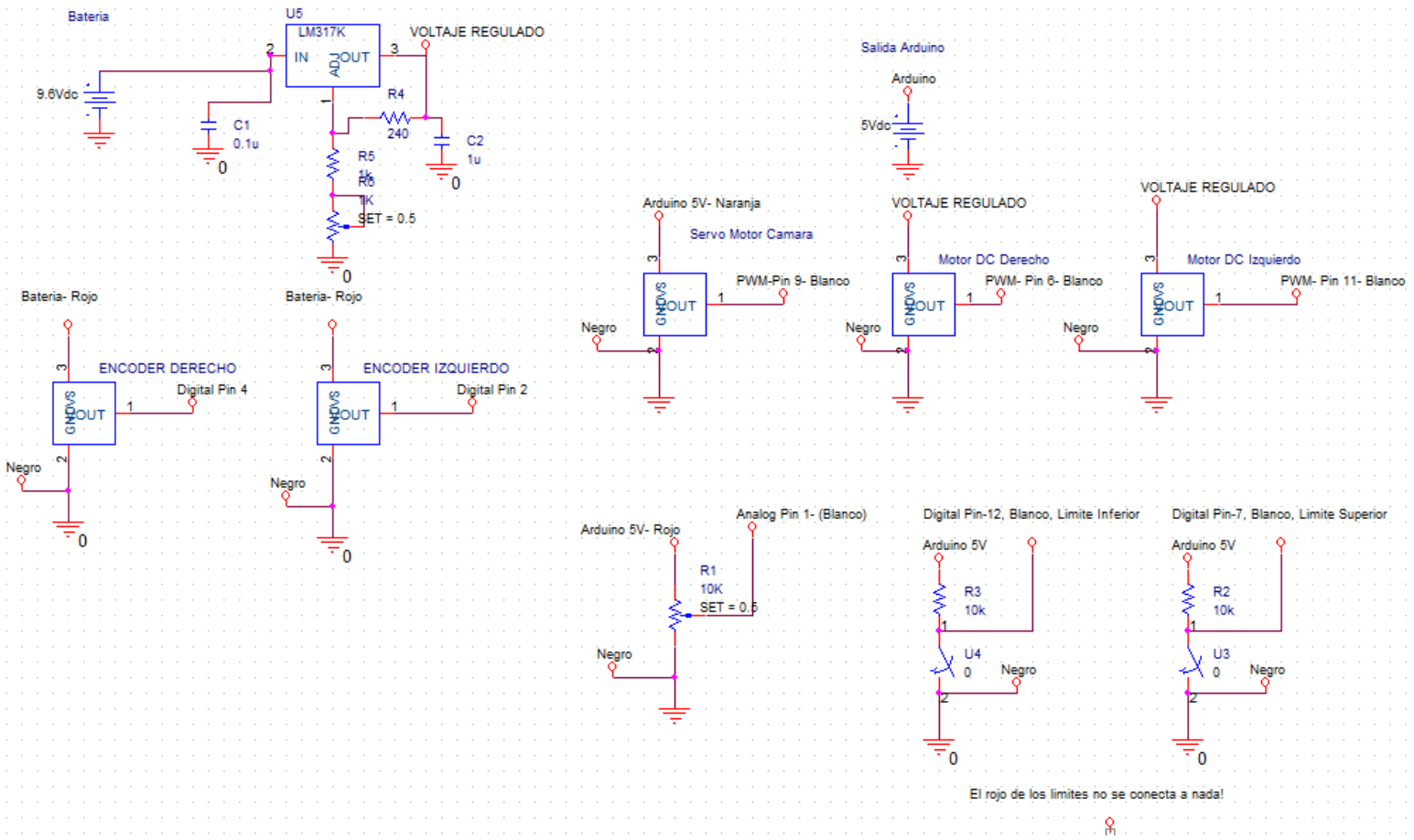


Figura G 1 Esquemático de los componentes

El circuito impreso del esquemático anterior se observa en la figura G 2, seguido de las respectivas conexiones resumidas en la Tabla G 1.

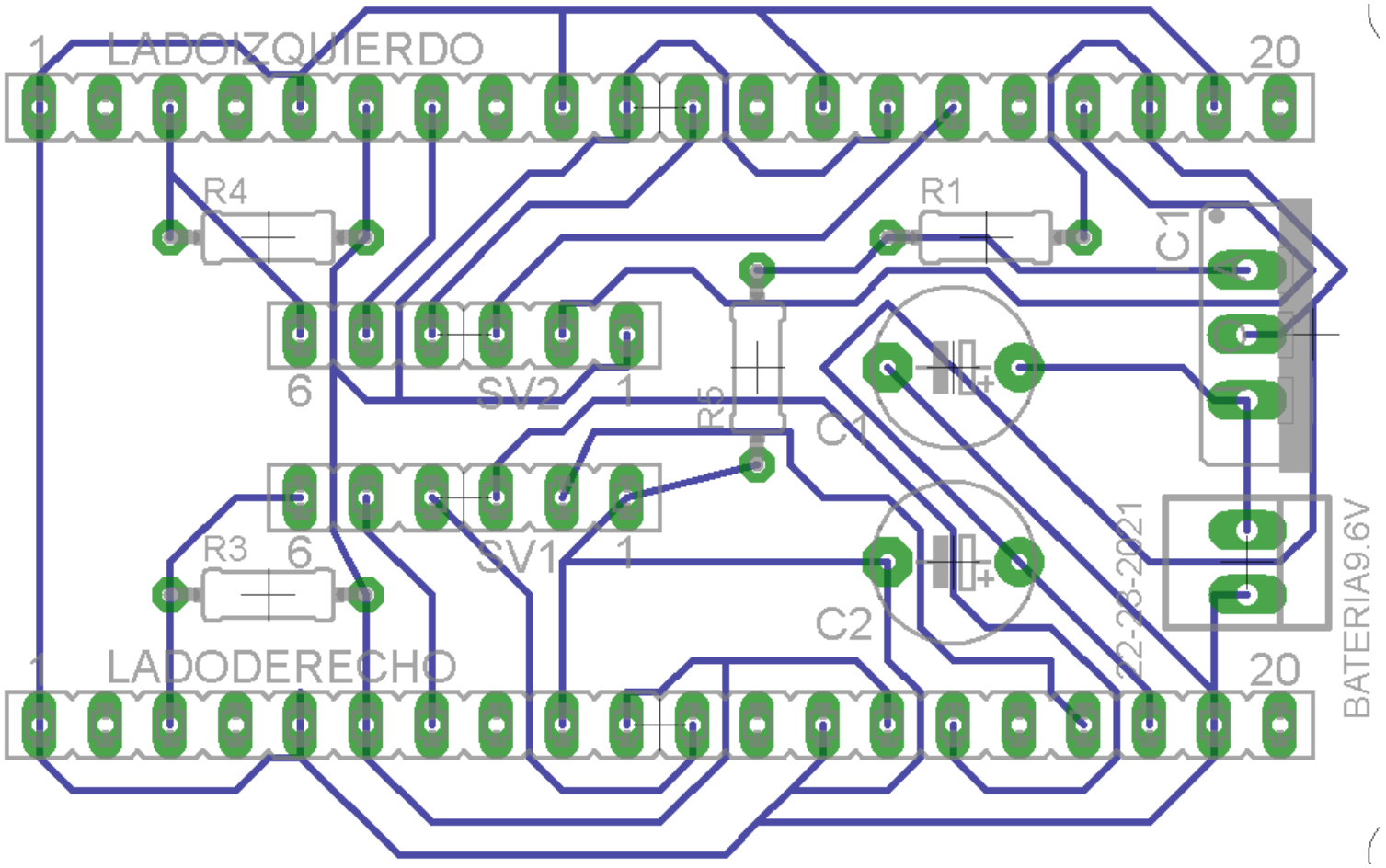


Figura G 2 Circuito Impreso

LADO IZQUIERDO				LADO DERECHO			
COMPONENTE	**CABLE**	**POSICIÓN EN LA REGLETA EXTERIOR**	**POSICIÓN EN LA REGLETA INTERIOR**	**COMPONENTE**	**CABLE**	**POSICIÓN EN LA REGLETA EXTERIOR**	**POSICIÓN EN LA REGLETA INTERIOR**
Limite Superior	Negro	1	6	**Límite Inferior**	Negro	1	6
	Rojo	2			Rojo	2	
	Blanco	3			Blanco	3	
Potenciómetro	Negro	5	5	**Servomotor**	Negro	5	5
	Rojo	6			Rojo	6	
	Blanco	7			Blanco	7	
Encoder Izquierdo 1	Negro	9	4	**Encoder Derecho 1**	Negro	9	4
	Rojo	10			Rojo	10	
	Blanco	11			Blanco	11	
Encoder Izquierdo 2	Negro	13	3	**Encoder Derecho 2**	Negro	13	3
	Rojo	14			Rojo	14	
	Blanco	15			Blanco	15	
Motor DC Izquierdo	Blanco	17	2	**Motor DC Derecho**	Blanco	17	2
	Naranja	18			Naranja	18	
	Negro	19			Negro	19	

Tabla G 1 Conexiones entre los componentes y el circuito impreso

La posición 1 de las regletas interiores del lado derecho y del lado izquierdo corresponde a la tierra y al voltaje regulado, respectivamente.

ANEXO H- CÓDIGO DE RECEPCIÓN DE DATOS DESDE ARDUINO UNO

```
#include <Servo.h>
Servo myservo;    //Se crea elservomotor
Servo myservo_2;  //Se crea el motor derecho
Servo myservo_3;  //Se crea el motor izquierdo
int pos;   // Control cámara
int pos_2;   // Control motor izquierdo
int pos_3;    // Control motor derecho

void setup()
{
myservo.attach(9);       // Asignación del Pin de Control de la Cámara
myservo_2.attach(11);    // Asignación del Pin de Control del motor izquierdo
myservo_3.attach(6);     //Asignación del Pin de Control del motor derecho
Serial.begin(9600);    // Abre el puerto serial a una tasa de transmisión de 9600
}

void loop()
{
if (Serial.available() > 0) {
pos = Serial.read();     // Recepción Control Cámara
pos_2 = Serial.read();    // Recepción Control Motor Izquierdo
pos_3 = Serial.read();    // Recepción Control Motor Derecho
myservo.write(pos);    // El servomotor varia su posición con respecto al valor ´pos´
myservo_2.write(pos_2);    // El motor derecho varia su velocidad con respecto al valor ´pos_2´
myservo_3.write(pos_3);     // El motor izquierdo varia su velocidad con respecto al valor ´pos_3´
Serial.println(pos);    // Muestra pos en Serial Monitor
Serial.println(pos_2);    // Muestra pos_2 en Serial Monitor
Serial.println(pos_3);     // Muestra pos_3 en Serial Monitor
}
}
```

ANEXO I- CÓDIGO ENVÍO DE DATOS DESDE ARDUINO UNO

```
#include <Servo.h>
Servo myservo;  // Se crea el servomotor
int pos = 50;    //Se define un valor manual para la posición de la cámara.
int analogPin=1;    // Definición del Pin del Potenciómetro
int digitalPin=12;   // Definición del Pin del Límite Inferior
int digitalPin2=7;   // Definición del Pin del Limite Superior
int val=0;   //Valores Iniciales
int val2=0;    //Valores Iniciales
int val3=0;     //Valores Iniciales

void setup() {
  Serial.begin(9600);     // Abre el puerto serial a una tasa de transmisión de 9600
  myservo.attach(9);     // Asignación del Pin de Control de la Cámara
}


void loop() {
          if (Serial.available() > 0) {     // Comprueba la disponibilidad del puerto
    val=digitalRead (digitalPin);     //Se lee límite Inferior
    Serial.println (val,2);     //Envía el valor de Val (Limite Inferior)
    val2=digitalRead (digitalPin2);     // Se lee límite superior.
    Serial.println (val2,2);     // Envía el valor de Val2 (Limite Superior)
    val3=analogRead (analogPin);     // Se lee valor del Potenciómetro
    Serial.println (val3);     //Envía el valor de Val3 (Potenciómetro)
          }
    }
```

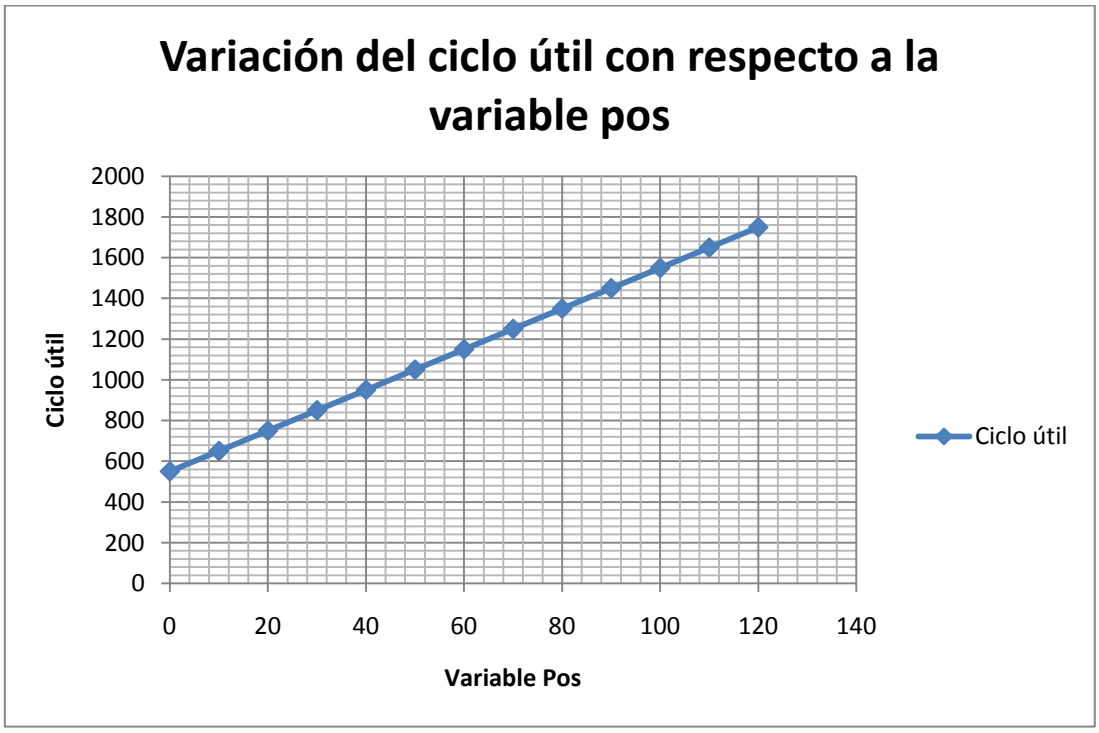
ANEXO J- CÓDIGO EN MATLAB® PARA ENVÍO Y RECEPCIÓN DE DATOS POR PUERTO SERIAL

```
delete(instrfindall);% Cierra todos los puertos que encuentre abiertos
PS=serial('COM1'); %Se define PS como el nombre del puerto COM1
set(PS,'Baudrate',9600); % Se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % Se configura bit de parada a uno
set(PS,'DataBits',8); % Se configura que el dato es de 8 bits
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS); % Se abre el puerto
control=50; % Valor de control

fwrite(PS,'A','uchar');% Envia un valor que indica que iniciará la
comunicación
fwrite(PS,'B','uchar'); % Envia una bandera que indica que valor enviara;
                        % B significa Dato para el servomotor
                        % C significa dato para motor DC derecho
                        % D significa dato para motor DC izquierdo
fwrite(PS,control,'uchar'); %Envio control que corresponda, en este caso, al
servo

variable1= fscanf(PS,'%c');% Almcena información sobre limite superior
variable2= fscanf(PS,'%c');%Almacena información sobre limite inferior
variable3= fscanf(PS,'%c');%Almacena información sobre potenciometro
variable4= fscanf(PS,'%c');%Almacena información sobre encoder derecho
variable5= fscanf(PS,'%c');% Almacena información sobre encoder izquierdo



fclose(PS); %Cierra puerto
delete(PS);%Elimina puerto
clear PS;%Deja la variable PS sin  información
```

ANEXO K- VARIACIÓN DEL CICLO ÚTIL SEGÚN LA VARIABLE POS

Variable Pos	Ciclo Útil (µs)
0	550
10	650
20	750
30	850
40	950
50	1050
60	1150
70	1250
80	1350
90	1450
100	1550
110	1650
120	1750

Tablas K 1 Variación del ciclo útil vs. Variable pos

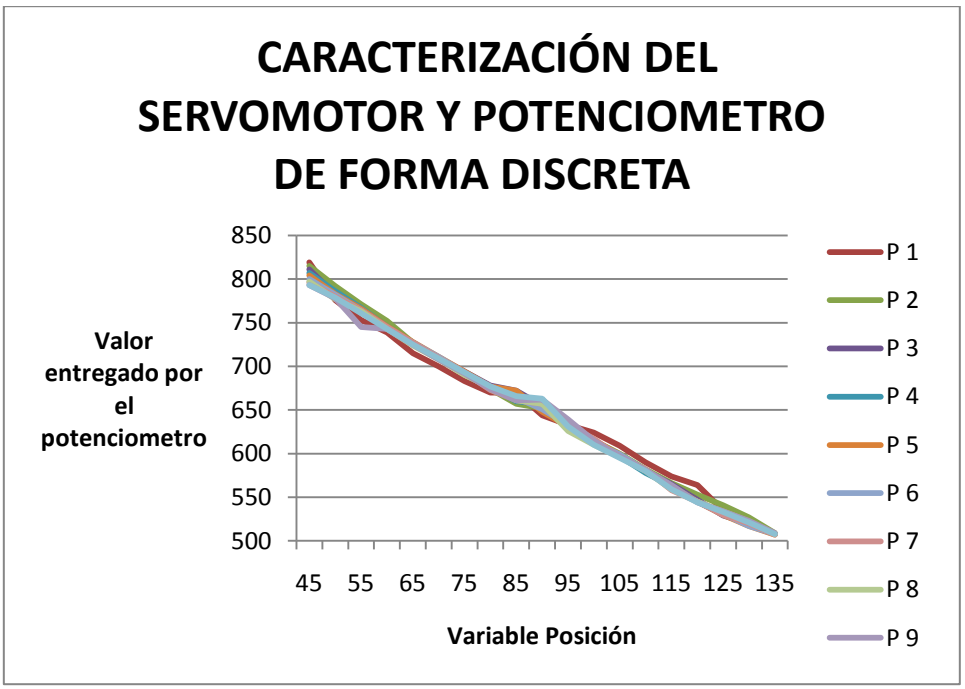


Grafica K 2 Variación del ciclo útil vs. Variable pos

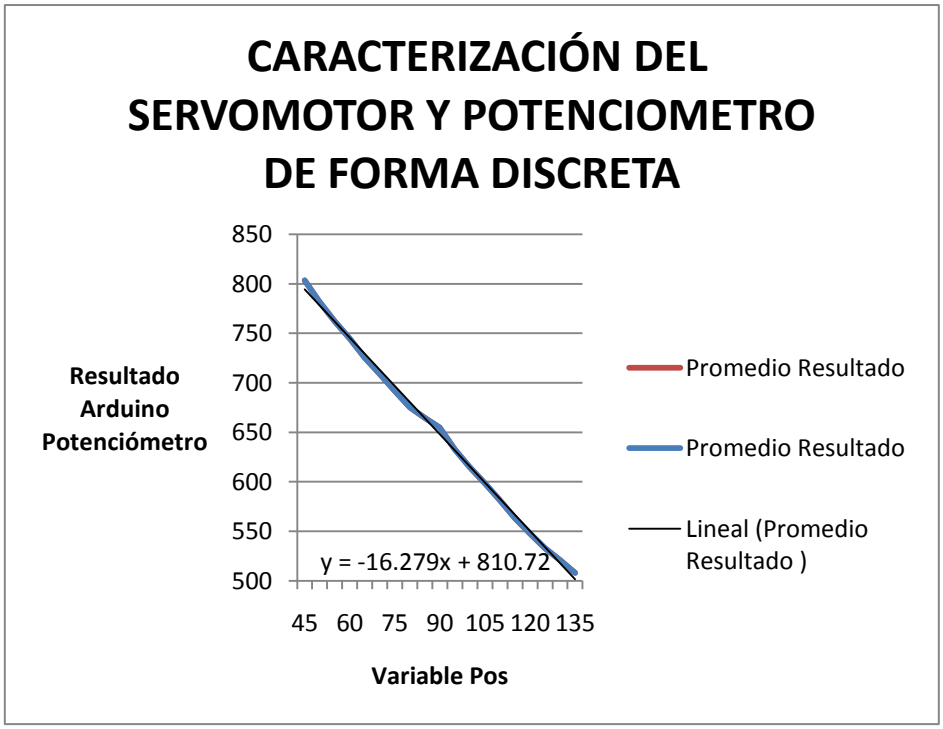
ANEXO L - CARACTERIZACIÓN DEL SERVOMOTOR Y POTENCIÓMETRO DE FORMA DISCRETA

CÁMARA	Resultados del Potenciómetro en Arduino										
Variable Pos	**P 1**	**P 2**	**P 3**	**P 4**	**P 5**	**P 6**	**P 7**	**P 8**	**P 9**	**P 10**	**Promedio**
45	819	815	811	807	804	800	796	797	793	793	803,5
50	776	792	782	786	783	782	780	777	778	779	781,5
55	753	771	766	767	766	764	764	763	745	761	762
60	739	752	744	746	746	744	745	743	743	743	744,5
65	715	727	726	727	728	727	727	724	725	725	725,1
70	700	709	710	710	711	711	710	708	708	709	708,6
75	683	692	694	691	694	693	693	690	692	693	691,5
80	670	674	678	675	677	677	673	676	674	677	675,1
85	670	657	672	660	672	664	661	661	661	666	664,4
90	644	652	655	657	648	651	657	657	662	663	654,6
95	633	637	637	629	631	633	633	626	639	631	632,9
100	624	615	613	613	615	613	617	610	615	610	614,5
105	609	600	597	597	599	599	596	596	597	595	598,5
110	590	582	580	578	580	581	582	580	581	580	581,4
115	574	566	565	563	563	560	558	559	564	559	563,1
120	564	553	547	545	545	545	545	545	544	544	547,7
125	535	541	529	531	530	531	530	534	534	533	532,8
130	525	527	520	518	517	517	520	522	523	521	521
135		509	508	508	507	508	507	508	509	508	508

Tabla L 1 Caracterización del servomotor y potenciómetro de forma discreta



Grafica L 1 Caracterización del servomotor y potenciómetro de forma discreta

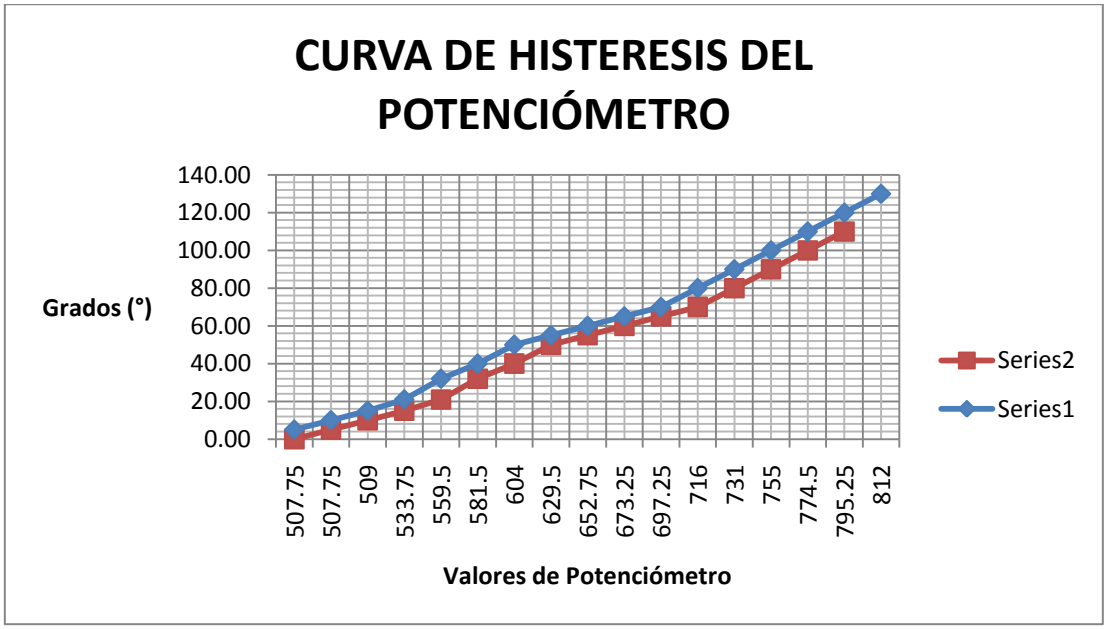


Grafica L 2 Caracterización del servomotor y potenciómetro de forma discreta

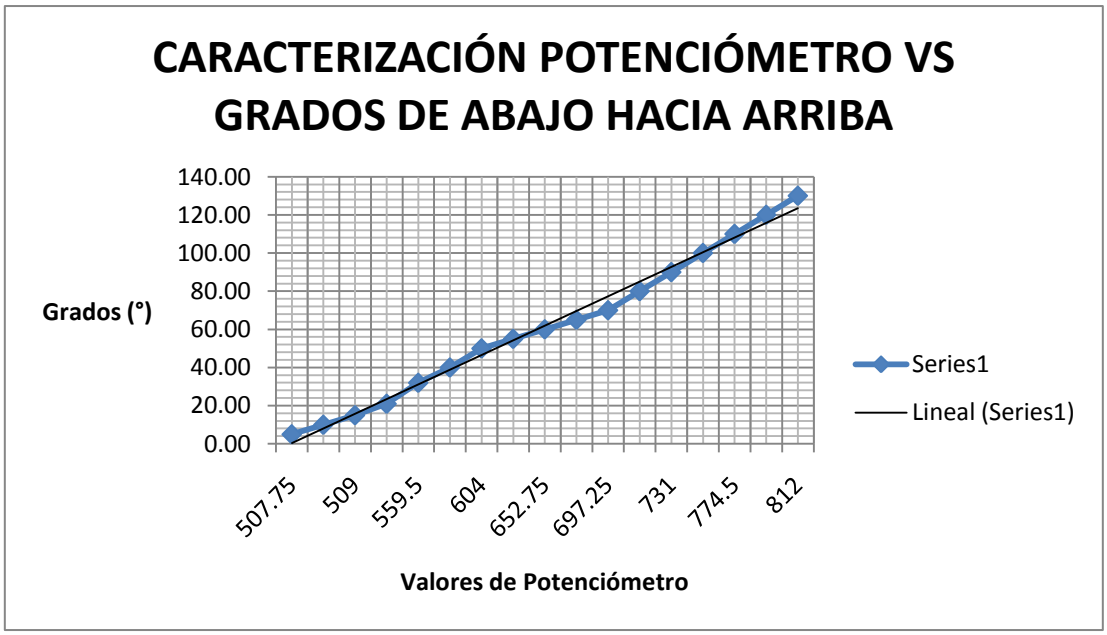
ANEXO M - CARACTERIZACIÓN DEL SERVOMOTOR Y POTENCIÓMETRO DE FORMA GRADUAL

Grados	Pos	MEDIDA DE ARRIBA HACIA ABAJO					PROMEDIO	Grados	Pos	MEDIDA DE ABAJO HACIA ARRIBA				PROMEDIO	Grados
130	40	x	x	x	x	x	x	130	40	800	816	816	816	812	130
120	45	X	x	x	x	x	x	120	45	795	795	795	796	795,25	120
110	50	836	834	834	833	834	834,2	110	50	774	775	774	775	774,5	110
100	55	827	827	827	828	827	827,2	100	55	755	755	755	755	755	100
90	60	812	815	817	814	815	814,6	90	60	723	734	733	734	731	90
80	65	781	778	778	781	779	779,4	80	65	716	716	716	716	716	80
70	70	756	757	761	761	759	758,8	70	70	697	697	697	698	697,25	70
65	75	733	733	729	731	729	731	65	75	675	675	675	668	673,25	65
60	80	705	706	706	702	706	705	60	80	652	653	653	653	652,75	60
55	85	683	682	684	682	683	682,8	55	85	629	630	630	629	629,5	55
50	90	659	658	659	656	658	658	50	90	603	604	604	605	604	50
40	95	635	632	635	635	633	634	40	95	581	582	581	582	581,5	40
32	100	609	610	609	609	610	609,4	32	100	558	561	560	559	559,5	32
21	105	587	587	585	586	587	586,4	21	105	533	534	534	534	533,75	21
15	110	561	564	564	563	563	563	15	110		510	509	508	509	15
10	115	545	545	544	544	543	544,2	10	115	508	509	507	507	507,75	10
5	120	529	531	532	530	530	530,4	5	120	508	509	507	507	507,75	5
0	125	519	520	521	519	520	519,8	0	125	x	x	x	x	x	0

Tabla M 1 Caracterización del servomotor y potenciómetro de forma gradual

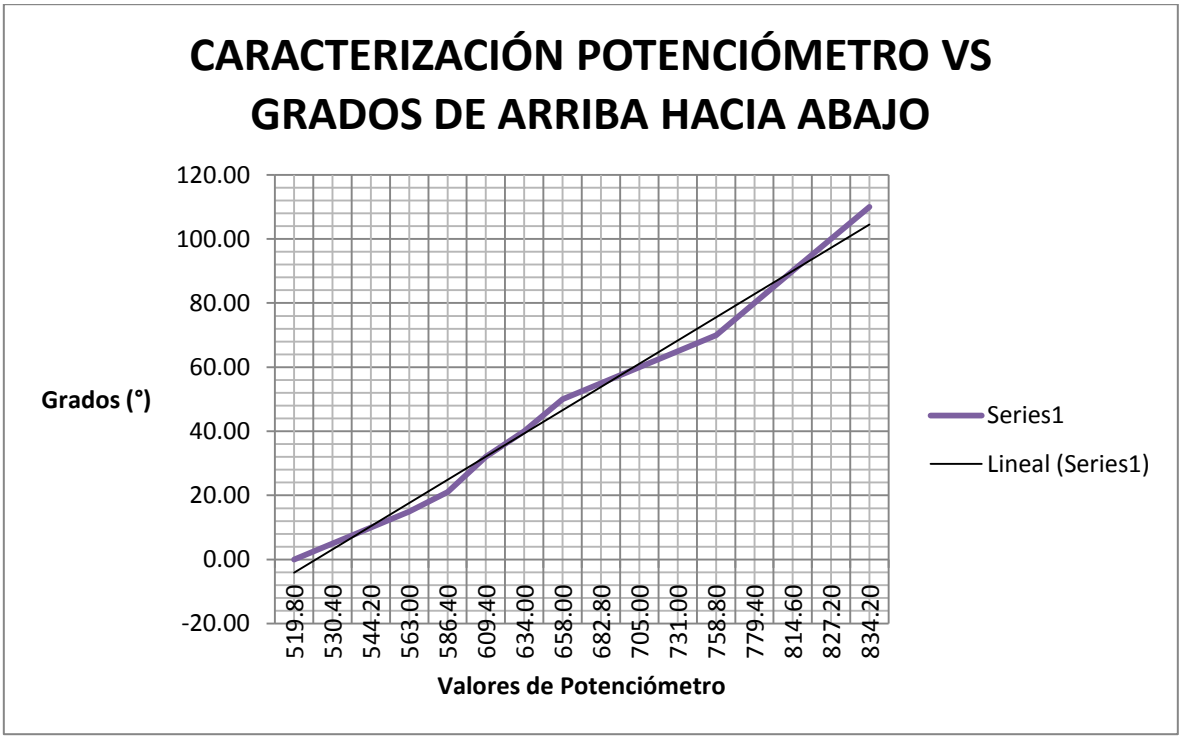


Grafica M 1 Curva de histéresis del potenciómetro



Ecuación: $y = 0.345 * x - 160.5$

Grafica M 2 Caracterización creciente



Ecuación: $y = 0.282 * x - 139.7$

Grafica M 3 Caracterización decreciente

ANEXO N – CARACTERIZACIÓN DE MOTORES DC Y SENSOR DE VELOCIDAD (ENCODER)

Motor Derecho (6) Encoder (4)			Motor Izquierdo (11) Encoder (2)		
POS -PWM	W Encoder		POS -PWM	W Encoder	
	Valor	W= Valor*2*pi/90		Valor_2	W= Valor*2*pi/90
45	121	8,44	45	159	11,09
	116	8,09		161	11,23
	124	8,65		161	11,23
	110	7,68		149	10,4
	123	8,58		160	11,16
PROMEDIO		8,288	PROMEDIO		11,022
48	128	8,93	48	155	10,82
	116	8,09		156	10,89
	123	8,58		155	10,82
	110	7,68		154	10,75
	113	7,88		154	10,75
PROMEDIO		8,232	PROMEDIO		10,806
51	106	7,4	51	151	10,54
	116	8,09		152	10,61
	119	8,3		152	10,61
	117	8,16		152	10,61
	107	7,47		151	10,54
PROMEDIO		7,884	PROMEDIO		10,582
54	109	7,61	54	149	10,4
	117	8,16		149	10,4
	118	8,23		150	10,47
	116	8,09		148	10,33
	118	8,23		149	10,4
PROMEDIO		8,064	PROMEDIO		10,4
57	115	8,02	57	145	10,12
	113	7,88		145	10,12
	126	8,79		146	10,19
	112	7,82		144	10,05
	118	8,23		140	9,77
PROMEDIO		8,148	PROMEDIO		10,05
60	100	6,98	60	139	9,7
	110	7,68		138	9,63
	111	7,75		139	9,7
	91	6,35		139	9,7
	94	6,56		139	9,7

PROMEDIO		7,064	PROMEDIO		9,686
63	93	6,49	63	134	9,35
	89	6,21		133	9,28
	99	6,91		132	9,21
	97	6,77		133	9,28
	104	7,26		131	9,14
PROMEDIO		6,728	PROMEDIO		9,252
66	97	6,77	66	127	8,86
	88	6,14		123	8,58
	101	7,05		126	8,79
	83	5,79		127	8,86
	93	6,49		126	8,79
PROMEDIO		6,448	PROMEDIO		8,776
69	87	6,07	69	114	7,95
	87	6,07		117	8,16
	88	6,14		116	8,09
	90	6,28		118	8,23
	89	6,21		120	8,36
PROMEDIO		6,154	PROMEDIO		8,158
72	81	5,65	72	106	7,4
	84	5,86		107	7,47
	82	5,72		106	7,4
	82	5,72		106	7,4
	86	6		105	7,33
PROMEDIO		5,79	PROMEDIO		7,4
75	66	4,61	75	90	6,28
	72	5,02		91	6,35
	63	4,4		92	6,42
	70	4,88		90	6,28
	64	4,47		94	6,56
PROMEDIO		4,676	PROMEDIO		6,378
78	52	3,63	78	74	5,16
	58	4,05		71	4,95
	61	4,26		76	5,3
	57	3,98		75	5,23
	50	3,49		73	5,09
PROMEDIO		3,882	PROMEDIO		5,146
81	34	2,37	81	50	3,49
	37	2,58		53	3,7
	40	2,79		66	4,61
	48	3,35		52	3,63
	38	2,65		55	3,84
PROMEDIO		2,748	PROMEDIO		3,854

84	18	1,26	84	27	1,88
	16	1,12		30	2,09
	22	1,54		30	2,09
	21	1,47		28	1,95
	17	1,19		31	2,16
PROMEDIO		1,316	PROMEDIO		2,034
CAMBIO DE DIRECCION					
105	10	0,7	105	22	1,54
	14	0,98		23	1,6
	15	1,05		19	1,33
	9	0,63		13	0,91
	4	0,28		17	1,19
PROMEDIO		0,728	PROMEDIO		1,314
108	26	1,81	108	47	3,28
	32	2,23		41	2,86
	33	2,3		48	3,35
	29	2,02		48	3,35
	36	2,51		49	3,42
PROMEDIO		2,174	PROMEDIO		3,252
111	48	3,35	111	66	4,61
	56	3,91		68	4,74
	43	3		68	4,74
	54	3,77		67	4,68
	51	3,56		69	4,81
PROMEDIO		3,518	PROMEDIO		4,716
114	56	3,91	114	87	6,07
	60	4,19		88	6,14
	63	4,4		86	6
	64	4,47		84	5,86
	55	3,84		89	6,21
PROMEDIO		4,162	PROMEDIO		6,056
117	77	5,37	117	104	7,26
	64	4,47		102	7,2
	75	5,23		100	6,98
	69	4,81		103	7,19
	81	5,65		102	7,12
PROMEDIO		5,106	PROMEDIO		7,15
120	77	5,37	120	115	8,02
	84	5,86		114	7,95
	93	6,49		113	7,88
	72	5,02		112	7,82
	79	5,51		112	7,82
PROMEDIO		5,65	PROMEDIO		7,898

123	90	6,28	123	123	8,58
	78	5,44		124	8,65
	89	6,21		125	8,72
	89	6,21		124	8,65
	92	6,42		123	8,58
PROMEDIO		6,112	PROMEDIO		8,636
126	105	7,33	126	132	9,21
	110	7,68		131	9,14
	113	7,88		129	9
	92	6,42		131	9,14
	97	6,77		133	9,28
PROMEDIO		7,216	PROMEDIO		9,154
129	103	7,19	129	138	9,63
	105	7,33		135	9,48
	104	7,26		137	9,56
	102	7,12		138	9,63
	103	7,19		137	9,56
PROMEDIO		7,218	PROMEDIO		9,572
132	98	6,84	132	139	9,7
	108	7,54		140	9,77
	115	8,02		145	10,12
	116	8,09		143	9,98
	111	7,75		144	10,05
PROMEDIO		7,648	PROMEDIO		9,924
135	111	7,75	135	146	10,19
	112	7,82		146	10,19
	113	7,88		146	10,19
	110	7,65		144	10,05
	102	7,12		146	10,19
PROMEDIO		7,644	PROMEDIO		10,162
138	104	7,26	138	148	10,33
	110	7,68		150	10,47
	115	8,02		148	10,33
	98	6,84		149	10,4
	111	7,75		148	10,33
PROMEDIO		7,51	PROMEDIO		10,372
141	113	7,88	141	154	10,75
	108	7,54		154	10,75
	126	8,79		154	10,75
	118	8,23		156	10,89
	123	8,58		156	10,89
PROMEDIO		8,204	PROMEDIO		10,806
144	124	8,65	144	160	11,16

	117	8,16		161	11,23
	116	8,09		158	11,02
	117	8,16		163	11,37
	122	8,51		161	11,23
PROMEDIO		8,314	PROMEDIO		11,202
147	123	8,58	147	161	11,23
	103	7,19		162	11,3
	112	7,82		161	11,23
	119	8,3		162	11,3
	120	8,37		160	11,16
PROMEDIO		8,052	PROMEDIO		11,244
150	115	8,02	150	162	11,3
	123	8,58		164	11,44
	125	8,72		163	11,37
	117	8,16		163	11,37
	119	8,3		162	11,3
PROMEDIO		8,356	PROMEDIO		11,356

Tabla N 1 Caracterización de motores y encoder

ANEXO O – PRUEBA DE CORRIENTE PARA LOS MOTORES DC

	CORRIENTE (mA)	
Pos	Motor Derecho	Motor Izquierdo
25	170	200
30	170	200
35	170	200
40	170	200
45	165	190
50	155	180
55	150	170
60	145	170
65	130	150
70	120	130
75	100	100
80	80	70
85	30	27
90	0	0
95	0	0
100	20	20
105	50	55
110	90	100
115	110	140
120	130	160
125	140-150	190
130	160	200
135	160	210
140	170	220
145	180	220
150	180	220
155	180	220
160	180	220

Tabla O 1 Corriente de los motores

ANEXO P – CÓDIGO FINAL EN ARDUINO UNO

DIAGRAMA DE FLUJO DEL PROGRAMA EN ARDUINO

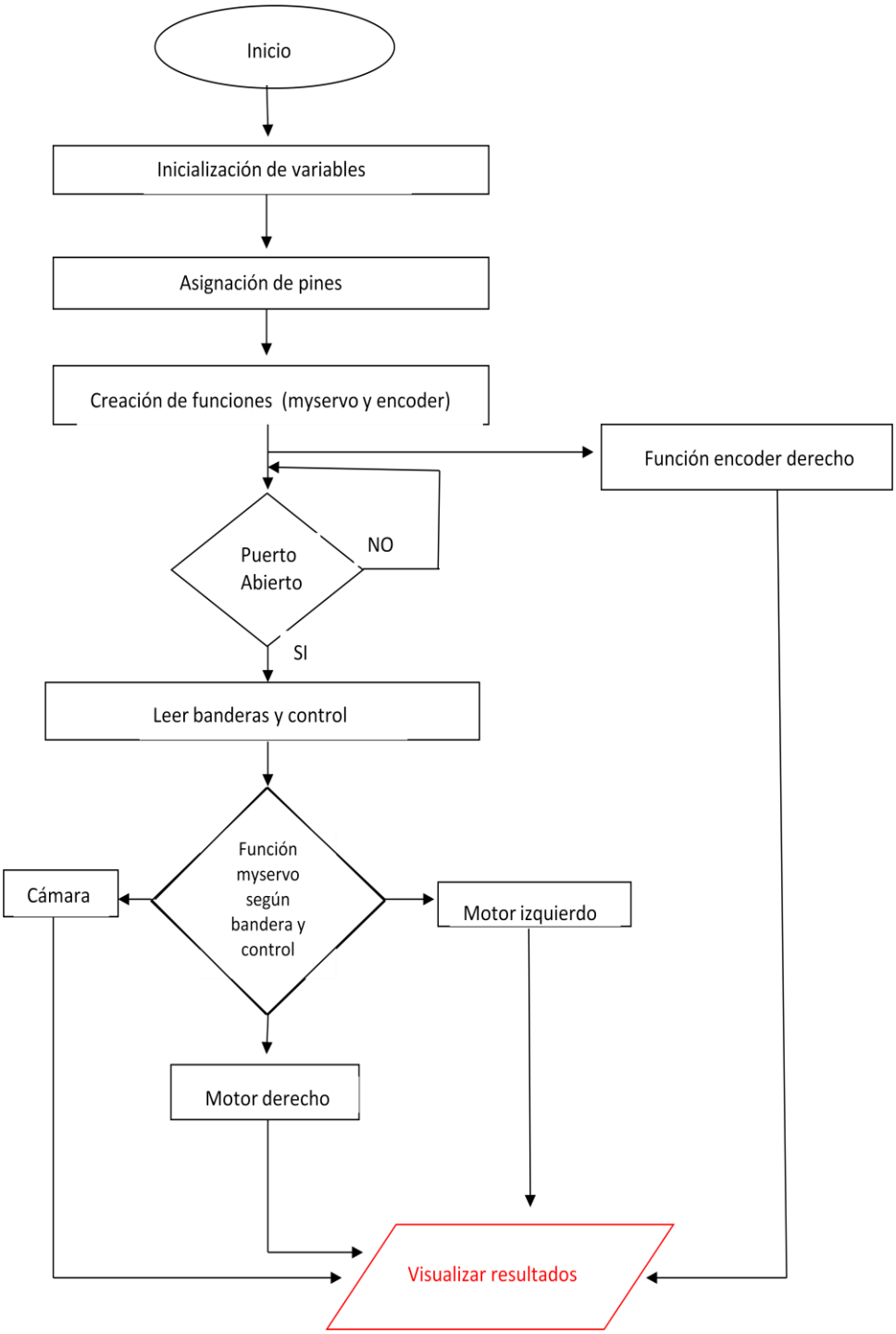


Figura P1 Diagrama de Flujo General del Programa

```
#include <Servo.h>

Servo myservo;      // Se crea el servomotor
Servo myservo_2;   // Se crea el motor izquierdo
Servo myservo_3;   // Se crea el motor derecho
int pos;   // Variable para la posición de la cámara
int pos_2;   // Variable para la velocidad del motor izquierdo
int pos_3;   // Variable para la velocidad del motor derecho
char bandera;  // Señal de control que indica el dato enviado
int control;  // El valor que llega
int analogPin=1;  // Potenciómetro conectado al pin A1
int digitalPin=12; // Limite Inferior conectado al pin 12
int digitalPin2=7; // Limite superior conectado al pin 7
int val=0;  // Se inicializa la variable val la cual entrega el valor del límite inferior
int val2=0;  // Se inicializa la variable val la cual entrega el valor del límite superior
int val3=0;   // Se inicializa la variable val la cual entrega el valor del potenciómetro
long previousMillis = 0;  // Se inicializa en 0
long interval = 1000;  // Se define el intervalo de tiempo de 1000 milisegundos (1 segundo)
long previousMillis1 = 0;   // Se inicializa en 0
long interval1 = 1000;   // Se define el intervalo de tiempo de 1000 milisegundos (1 segundo)
int valor; // Variable que guardará el número de pulsos del encoder izquierdo
int valor_2;  // Variable que guardará el número de pulsos del encoder derecho
int encoder0PinA = 4;  // El encoder del motor derecho se conecta al pin digital 4
int encoder0Pos = 0;  // Se inicializa el valor de los pulsos que se van a contar
int encoder0PinALast = LOW;    // Comienza en bajo la señal
int n = LOW;
float velocidad=0;  // Se inicializa el valor de la velocidad en 0
int encoder1PinB = 2;  // El encoder del motor izquierdo se conecta al pin digital 2
int encoder1Pos1 = 0;    // Se inicializa el valor de los pulsos que se van a contar
int encoder1PinBLast = LOW;   // Comienza en bajo la señal
int m = LOW;
float velocidad_2=0; // Se inicializa el valor de la velocidad en 0

void setup()
{
  myservo.attach(9);      // Servomotor para la cámara
  myservo_2.attach(11);   // Motor izquierdo
  myservo_3.attach(6);   // Motor derecho
  pinMode(encoder0PinA, INPUT);  // Se inicializa como entrada al pin del encoder derecho
  digitalWrite(encoder0PinA, HIGH);  // Se toma el valor cuando la señal este en alto
  pinMode(encoder1PinB, INPUT);   // Se inicializa como entrada al pin del encoder izquierdo
  digitalWrite(encoder1PinB, HIGH);  // Se toma el valor cuando la señal este en alto

  attachInterrupt(0, doEncoder, CHANGE);  // Función que se realiza al haber un cambio de pulso
  attachInterrupt(1, doEncoder_2, CHANGE); // Función que se realiza al haber un cambio
  Serial.end();  // Borrar el puerto serial
  Serial.begin (9600);  // Se especifica la velocidad de la transmisión de datos en serie.
  Serial.println("start");  // Se visualiza en el serial monitor ´start´ al empezar el programa
}

void loop()
 {
   if (Serial.available() > 0) {  // Si está abierto el puerto
```

```
  bandera= Serial.read();  // Se lee la bandera
  control = Serial.read();  // Se lee e control
  if (bandera == 'B') {   // Cuando la bandera es B
  myservo.write(control); // Se toma el valor para el servomotor de la cámara
  }
  if (bandera == 'C') {   //Cuando la bandera es C
  myservo_2.write(control);  // Se toma el valor para el motor izquierdo
  }
  if (bandera=='D') {   //Cuando la bandera es D
 myservo_3.write(control);  // Se toma el valor para el motor derecho
  }
  val=digitalRead (digitalPin);  // Límite Inferior
   Serial.println (val,2);
   val2=digitalRead (digitalPin2); // Límite Superior
   Serial.println (val2,2);
   val3=analogRead (analogPin);  // Potenciómetro
   Serial.println (val3);
    Serial.println (velocidad); // Se visualiza la velocidad del motor izquierdo
    Serial.println (velocidad_2);  // Se visualiza la velocidad del motor derecho
 }
 }

void doEncoder() {
  unsigned long currentMillis = millis();   // Comienza a contar el tiempo
    n = digitalRead(encoder0PinA);    // Se guarda en la variable ´n´ la lectura del encoder
      if ((encoder0PinALast == LOW) && (n == HIGH)) {    // cuando hay pulso, la variable cuenta
             encoder0Pos++;   // Conteo de la variable

              if(currentMillis - previousMillis > interval) {   // Después del segundo transcurrido,
                int valor = encoder0Pos; // ´valor´ guarda la cantidad de pulsos contados
                encoder0Pos=0;  // Se reinicia para volver a contar
                velocidad = (valor*2*3.14)/90;   // Se hace la fórmula para determinar la velocidad
                previousMillis = currentMillis;   // Se reinicia el tiempo
               }
   }
   encoder0PinALast = n;   // Se reinicia la variable ´n´
}

void doEncoder_2() {
  unsigned long currentMillis1 = millis();  // Comienza a contar el tiempo
    m = digitalRead(encoder1PinB);  // Se guarda en la variable ´m´ la lectura del encoder
      if ((encoder1PinBLast == LOW) && (m == HIGH)) {   // cuando hay pulso, la variable cuenta
             encoder1Pos1++;   // Conteo de la variable

              if(currentMillis1 - previousMillis1 > interval1) {     // Después del segundo transcurrido,
                int valor_2 = encoder1Pos1;   // ´valor_2´ guarda la cantidad de pulsos contados
                encoder1Pos1=0;  // Se reinicia para volver a contar
                 velocidad_2 = (valor_2*2*3.14)/90; // Se hace la fórmula para determinar la velocidad
                previousMillis1 = currentMillis1;  // Se reinicia el tiempo
               }
   }
   encoder1PinBLast = m  // Se reinicia la variable ´n´
 }
```

ANEXO Q – CÓDIGO EN MATLAB PARA EL CÁLCULO DE LOS VALORES TEÓRICOS DEL MODELO DINÁMICO

```
% FUNCIÓN PARA CALCULAR LOS VALORES TEÓRICOS DEL MODELO DINÁMICO

%Entradas
wencoderderecho=3.16;
wencoderizquierdo=2.32;
angulo=0.117;
r=0.0349; %radio de las llantas
l=0.23; % distancia entre las llantas de cada costado
matrizvelocidades= [r/2 r/2; r/l -r/l]*[2.5*wencoderderecho;
2.5*wencoderizquierdo] %Matriz que calcula [v w]
matrizfinal=[cos(angulo) 0; sin(angulo) 0; 0 1]*B; %Matriz que calcula salidas
[x1 x2 teta]

%Salidas
Velocidadlineal=matrizvelocidades(1,1)
VelocidadenX1= matrizfinal(1,1)
VelocidadenX2= matrizfinal(2,1)
```

ANEXO R - PRUEBAS DE VALIDACIÓN DEL MODELO DINÁMICO

CASO 1: Desplazamiento hacia adelante en línea recta

Promedio de Velocidades		
Pruebas	Velocidad angular motor derecho (rad/seg)	Velocidad angular motor izquierdo (rad/seg)
1	1,33	1,54
2	1,33	1,54
3	0,84	1,6
4	1,12	1,4
5	1,12	1,33
6	1,12	1,47
Promedio	1,143333333	1,48

Tabla R 1 Promedio de las velocidades angulares de cada motor para caso 1

MEDICIONES SOBRE EL PLANO CON ω derecho= 1,14 rad/s y ω izquierdo= 1,48 rad/s		
PRUEBA	Prueba AA	Prueba BB
TIEMPO [s]	6,2	3,6
ANGULO DE DESVIACIÓN (θ [radianes])	0	0
ANGULO DE DESVIACIÓN PROMEDIO (θ [radianes])	0	0
DISTANCIA RECORRIDA EN X1 (x_1) [m]	0,815	0,365
DISTANCIA RECORRIDA EN X2 (x_2) [m]	0	0
DISTANCIA DESDE EL ORIGEN [m]	0,815	0,365
VELOCIDAD LINEAL MEDIDA (v) [m/s]	0,131452	0,101389
VELOCIDAD LINEAL MEDIDA PROMEDIO (v) [m/s]	0,116420251	
VELOCIDAD LINEAL CALCULADA POR MODELO (v) [m/s]	0,1143	
VARIACIÓN X1 MEDIDA ($\dot{x_1}$) [m]	0,131452	0,101389
VARIACIÓN X1 PROMEDIO ($\dot{x_1}$) [m]	0,116420251	
VARIACIÓN X1 CALCULADA ($\dot{x_1}$) [m]	0,1143	
VARIACIÓN X2 MEDIDA ($\dot{x_2}$) [m]	0	0
VARIACIÓN X2 PRÁCTICA ($\dot{x_2}$) [m]	0	
VARIACIÓN X2 TEÓRICA ($\dot{x_2}$) [m]	0	

Tabla R 2 Mediciones sobre el plano para el caso 1

CASO 2: Desplazamiento hacia atrás en línea recta

Promedio de Velocidades		
Pruebas	Velocidad angular motor derecho (rad/s)	Velocidad angular motor izquierdo (rad/s)
1	0,77	1,19
2	1,05	1,05
3	0,42	1,05
4	0,84	1,26
5	0,77	1,12
6	0,77	1,05
Promedio	0,77	1,12

Tabla R 3 Promedio de las velocidades angulares para el caso 2

MEDICIONES SOBRE EL PLANO CON ω derecho= 0,77 rad/s y ω izquierdo= 1,12 rad/s		
PRUEBA	Prueba CC	Prueba DD
TIEMPO [s]	5.3	4.1
ANGULO DE DESVIACIÓN (θ [radianes])	0	0
ANGULO DE DESVIACIÓN PROMEDIO (θ [grados])	0	0
DISTANCIA RECORRIDA EN X1 (x_1) [m]	0.475	0.435
DISTANCIA RECORRIDA EN X2 (x_2) [m]	0	0
DISTANCIA DESDE EL ORIGEN [m]	0.475	0.435
VELOCIDAD LINEAL MEDIDA (v) [m/s]	0.089622642	0.106097561
VELOCIDAD LINEAL MEDIDA PROMEDIO (v) [m/s]	0.097860101	
VELOCIDAD LINEAL CALCULADA POR MODELO (v) [m/s]	0.0825	
VARIACIÓN X1 MEDIDA (x_1) [m]	0.089622642	0.106097561
VARIACIÓN X1 PROMEDIO (x_1) [m]	0.097860101	
VARIACIÓN X1 CALCULADA (x_1) [m]	0.0825	
VARIACIÓN X2 MEDIDA (x_2) [m]	0	0
VARIACIÓN X2 PROMEDIO (x_2) [m]	0	
VARIACIÓN X2 CALCULADA (x_2) [m]	0	

Tabla R 4 Mediciones sobre el plano para el caso 2

CASO 3: Desplazamiento hacia adelante con desviación a la derecha

Promedio de Velocidades		
Pruebas	Velocidad angular motor derecho (rad/s)	Velocidad angular motor izquierdo (rad/s)
1	2,37	3,07
2	2,09	3,14
3	2,16	3,28
4	2,65	3,21
5	2,44	3,14
6	2,23	3,14
Promedio	2,323333333	3,163333333

Tabla R 5 Promedio de las velocidades angulares para el caso 3

MEDICIONES SOBRE EL PLANO CON ω derecho= 2.32 rad/s y ω izquierdo= 3,16 rad/s		
PRUEBA	Prueba I	Prueba J
TIEMPO [s]	4,7	3,2
ANGULO DE DESVIACIÓN PROMEDIO (θ [radianes])	0,157	0,078
DISTANCIA RECORRIDA EN X1 (x_1) [m]	1,257	0,86
DISTANCIA RECORRIDA EN X2 (x_2) [m]	0,083	0,04
DISTANCIA DESDE EL ORIGEN [m]	1,259	0,86
VELOCIDAD LINEAL MEDIDA (v) [m/s]	0,267872	0,26875
VELOCIDAD LINEAL MEDIDA PROMEDIO (v) [m/s]	0,26831117	
VELOCIDAD LINEAL CALCULADA POR MODELO (v) [m/s]	0,2391	
VARIACIÓN X1 MEDIDA ($\dot{x_1}$) [m]	0,267447	0,26875
VARIACIÓN X1 PROMEDIO ($\dot{x_1}$) [m]	0,268098404	
VARIACIÓN X1 CALCULADA ($\dot{x_1}$) [m]	0,2374	
VARIACIÓN X2 MEDIDA ($\dot{x_2}$) [m]	0,01766	0,0125
VARIACIÓN X2 PROMEDIO ($\dot{x_2}$) [m]	0,015079787	
VARIACIÓN X2 CALCULADA ($\dot{x_2}$) [m]	0,0279	

Tabla R 6 Mediciones sobre el plano para el caso 3

CASO 4: Desplazamiento hacia adelante con desviación a la izquierda

Promedio de Velocidades		
Pruebas	Velocidad angular motor derecho (rad/s)	Velocidad angular motor izquierdo (rad/s)
1	6.63	5.09
2	6.63	5.09
3	6.35	4.19
4	6.35	4.47
5	6.7	5.3
6	6.91	5.51
Promedio	6.595	4.941666667

Tabla R 7 Promedio de las velocidades angulares para el caso 4

MEDICIONES SOBRE EL PLANO CON ω derecho= 6.595 rad/s y ω izquierdo= 4.941 rad/s		
PRUEBA	Prueba A	Prueba B
TIEMPO [s]	6.3	6
ANGULO DE DESVIACIÓN (θ[radianes])	0,349	0,418
ANGULO DE DESVIACIÓN PROMEDIO (θ [radianes])	0,349	0,418
DISTANCIA RECORRIDA EN X1 (x_1) [m]	3.072	3.125
DISTANCIA RECORRIDA EN X2 (x_2) [m]	0.41	0.43
DISTANCIA DESDE EL ORIGEN [m]	3.099	3.151
VELOCIDAD LINEAL MEDIDA (v) [m/s]	0.49	0.525083
VELOCIDAD LINEAL MEDIDA PROMEDIO (v) [m/s]	0.50849	
VELOCIDAD LINEAL CALCULADA POR MODELO (v) [m/s]	0.5032	
VARIACIÓN X1 MEDIDA ($\dot{x_1}$) [m]	0.487619	0.520833
VARIACIÓN X1 PROMEDIO ($\dot{x_1}$) [m]	0.50422619	
VARIACIÓN X1 CALCULADA ($\dot{x_1}$) [m]	0.4771	
VARIACIÓN X2 MEDIDA ($\dot{x_2}$) [m]	0.065079	0.071667
VARIACIÓN X2 PROMEDIO ($\dot{x_2}$) [m]	0.068373016	
VARIACIÓN X2 CALCULADA ($\dot{x_2}$) [m]	0.1602	

Tabla R 8 Mediciones sobre el plano para el caso 4

ANEXO S - CÓDIGO FINAL DE LA INTERFAZ EN MATLAB®

DIAGRAMA DE FLUJO DE LA INTERFAZ EN MATLAB®

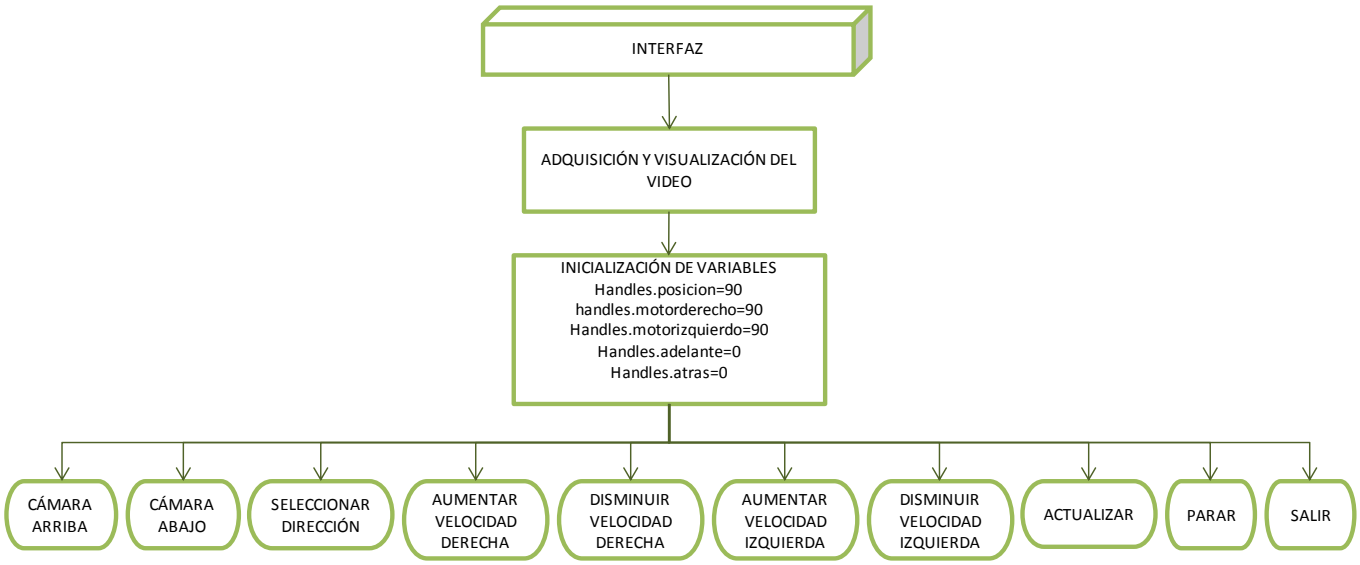


Figura S1 Diagrama de Flujo General del Programa

```
%CÓDIGO DE LA INTERFAZ DEL PROYECTO "DISEÑO DE UNA INTERFAZ PARA CONTROLAR
%EL MOVIMIENTO DE UN ROBOT (3-DOF) UTILIZANDO COMUNICACIONES INALÁMBRICAS

function varargout = videoymovimiento(varargin)
% VIDEOYMOVIMIENTO M-file for videoymovimiento.fig
%      VIDEOYMOVIMIENTO, by itself, creates a new VIDEOYMOVIMIENTO or raises
the existing
%      singleton*.
%
%      H = VIDEOYMOVIMIENTO returns the handle to a new VIDEOYMOVIMIENTO or
the handle to
%      the existing singleton*.
%
%      VIDEOYMOVIMIENTO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in VIDEOYMOVIMIENTO.M with the given input
arguments.
%
%      VIDEOYMOVIMIENTO('Property','Value',...) creates a new VIDEOYMOVIMIENTO
or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before videoymovimiento_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to videoymovimiento_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help videoymovimiento

% Last Modified by GUIDE v2.5 24-May-2011 15:50:21

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @videoymovimiento_OpeningFcn, ...
                   'gui_OutputFcn',  @videoymovimiento_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before videoymovimiento is made visible.
function videoymovimiento_OpeningFcn(hObject,eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to videoymovimiento (see VARARGIN)

handles.output = hObject;
handles.rgb = [];
handles.noback = [];
guidata(hObject, handles);

%
if strcmp(get(hObject,'Visible'),'off')
try
%Inicial código de adquisición de video
imaqreset
handles.vidobj = videoinput('winvideo');
start(handles.vidobj);
guidata(hObject, handles);
vidRes = get(handles.vidobj, 'VideoResolution');%Resolución del video
nBands = get(handles.vidobj, 'NumberOfBands');%Numero de bandas
hImage = image(zeros(vidRes(2), vidRes(1), nBands), 'Parent',...
handles.video_cam);
preview(handles.vidobj, hImage);
catch
%Si la cámara no esta conectada, poner una foto en su lugar.
%msgbox('NO HAY CÁMARA CONECTADA. Cargando Profile.jpg.')
hImage = image(imread('profile.jpg'), 'Parent',handles.video_cam);
end
end
axis off;

%Inicialización de Variables

handles.posicion = 90; %Control de la Cámara
guidata(hObject, handles);

handles.motorderecho=90; %Control del motor derecho
guidata(hObject, handles);

handles.motorizquierdo=90;%Control del motor izquierdo
guidata(hObject, handles);

handles.adelante=0; %Variable que indica si el movimiento es hacia adelante
guidata(hObject, handles);

handles.atras=0;%Variable que indica si el movimiento es hacia atras
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = videoymovimiento_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in camarabajo.
function camarabajo_Callback(hObject, eventdata, handles)
% hObject    handle to camarabajo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

sumando=5; %Valor de la variación gradual
handles.posicion1=handles.posicion + sumando;
if (handles.posicion1 < 125) %Mientras este menor a su máximo valor, aumenta
el valor de control
 handles.posicion = handles.posicion1;
else
    handles.posicion1=125;
end
guidata(hObject, handles);

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
x=handles.posicion1;

fwrite(PS,'A','uchar');
fwrite(PS,'B','uchar'); %Control que indica envio de control para servomotor
fwrite(PS,x,'uchar'); %posicion camara
%Se mantiene el mismo orden de recepción de datos durante todo el programa
handles.variable1= fscanf(PS,'%c');%Limite Inferior
handles.variable2= fscanf(PS,'%c');%Limite Superior
handles.variable3= fscanf(PS,'%c');%Potenciometro
handles.variable4= fscanf(PS,'%c');%Velocidad Encoder Derecho
handles.variable5= fscanf(PS,'%c');%Velocidad Encoder Izquierdo

limiteinferior= str2double(handles.variable1);%Obtiene valor del limite
inferior
limitesuperior= str2double(handles.variable2);%Obtiene valor del limite
superior

if (limiteinferior == 0) %Condición que indica si llego a su limite inferior
 set(handles.text5,'String','LLego a su limite inferior');
else
  set(handles.text5,'String','');
end
guidata(hObject, handles);
if (limitesuperior == 0) %Condición que indica si llegoa su limite superior
 set(handles.text3,'String','LLego a su limite superior');
 else
 set(handles.text3,'String','');
end
guidata(hObject, handles);
```

```
pote= str2double(handles.variable3);%Obtención del valor del potenciómetro
grados= 0.282*grados1-139.7; %Calculo de los grados equivalentes
set(handles.text4,'String',grados);%Muestra en la interfaz los grados
set(handles.text16,'String',handles.variable4);
set(handles.text15,'String',handles.variable5);
fclose(PS);
delete(PS);
clear PS;

% --- Executes on button press in camaraarriba.
function camaraarriba_Callback(hObject, eventdata, handles)
% hObject    handle to camaraarriba (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
sumando=5;%Valor de la variación gradual
handles.posicion1=handles.posicion - sumando;
if (handles.posicion1 > 40) %mientras este por encima de su minimo valor,
realizar variación
 handles.posicion = handles.posicion1;
else
    handles.posicion1=40;
end
guidata(hObject, handles);

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
x=handles.posicion1;

fwrite(PS,'A','uchar');
fwrite(PS,'B','uchar'); %Control que indica envio de control para servomotor
fwrite(PS,x,'uchar'); %posicion camara

handles.variable1= fscanf(PS,'%c');%Limite Inferior
handles.variable2= fscanf(PS,'%c');%Limite Superior
handles.variable3= fscanf(PS,'%c');%Potenciometro
handles.variable4= fscanf(PS,'%c');%Velocidad Encoder Derecho
handles.variable5= fscanf(PS,'%c');%Velocidad Encoder Izquierdo

limiteinferior= str2double(handles.variable1);%Obtiene valor del limite
inferior
limitesuperior= str2double(handles.variable2);%Obtiene valor del limite
superior

if (limiteinferior == 0) %Condición que indica si llego a su limite inferior
 set(handles.text5,'String','LLego a su limite inferior');
```

```
else
  set(handles.text5,'String','');
end
guidata(hObject, handles);
if (limitesuperior == 0) %Condición que indica si llegoa su limite superior
 set(handles.text3,'String','LLego a su limite superior');
 else
 set(handles.text3,'String','');
end
guidata(hObject, handles);

pote= str2double(handles.variable3);%Obtiene el valor del potenciómetro
grados= 0.345*grados1-160.5;%Calcula la ubicación en grados
set(handles.text4,'String',grados);
set(handles.text16,'String',handles.variable4);
set(handles.text15,'String',handles.variable5);
fclose(PS);
delete(PS);
clear PS;

% --- Executes on button press in salir.
function salir_Callback(hObject, eventdata, handles)
% hObject    handle to salir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close (gcbf)


% --- Executes on button press in aumentarvelocidadizq.
function aumentarvelocidadizq_Callback(hObject, eventdata, handles)
% hObject    handle to aumentarvelocidadizq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.adelante==1)
%El motor izquierdo aumenta su velocidad hacia adelante con valores en 45 y
%90.
sumando=1;
handles.motorizquierdo1=handles.motorizquierdo - sumando;

if (handles.motorizquierdo1 > 45)
 handles.motorizquierdo = handles.motorizquierdo1;
else
    handles.motorizquierdo1=45;
end
guidata(hObject, handles);
if (handles.motorizquierdo1 < 90)
 handles.motorizquierdo = handles.motorizquierdo1;
else
    handles.motorizquierdo1=90;
end
guidata(hObject, handles);

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
```

```
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize',6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);

z=handles.motorizquierdo1;
fwrite(PS,'A','uchar');
fwrite(PS,'C','uchar'); %Control
fwrite(PS,z,'uchar'); %motor izquierdo

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la velocidad del motor
por razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',velocidad11);
set(handles.text15,'String',velocidad22);

fclose(PS);
delete(PS);
clear PS;
end
guidata(hObject, handles);

if (handles.atras==1)
%El motor izquierdo aumenta su velocidad hacia atras con valores de control
%entre 90 y 130.
sumandoizq=1;
handles.motorizquierdo1=handles.motorizquierdo + sumandoizq;

if (handles.motorizquierdo1 < 130)
 handles.motorizquierdo = handles.motorizquierdo1;
else
    handles.motorizquierdo1=130;
end

if (handles.motorizquierdo1 > 90)
 handles.motorizquierdo = handles.motorizquierdo1;
else
    handles.motorizquierdo1=90;
end
delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
```

```
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
z=handles.motorizquierdo1;
fwrite(PS,'A','uchar');
fwrite(PS,'C','uchar'); %Control
fwrite(PS,z,'uchar'); %motor izquierdo

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la velocidad del motor
por razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',-velocidad11);
set(handles.text15,'String',-velocidad22);

fclose(PS);
delete(PS);
clear PS;
end
guidata(hObject, handles);

% --- Executes on button press in disminuirvelocidadizq.
function disminuirvelocidadizq_Callback(hObject, eventdata, handles)
% hObject    handle to disminuirvelocidadizq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.adelante==1)
%El motor izquierdo disminuye su velocidad hacia adelante aumentando el valor
%de control desde 40 hasta 90.
sumando=1;
handles.motorizquierdo1=handles.motorizquierdo + sumando;

if (handles.motorizquierdo1 > 40)
 handles.motorizquierdo = handles.motorizquierdo1;
else
    handles.motorizquierdo1=40;
end

if (handles.motorizquierdo1 < 90)
 handles.motorizquierdo = handles.motorizquierdo1;
```

```
else
    handles.motorizquierdo1=90;
end

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
z=handles.motorizquierdo1;
fwrite(PS,'A','uchar');
fwrite(PS,'C','uchar'); %Control
fwrite(PS,z,'uchar'); %motor izquierdo

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la velocidad del motor
por razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',velocidad11);
set(handles.text15,'String',velocidad22);
fclose(PS);
delete(PS);
clear PS;
end
guidata(hObject, handles);

if (handles.atras==1)
%El motor izquierdo disminuye su velocidad hacia atras disminuyendo su
%valor de control de 130 a 90.
sumandoizq=1;
handles.motorizquierdo1=handles.motorizquierdo - sumandoizq;

if (handles.motorizquierdo1 < 130)
 handles.motorizquierdo = handles.motorizquierdo1;
else
    handles.motorizquierdo1=130;
end

if (handles.motorizquierdo1 > 90)
 handles.motorizquierdo = handles.motorizquierdo1;
else
```

```
    handles.motorizquierdo1=90;
end

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
z=handles.motorizquierdo1;
fwrite(PS,'A','uchar');
fwrite(PS,'C','uchar'); %Control
fwrite(PS,z,'uchar'); %motor izquierdo

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la velocidad del motor
por razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',-velocidad11);
set(handles.text15,'String',-velocidad22);

fclose(PS);
delete(PS);
clear PS;
end
guidata(hObject, handles);

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1

direccion= get(hObject, 'Value');
switch direccion

    case 1 %Si no se indica el sentido no es posible el movimiento
```

```
  handles.adelante = 0;
  handles.atras=0;

    case 2 %Movimiento hacia adelante

  handles.adelante = 1; %Variable que indica que el movimiento es hacia
adelante
  handles.atras=0;
  handles.motorderecho=90;
  handles.motorizquierdo=90;

    case 3

  handles.adelante = 0;
  handles.atras=1;
  handles.motorderecho=90;
  handles.motorizquierdo=90;

end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in parar.
function parar_Callback(hObject, eventdata, handles)
% hObject    handle to parar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles.motorderecho=90; %Variable de motor derecho en 90 (valor de parado)
guidata(hObject, handles);
 handles.motorizquierdo=90;%Variable de motor izquierdo en 90 (valor de
parado)
 guidata(hObject, handles);

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera
```

```
fopen(PS);

fwrite(PS,'A','uchar');
fwrite(PS,'C','uchar'); %Control
fwrite(PS,90,'uchar'); %motor derecho
fwrite(PS,'D','uchar'); %Control
fwrite(PS,90,'uchar'); %motor derecho

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',velocidad11);
set(handles.text15,'String',velocidad22);

fclose(PS);
delete(PS);
clear PS;

% --- Executes on button press in aumentarvelocidadder.
function aumentarvelocidadder_Callback(hObject, eventdata, handles)
% hObject    handle to aumentarvelocidadder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.adelante==1)
%El motor derecho aumenta su velocidad hacia adelante aumentando el valor
%de control de 90 a 140
sumando=3;
handles.motorderecho1=handles.motorderecho + sumando;

if (handles.motorderecho1 < 140)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=140;
end

if (handles.motorderecho1 > 90)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=90;
end

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
```

```
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
y=handles.motorderecho;

fwrite(PS,'A','uchar');
fwrite(PS,'D','uchar'); %Control
fwrite(PS,y,'uchar'); %motor derecho

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',velocidad11);
set(handles.text15,'String',velocidad22);

fclose(PS);
delete(PS);
clear PS;

end
guidata(hObject, handles);

if (handles.atras==1)
%El motor derecho aumenta su velocidad hacia atras disminuyendo el valor
%de control de 90 a 45
sumandoder=3;
handles.motorderecho1=handles.motorderecho - sumandoder;

if (handles.motorderecho1 > 45)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=45;
end

if (handles.motorderecho1 < 90)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=90;
end

delete(instrfindall);
PS=serial('COM1');
```

```
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
y=handles.motorderecho;

fwrite(PS,'A','uchar');
fwrite(PS,'D','uchar'); %Control
fwrite(PS,y,'uchar'); %motor derecho

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',-velocidad11);
set(handles.text15,'String',-velocidad22);
fclose(PS);
delete(PS);
clear PS;

end
guidata(hObject, handles);

% --- Executes on button press in disminuirvelocidadder.
function disminuirvelocidadder_Callback(hObject, eventdata, handles)
% hObject    handle to disminuirvelocidadder (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if (handles.adelante==1)
%El motor derecho disminuye su velocidad disminuyendo el valor de control
%desde 140 a 90.
sumando=3;
handles.motorderecho1=handles.motorderecho - sumando;
if (handles.motorderecho1 < 140)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=140;
end
if (handles.motorderecho1 > 90)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=90;
```

```
end
delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera
fopen(PS);
y=handles.motorderecho;
fwrite(PS,'A','uchar');
fwrite(PS,'D','uchar'); %Control
fwrite(PS,y,'uchar'); %motor derecho

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');
%Adquiere la velocidad de las llantas multiplicando por la razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',velocidad11);
set(handles.text15,'String',velocidad22);
fclose(PS);
delete(PS);
clear PS;
end
guidata(hObject, handles);

if (handles.atras==1)
%El motor derecho disminuye su velocidad disminuyendo el valor de control
%desde 140 a 90.
sumandoder=3;
handles.motorderecho1=handles.motorderecho + sumandoder;

if (handles.motorderecho1 > 45)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=45;
end
if (handles.motorderecho1 < 90)
 handles.motorderecho = handles.motorderecho1;
else
    handles.motorderecho1=90;
end

delete(instrfindall);
PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
```

```
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
%set(PS,'Terminator','CR/LF');% "c" caracter con que finaliza el envío
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
%set(PS,'Timeout',5); % 5 segundos de tiempo de espera

fopen(PS);
y=handles.motorderecho;

fwrite(PS,'A','uchar');
fwrite(PS,'D','uchar'); %Control
fwrite(PS,y,'uchar'); %motor derecho

handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');

%Adquiere la velocidad de las llantas multiplicando la velocidad de los
motores por la razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',-velocidad11);
set(handles.text15,'String',-velocidad22);

fclose(PS);
delete(PS);
clear PS;
end
guidata(hObject, handles);

% --- Executes on button press in actualizar.
function actualizar_Callback(hObject, eventdata, handles)
% hObject    handle to actualizar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Actualiza el valor de los sensores
delete(instrfindall);

PS=serial('COM1');
set(PS,'Baudrate',9600); % se configura la velocidad a 9600 Baudios
set(PS,'StopBits',1); % se configura bit de parada a uno
set(PS,'DataBits',8); % se configura que el dato es de 8 bits, debe estar
entre 5 y 8
set(PS,'Parity','none'); % se configura sin paridad
set(PS,'OutputBufferSize',4); % "n" es el número de bytes a enviar
set(PS,'InputBufferSize' ,6); % "n" es el número de bytes a recibir
```

```
fopen(PS);
handles.variable1= fscanf(PS,'%c');
handles.variable2= fscanf(PS,'%c');
handles.variable3= fscanf(PS,'%c');
handles.variable4= fscanf(PS,'%c');
handles.variable5= fscanf(PS,'%c');

if (limiteinferior == 0)
 set(handles.text5,'String','LLego a su limite inferior');
else
  set(handles.text5,'String','');
end
guidata(hObject, handles);
if (limitesuperior == 0)
 set(handles.text3,'String','LLego a su limite superior');
 else
 set(handles.text3,'String','');
end
guidata(hObject, handles);
%Adquiere la velocidad de las llantas multiplicando la velocidad de los
motores por la razon de
%transmisión
velocidad1=str2double(handles.variable4);
velocidad11=2.5*velocidad1;
velocidad2=str2double(handles.variable5);
velocidad22=2.5*velocidad2;

set(handles.text16,'String',velocidad11);
set(handles.text15,'String',velocidad22);
fclose(PS);
delete(PS);
clear PS;
```

ANEXO T - COSTOS DEL PROYECTO

COSTOS FINALES DEL PROYECTO					
Sección del Robot	**Componentes**	**Cantidad**	**Valor en Pesos**	**Valor en Dólares**	**Sitio Web**
CONSTRUCCIÓN ROBOT	Batería 9.6 V	1	$ 36.611,69	$ 19,99	http://www.vexrobotics.com/
	Wheel (4 pack) 2.75"	1	$ 18.296,69	$ 9,99	
	Servo Motor	1	$ 36.611,69	$ 19,99	
	Motor	2	$ 36.611,69	$ 19,99	
	Sensor Limite (2)	1	$ 23.791,19	$ 12,99	
	Gear Kit (engranajes)	1	$ 23.791,19	$ 12,99	
	Advance Motion Kit (conectores)	1	$ 109.871,69	$ 59,99	
	Metal and Hardware kit	2	$ 146.501,69	$ 79,99	
	Tornillos (pack 100)	1	$ 14.633,69	$ 7,99	
TRANSMISIÓN- RECEPCIÓN DE VIDEO	Mini cámara infrarroja inalámbrica con receptor de video	1	$ 99.800,00	$ 54,49	www.jacanas.com
	Batería 9 V	1	$ 10.000,00	$ 5,49	
	Capturadora de video - Sensoray Model 2250S	1	$ 487.179,00	$ 266,00	http://www.sensoray.com/products/2250.htm
SISTEMA DE PROGRAMACIÓN Y COMUNICACIÓN	Tarjeta Arduino UNO	1	$ 80.000,00	$ 43,68	www.tiendaderobotica.com
	Modulo Arduino - Xbee Shield	1	$ 43.000,00	$ 23,48	
	Tarjetas Xbee	2	$ 75.000,00	$ 40,95	
	Tarjeta Xbib-R-DEV	1	$ 71.000,00	$ 38,77	
TOTAL			$ 1.305.200,17	$ 712,64	Conversión consultada el 26 de Mayo de 2011 en www.google.com

Tabla T1 Costos del proyecto