ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERIA EN ELECTRICIDAD Y COMPUTACION

“Uso de MATLAB y SIMULINK para el control de robots y la observación de sensores de sonido y tacto”

INFORME DE MATERIA DE GRADUACIÓN

Previo a la obtención del título de:

INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES

INGENIERO EN ELECTRICIDAD ESPECIALIZACION ELECTRONICA y AUTOMATIZACION INDUSTRIAL

Presentado por:

Jonathan Alexander Luna Consuegra

Nelly Colombia Zambrano Rosero

GUAYAQUIL – ECUADOR

AÑO 2009

AGRADECIMIENTO

Deseamos expresar nuestro agradecimiento principalmente a Dios por bendecirnos con la disciplina y la fuerza necesaria para llevar a cabo este proyecto de graduación.

También a nuestros padres que nos han apoyado para lograr la culminación de esta meta profesional.

Y a todos y cada uno de los Ingenieros que intervinieron en nuestro desarrollo y transformación de estudiantes a profesionales, además un reconocimiento especial al Ing. Carlos Valdivieso por la guía brindada a lo largo de este periodo de preparación del proyecto.

DEDICATORIA

Dedicamos este proyecto de graduación a Dios y a nuestros padres por el apoyo y los valores inculcados en nosotros y que hoy dan como resultado el gran logro que significa la culminación de nuestra vida de estudiantes para convertirnos en responsables profesionales de éxito.

DECLARACIÓN EXPRESA

La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL.

____

Jonathan Alexander Luna Consuegra

____

Nelly Colombia Zambrano Rosero

TRIBUNAL DE GRADUACIÓN

____

Ing. Carlos Valdivieso A.
Director de Tesis

____

Ing. Hugo Villavicencio V.
Delegado del Decano

____

Ing. Jorge Aragundi
Sub-Decano de la FIEC

RESUMEN

El desarrollo de este proyecto busca construir un robot que pueda ser controlado desde una PC vía bluetooth, con un programa desarrollado en MATLAB.

El robot realizará un monitoreo de un área de trabajo, basándose en el nivel de ruido presente, el mismo que se asumirá constante para funcionamiento normal y se definirán niveles mínimo y máximo para generar alarmas.

Así mismo estamos asumiendo que dicho robot tendrá una vía delimitada para transitar por lo que esta debería permanecer libre de obstáculos, sin embargo en virtud de que podría por algún evento fortuito haber un obstáculo en esta vía, cuenta con una rutina de evasión de obstáculos que le permitirá a través de sus sensores, detectarlos e intentar evadirlos y en caso de no lograrlo emitir una alarma al respecto.

Finalmente se realiza un envío de los datos medidos al final de cada ciclo y en el caso de detenerse, como parte de la rutina de detección de alarmas.

INDICE GENERAL

INDICE FIGURAS

INDICE TABLAS

INTRODUCCIÓN

En el mundo moderno el ruido es considerado una molestia sin embargo es un factor siempre presente en las industrias que generalmente se desea eliminar de muchas maneras pero en nuestro caso será una herramienta para la detección de fallas y la generación de alarmas.

Reconociendo la necesidad de las industrias de maximizar la eficiencia de sus procesos y anticipar cualquier falla en sus sistemas, este proyecto tiene como propósito brindar una herramienta de ayuda en el control del buen funcionamiento de la máquinaria a través de la medición del nivel de ruido y sus variaciones, permitiendo obtener alarmas ante cualquier cambio detectado en la operación.

En el desarrollo de este proyecto se utiliza un Lego Mindstorm programado para recorrer un área y medir el nivel de ruido en la misma, permitiendo hacer una comparación de los niveles que normalmente se deberían reportar con los obtenidos en la medición actual y generar una alarma ante cualquier cambio en el ruido del ambiente. Esto es posible gracias al uso de sensores de tacto y principalmente de sonido.

CAPITULO 1

1. GENERALIDADES

1.1. Antecedentes

Reconocemos que el control de máquinas de las industrias generalmente fue realizado por un ingeniero o técnico quien se encargaba de monitorear personalmente el funcionamiento de las mismas sin embargo esto incluía muchos riesgos para el personal que trabaja en este tipo de control como para la empresa debido a la gran probabilidad de que ocurra un accidente.

Estos accidentes llevaban a las industrias a enfrentar fuertes demandas que exigían grandes indemnizaciones, además de la pérdida humana, y de los defectos físicos que sufrían quienes tenían estos accidentes de los cuales algunos son irreversibles, las industrias también podían perder dinero en la reparación de las máquinas que se dañaban por una alarma de error tardía o en su defecto comprar nuevas máquinas.

Uno de los mayores causantes de estos accidentes es el ruido que se produce en los respectivos cuartos de máquinas.

El ruido afecta al hombre físicamente, psicológicamente y sociológicamente. El ruido puede dañar el oído, interferir la comunicación, causar molestias, producir cansancio y reducir la eficiencia.

Los ruidos intensos o la permanencia durante largo tiempo en un ambiente ruidoso pueden causar una reducción permanente de la sensibilidad auditiva debido a los daños producidos en los órganos sensoriales del oído interno. Este tipo de daños en el oído es irreversible y nunca más podrán ser recuperados.

El riesgo de dañar el oído incrementa con el nivel del sonido y con el tiempo de permanencia en un ambiente ruidoso, y también depende de las características del sonido. Además, la sensibilidad al ruido depende particularmente del propio individuo. En algunas personas se pueden producir daños auditivos en un breve intervalo de tiempo, en cambio, otras pueden trabajar en ambientes ruidosos sin sufrir daños auditivos demostrables, incluso durante toda su vida laboral.

Después de soportar durante un período breve de tiempo un ruido intenso y, a continuación, pasar a una zona tranquila, los sonidos suaves no se perciben. Este tipo de pérdida de audición se denomina temporal. Si el sonido no ha sido demasiado intenso o la duración de la exposición demasiado grande, se recupera la audición normal después de un período de reposo.

No es solamente la audición lo que puede resultar influenciado por un ruido intenso. El ruido puede afectar la circulación sanguínea, causar estrés y otros efectos psicológicos. El ruido industrial también está relacionado con otros problemas del ambiente industrial.

El ruido también puede aumentar el riesgo de accidente laboral, creando problemas de seguridad como consecuencia de que las señales audibles de alarma son menos perceptibles y las voces quedan enmascaradas.

1.2. Descripción general del sistema

Nuestro sistema consiste en un robot Lego Mindstorm programable que recorrerá un área previamente establecida en su programación, en su recorrido sensará los niveles de ruido producidos por las máquinas en funcionamiento que se encuentren en el área establecida, luego comparara estos niveles de ruido con los niveles que normalmente deberían existir en dicha área, los cuales serán incluidos en la programación previa del robot. Si al comparar los valores de ruido el robot detecta una variación en ellos enviará una alarma de acuerdo a la variación; es decir indicará si el nivel ha bajado lo que significa que alguna máquina ha dejado de funcionar, o si el nivel ha subido lo que significa que alguna máquina está funcionando a mayor nivel del establecido.

Nuestro sistema también enviará el registro del monitoreo realizado, los cuales serán presentados en un archivo EXCEL y servirán parar la presentación de los reportes de control de los diferentes cuartos de máquinas monitoreados.

Para el recorrido de la ruta trazada el robot usara servomotores y ruedas que vienen en el kit de Mindstorm que le van a brindar movilidad, además de los sensores que servirán para asegurarse de seguir la ruta trazada sin desviarse y evadir los obstáculos presentes en el camino.

La principal herramienta de nuestro sistema es el sensor de sonido del kit de Mindstorm el cual nos permitirá medir el nivel de sonido presente en el área y emitir una alarma en el caso de sobrepasar el valor de umbral que se fija en el programa.

Una vez fijada el área de recorrido, y definido el camino a seguir por el robot, con la ayuda de un área de color determinado, se utilizará el sensor de luz para guiar a nuestro robot a través del área siguiendo el camino pre-establecido y evadiendo cualquier obstáculo que se presente usando el sensor de contacto

1.3. Análisis y Justificación

En un ambiente industrial donde se trabaja con procesos pre-establecidos se presume un nivel de ruido constante, lo que permite advertir ante cualquier cambio en este nivel un desperfecto en las maquinarias que intervienen en este proceso, estos cambios de nivel de ruido podrían no ser percibidos inicialmente por el oído humano, terminando en ocasiones en fallas que van desde el simple hecho de detener la producción de un área hasta accidentes desafortunados convirtiéndose en ambos casos en pérdidas para a industria.

Haciendo estas consideraciones es perfectamente aplicable un sistema de alarma que esté basado en los cambios que se presentan en el nivel de ruido de los ambientes de trabajo en cada área de la empresa para reducir desde fallas imprevistas en maquinarias hasta accidentes graves para el personal, a tan solo un mantenimiento correctivo con el uso de herramientas sencillas tal como se presentará a lo largo del desarrollo de este proyecto.

CAPITULO 2

2. MARCO TEORICO

Fundamentos Teóricos

En el desarrollo de este proyecto el ruido desarrolla un papel importante, por lo tanto comenzaremos con una breve introducción sobre este tema, ya que es el parámetro sobre el cual vamos a trabajar, además de una revisión del sentido del tacto el mismo que también tendrá una participación relevante.

Ruido

En el medio ambiente se define como ruido todo sonido no deseado por el receptor.

En términos generales podemos definir al ruido como un sonido desagradable y molesto, con niveles excesivamente altos que son potencialmente nocivos para la audición.

En el ámbito de la comunicación sonora es aquel que no contiene información clara que el receptor no es capaz de identificar, individualizar o comprender.

Existen varios mecanismos de exposición a un ambiente ruidoso, esto puede ser de manera continua, fluctuante, intermitente o impulsiva y dependerá de ello la profundidad y la rapidez con la que se desarrolle la pérdida auditiva, aunque en cualquiera de estos casos, es lamentablemente irreversible.

Tipos de Ruido

Continuo constante: Es aquel cuyo nivel sonoro es prácticamente constante durante todo el período de medición, las diferencias entre los valores máximos y mínimos no exceden a 6 dB(A).

Continuo fluctuante: Es aquel cuyo nivel sonoro fluctúa durante todo el período de medición, presenta diferencias mayores a 6dB(A) entre los valores máximos y mínimos.

Intermitente: Presenta características estables o fluctuantes durante un segundo o más, seguidas por interrupciones mayores o iguales a 0,5 segundos.

Impulsivo o de impacto: Son de corta duración, con niveles de alta intensidad que aumentan y decaen rápidamente en menos de 1 segundo, presenta diferencias mayores a 35dB(A) entre los valores máximos y mínimos.

Con la finalidad de medir este parámetro se utilizará un kit Lego Mindstorms NXT que viene con sensores de sonido, tacto, luz y ultrasonido por lo que revisaremos brevemente como trabajan el kit y los sensores.

Tacto

El sentido del tacto o mecanorrecepción es aquel que permite a los organismos percibir cualidades de los objetos y medios como la presión, temperatura, aspereza o suavidad, dureza, etc. En el ser humano se considera uno de los cinco sentidos básicos.

Figura 2-1 Manos.

El sentido del tacto se halla principalmente en la piel, órgano en el que se encuentran diferentes clases de receptores nerviosos que se encargan de transformar los distintos tipos de estímulos del exterior en información susceptible de ser interpretada por el cerebro.

Tipos de receptores nerviosos

Los principales receptores nerviosos encargados que realizan esta función son los corpúsculos del tacto y los corpúsculos o discos de Merkel, Los corpúsculos son células nerviosas especializadas, situadas en diferentes capas de la piel.

- **Corpúsculos de Meissner:** Son de pequeño tamaño, miden entre 50 y 100 micras. Se encuentran formados por la terminación en espiral de un axón en el interior de una cápsula conjuntiva ovoidal, se encuentran en áreas sensibles como labios, yemas de dedos, pezones, palma de mano y especialmente en zonas donde no hay pelo. Están asociados con la capacidad de leer el lenguaje Braille o disfrutar de un beso, permiten reconocer la zona del cuerpo tocada y permiten identificar la textura de los objetos que actúan como estímulo. Estos receptores fueron descubiertos por un médico Alemán llamado Georg Meissner a quien deben su nombre.

- **Células o discos de Merkel:** son células capaces de actuar como receptores sensitivos ante la presión. Están concentradas predominantemente en las palmas de las manos y en las plantas de los pies. Las células de Merkel se ubican en la capa germinativa y se asocian a las células epiteliales por medio de desmosomas y su citoplasma se caracteriza por la abundancia en filamentos intermedios de citoqueratina.

  La cara basal, más externa, de las células de Merkel está asociada a una terminal nerviosa que adopta una forma de disco y que corresponde a la terminal de una fibra aferente, fibra que pertenece al axón de una neurona sensitiva, por el que está unido a ella.

  El citoplasma de las células de Merkel es capaz de sintetizar y acumular vesículas membranosas que contienen un material denso que almacena cromograninas asociadas a moléculas pequeñas. Cuando la célula de Merkel es deformada por una compresión de la epidermis, ésta tiende a liberar sus vesículas, vesículas que tienen una sustancia capaz de actuar como neurotransmisor y que pueden inducir a la depolarización del terminal nervioso asociado a ella, la cual eventualmente generaría la descarga de un potencial de acción en el axón de la neurona sensitiva.

Por sus características se considera que las células de Merkel pertenecen al sistema neuroendocrino difuso.

- **Corpúsculos de Pacini:** Son corpúsculos táctiles localizados en el nivel profundo de la hipodermis. Tienen forma ovalada, de medio milímetro de longitud aproximadamente y están formados por una cápsula gruesa de capas concéntricas en cuyo interior se encuentra la terminación nerviosa. Están ubicados en la zona profunda de la piel, sobre todo en los dedos de las manos y de los pies, pero son poco abundantes. Se tratan de dendritas (prolongaciones neuronales) encapsuladas en calvas (células de la neuroglía) rodeadas de tejido conectivo fibroso que detectan presiones y deformaciones de la piel. Los estímulos de los corpúsculos de Pacini tienen poca duración. Son los que responden al grado de presión y a las vibraciones que sentimos; nos permiten darnos cuenta de la consistencia y peso de los objetos y saber si son duros o blandos. En algunos casos, el peso se mide de acuerdo al esfuerzo que nos causa levantar un objeto. Por eso se dice que el peso se siente por el "sentido muscular". Éstos fueron descubiertos por Abraham Vater en 1741, que los bautizó como Nervæ Papilæ., aunque cayeron en el olvido hasta 1831, año en que Filipo Pacini, orientó su investigación hacia estos receptores nerviosos.

- **Corpúsculos de Ruffini:** Son terminaciones nerviosas, receptores de calor, alargadas y sensitivas que se hallan distribuidas en la dermis y en la región subcutánea, constituidos por finas fibras de colágeno (proteína de la piel) que termina en una especie de botón.

- **Corpúsculos de Krause:** son los encargados de registrar la sensación de frío, que se produce cuando entramos en contacto con un cuerpo o un espacio que está a menor temperatura que nuestro cuerpo. La sensibilidad es variable según la región de la piel que se considere. Son corpúsculos táctiles localizados en el nivel profundo de la hipodermis, parecidos a los de Paccini, pero más pequeños (miden 50 micras) y simplificados, como dendritas ramificadas y encapsuladas en una cavidad con forma de bulbo. Se encuentran extendidos por todo el cuerpo y en el tejido submucoso de la boca la nariz, ojos, genitales, etc.

- **Terminaciones Nerviosas Libres:** Son los receptores más simples y son las encargadas de transmitir el impulso al cerebro. Se considera que existen unos cuatro millones de puntos de dolor repartidos por el cuerpo humano, se reparten por la mayor parte de la superficie corporal, ya que son dendritas ramificadas entre las células epiteliales, especializadas en la recepción del dolor. No existen corpúsculos específicos que actúen como receptores del dolor sino que la sensación dolorosa es captada por terminaciones libres y cuyas ramificaciones se extienden por la capa profunda de la epidermis, (capa de Malphigi) habiendo lugares en la piel donde alcanzan concentraciones de 200 unidades por centímetro cuadrado.

- **Músculo Horripilador:** Cada uno de los pelos de nuestro cuerpo dispone de un músculo llamado horripilador que se inserta en él y que, cuando se contrae, mueve al pelo enderezándolo, con lo cual se nos pone la "carne de gallina".

Estos impulsos mecanorreceptores se conducen hacia la médula y encéfalo por medio de la vía del haz espinotalámico ventral. La información que lleva se integra para dar lugar a respuestas motoras medulares somáticas y autonómicas en forma de reflejos.

LEGO MINDSTORMS NXT

Lego Mindstorms NXT es un kit de robótica programable, este set permite a grupos de estudiantes, construir y programar soluciones robóticas de la vida real, donde el objetivo no sólo es crear estructuras robóticas, sino programarlas y darles vida para que realicen distintas acciones.

Figura 2-2 Sensores Lego Mindstorm

LEGO NXT (la nueva generación), Mindstorms NXT de Lego incluye motores servo, sensores de sonido, sensores de luz, sensores de tacto. También se puede comunicar vía Bluetooth, tiene puerto USB, pantalla de 100×64 píxeles y parlante de 8 Khz. entre otros. El cerebro del robot es un procesador ARM7 de 32 bit, al cual le puedes dar órdenes desde tu PC o Mac, a continuación el detalle.

- El NXT: Es el cerebro de un robot MINDSTORMS, controlado por computadora de ladrillo de LEGO que permite que el robot cobre vida y realice diferentes operaciones.
- El sensor de contacto: El sensor de contacto detecta cuando se está presionando por algo y cuando se libera de nuevo.
- Sensor de sonido: Hace que el robot escuche y reaccione ante el sonido.
- El sensor de luz: El sensor de luz permite que el robot distinga entre luz y oscuridad. Puede leer la intensidad de la luz en una habitación y medir la intensidad de la luz de las superficies de color.
- El sensor de ultrasonidos: El sensor ultrasónico permite al robot ver y detectar objetos. También se puede utilizar para hacer que su robot evite obstáculos, al sentido y medida a distancia, y detecte el movimiento.
- Servo Motores: Permite dar al robot la capacidad de movimiento. Si utiliza el movimiento en el bloque de LEGO MINDSTORMS NXT software para programar sus motores.

Características

- Lo mas innovador del nuevo software es que tiene editor de sonido, este graba cualquier tipo de sonido y después se programa para que el Ladrillo NXT pueda decir el sonido anteriormente grabado.
- Con el editor de imagen se puede subir una imagen para que el Ladrillo NXT lo represente en su pantalla.
- Aplicación del control remoto: Usando la conexión por medio de Bluetooth se puede tener mando directo sobre el robot desde la computadora
- Incluye 619 Piezas (Incluyendo las piezas electrónicas) entre ellas:
- 1 Sensor de Color RGB las características de éste son:
- Se puede utilizar en modo de sensor de color, usando el software estándar, puede detectar seis colores: negro, blanco, rojo, verde, azul y amarillo.
- Se puede usar en modo de lámpara: es posible controlar los LED's del sensor como verde, rojo y amarillo.
- Se puede utilizar en modo de sensor de luz: En este modo, actúa como el sensor de luz antigua: devuelve un valor de 0-100 donde 0 es más oscuro, y el 100 es más brillante. En este modo, también es posible elegir un LED de color deseado (de nuevo, rojo, azul o verde).
- Incluye 2 Sensores de tacto: Da al robot sentido del tacto
- Incluye 1 Sensor Ultrasónico: Este sensor podría ser resumido como los "ojos" del robot
- Incluye 3 Motores: Hacen que el robot se mueva
- Incluye el Ladrillo NXT: Resumido es el cerebro del robot

Componentes del sistema

En este caso debido al uso que le vamos a dar y al parámetro central de nuestra medición, o sea el ruido, daremos una descripción más amplia del sensor de sonido, sin embargo también se incluirá un resumen de los sensores de tacto y de luz los cuales también tendrán participación en el desarrollo de este proyecto.

El Sensor de Sonido

El sensor de sonido incluido en el kit de Lego Mindstorms NXT es bastante interesante y tiene varias aplicaciones.

Figura 2-3 Sensor de Sonido

El sensor solo detecta la "cantidad" de sonido y algún tipo de tono o modulación, pero aun así hay muchas aplicaciones ingeniosas que se le pueden dar.

Este sensor lee el sonido ambiental y nos regresa una medida de 0 a 100%. Podemos configurarlo para que lea Decibeles o Decibeles Ajustados. En términos muy simples los decibeles ajustados solo incluye sonidos que el oído humano puede escuchar, al contrario de los decibeles normales que podría incluir frecuencias que no podemos escuchar pero que el sensor de sonido capta.

Los decibeles se miden en una escala logarítmica medio complicada, por lo cual este sensor por defecto nos regresa, como ya se dijo, valores entre 0 y 100%. Estos valores corresponden más o menos a:

- **4-5%** Una casa silenciosa.
- **5-10%** Alguien hablando lejos.
- **10-30%** Es una conversación cerca del sensor o música en un volumen normal.
- **30-100%** Gente gritando o música a volumen alto.

Lo mas fácil para probar que lectura nos da este sensor es conectarlo a algún puerto del bloque programable y seleccionar la opción de "View" y dentro ya sea "Sound Sensor dB" o "Sound Sensor dBA" esto nos indicará la lectura continua que esta teniendo el sensor.

Figura 2-4 Prueba Sensor de Sonido

El Sensor de Luz

El sensor de luz es sin duda uno de los más útiles e interesantes de todo el kit del Lego Mindstorms NXT. Este sensor le permite a nuestro robot distinguir entre luz y obscuridad, midiendo la intensidad de la luz le permite a nuestro robot "ver" en blanco y negro.

Figura 2-5 Sensor de Luz

El sensor se puede usar en dos modos:

El primer modo detecta la luz del ambiente y se puede usar para detectar si un cuarto tiene la luz prendida o apagada, o la intensidad de la luz que entra por la ventana dependiendo de la hora del día o incluso para programar un robot que siga una fuente de luz.

En el segundo modo el mismo sensor emite una luz y luego mide que tanto rebota o refleja esta luz en las superficies. Este modo lo podemos usar para diferenciar el brillo de los colores en una superficie (¡el famoso robot seguidor de la línea negra se basa en este principio!). Incluso se ha usado para detectar la distancia hasta el suelo con una mejor precisión que con el sensor ultrasónico (en robots como el NXTWay).

El sensor nos da una lectura desde 0 (completa a obscuridad) hasta 100 (muy brillante).

Para poder probar el sensor de luz lo podemos conectar a un puerto del ladrillo programable:

1. Entramos al menú "View"
2. Seleccionamos "Light Sensor" y el puerto donde lo tenemos conectado.
3. Presionamos el botón naranja para correr el programa.

Y podemos probar el sensor de luz en diferentes superficies y colores para ver que tanto reflejan la luz. El tapete de pruebas ya tiene una barra de diferentes colores para que lo puedas probar.

El Sensor de Tacto

Figura 2-6 Sensor de Tacto

El sensor de tacto usado en conjunto con el NXT, el sensor de tacto detecta si está presionado o suelto, es capaz de contar múltiples toques. Permite el ensamble de un eje LEGO en el botón de tacto.

Figura 2-7 Modo de función del Sensor de Tacto

Herramientas de administración y desarrollo

Para desarrollar el programa de control que se utilizará usaremos las herramientas de trabajo Matlab y Simulink, por lo que es conveniente una breve introducción en el uso de los mismos.

MATLAB

Matlab es un software matemático muy poderoso, comercialmente disponible, desarrollado y distribuido por Mathworks, Inc. Es utilizado ampliamente en la academia y la industria debido a sus capacidades avanzadas, además posee una serie de herramientas que contienen funciones comúnmente usadas en ingeniería.

Todas herramientas facilitan el desarrollo de aplicaciones complejas de una forma más versátil.

Matlab es el nombre abreviado de “MATrix LABoratory”, es un programa especializado para realizar cálculos numéricos. Una de las capacidades más atractivas es la de poseer un lenguaje de programación propio.

Hoy en día, Matlab es usado en una variedad de áreas de aplicación incluyendo procesamiento de señales e imágenes, diseño de sistemas de control, ingeniería financiera e investigación médica.

En general, matrices y vectores son el corazón de Matlab, todos los datos son almacenados como vectores. Además de esto cuenta con herramientas de desarrollo como la GUI (Graphic User Interface), la cual permite generar entornos gráficos para las diferentes aplicaciones.

Características Del Entorno

Matlab provee acceso inmediato a las características gráficas especializadas requeridas en ingeniería y ciencias. Posee una serie de herramientas que facilitan el procesamiento de información, entre los cuales se destacan:

Representaciones 2-D y 3-D, incluyendo datos triangulados

y reticulados.

Representaciones 3-D quiver, ribbon, y stem

Control de fuentes, letras Griegas, símbolos, subíndices y

superíndices.

Selección expandida de símbolos marcadores de curvas.

Gráficos de torta, de barras 3-D y gráficos de barras

horizontales.

Gráficos 3-D y sólido modelado.

Representación de imágenes y archivos I/O.

Gráficos comentados.

Leer/Escribir archivos de datos Hierarchical Data Format

(HDF).

Presentación de OpenGL software y hardware.

Animación.

Soporte de colores verdaderos (24-bit RGB).

Fuentes múltiples de luz para superficies coloreadas.

Vista basada en cámara y control de perspectiva.

Iluminación Plana, Gouraud y Pong.

Soporte eficiente de imagen de datos de 8-bit.

Control de eje y cámara.

Propiedades de superficie y match.

Modelos de iluminación.

Control gráfico de objetos.

Impresión y representación de copias.

Formatos gráficos exportables.

MINDSTORMS NXT Toolbox para MATLAB. Está desarrollado para controlar robots NXT con MATLAB.

El Robotic Toolbox ofrece muchas funciones que son útiles en la robótica, tales como la cinemática, dinámica, y la generación de trayectoria. El toolbox es útil para la simulación, así como analizar los resultados obtenidos con los experimentos con robots reales.

El Toolbox se basa en un método muy general de la representación de la cinemática y la dinámica de los manipuladores y modelos de enlace-serial de los robots conocidos.

Ventajas del Toolbox son las siguientes:

• El código es bastante maduro y proporciona un punto de comparación con otras implementaciones de los mismos algoritmos,

• Dado que el código fuente está disponible, es un beneficio para la comprensión y la enseñanza.

El Toolbox proporciona funciones para la manipulación de tipos de datos tales como vectores, transformaciones homogéneas cuya posición y orientación necesitan ser representadas en 3-dimensiones. También cuenta con instalaciones para mostrar gráficamente la postura de cualquier robot. El robot se dibuja como una serie de segmentos de línea que une los orígenes de los marcos de enlace de referencia.

Simulink

Simulink es una herramienta para el modelaje, análisis y simulación de una amplia variedad de sistemas físicos y matemáticos, inclusive aquellos con elementos no lineales y aquellos que hacen uso de tiempos continuos y discretos. Como una extensión de MatLab, Simulink adiciona muchas características específicas a los sistemas dinámicos, mientras conserva toda la funcionalidad de propósito general de MatLab. Así Simulink no es completamente un programa separado de MatLab, sino un anexo a él. El ambiente de MatLab está siempre disponible mientras se ejecuta una simulación en Simulink.

Simulink tiene dos fases de uso: la definición del modelo y el análisis del modelo. La definición del modelo significa construir el modelo a partir de elementos básicos construidos previamente, tal como, integradores, bloques de ganancia o servomotores. El análisis del modelo significa realizar la simulación, linealización y determinar el punto de equilibrio de un modelo previamente definido.

Para simplificar la definición del modelo Simulink usa diferentes clases de ventanas llamadas ventanas de diagramas de bloques. En estas ventanas se puede crear y editar un modelo gráficamente usando el mouse. Simulink usa un ambiente gráfico lo que hace sencillo la creación de los modelos de sistemas.

Después de definir un modelo este puede ser analizado seleccionando una opción desde los menús de Simulink o entrando comandos desde la línea de comandos de MatLab.

Simulink puede simular cualquier sistema que pueda ser definido por ecuaciones diferenciales continuas y ecuaciones diferenciales discretas. Esto significa que se puede modelar sistemas continuos en el tiempo, discretos en el tiempo o sistemas híbridos.

Simulink usa diagramas de bloques para representar sistemas dinámicos. Mediante una interface gráfica con el usuario se pueden arrastrar los componentes desde una librería de bloques existentes y luego interconectarlos mediante conectores y alambre. La ventana principal de Simulink se activa escribiendo simulink en la línea de comandos de MatLab, y se muestra a continuación:

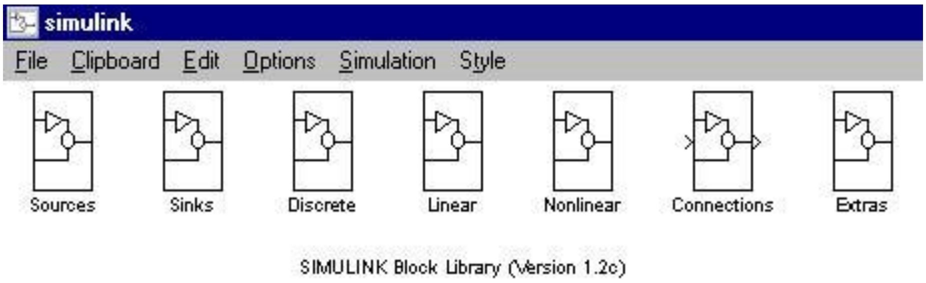


Figura 2-8 Librería de Bloque de Simulink

Haciendo doble click en cualquiera de las librerías presentes en esta ventana se abrirá otra ventana conteniendo una cantidad de bloques relativos a dicha librería.

Para realizar un sistema debe abrirse una nueva ventana de diagrama de bloques seleccionando la opción file del menú principal del Simulink y allí la opción new. En esta nueva ventana se colocarán todos los bloques interconectados que formarán el sistema deseado.

Como ejemplo se ha tomado un generador de ondas seno de la librería de fuentes "sources" y un osciloscopio de la librería "sinks", ambos se unieron mediante un conector usando el mouse. Este sistema se almacena como un archivo-m.

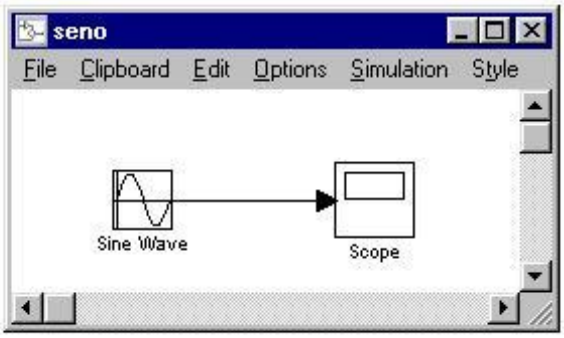


Figura 2-9 Ejemplo de uso de las librerías

Haciendo doble click sobre cada elemento del sistema se pueden ver y modificar sus características. Por ejemplo, al generador seno se le puede modificar su amplitud, frecuencia y fase. Al osciloscopio se le definen las escalas horizontal y vertical.

Para ejecutar el programa se usa la opción simulation en el menú de la ventana del archivo-m creado. En este submenú está la opción start que permite ejecutar el programa. También está la opción parameters que activa el panel de control de Simulink en donde se definen los métodos y parámetros usados para la simulación, tal como se muestra a continuación:

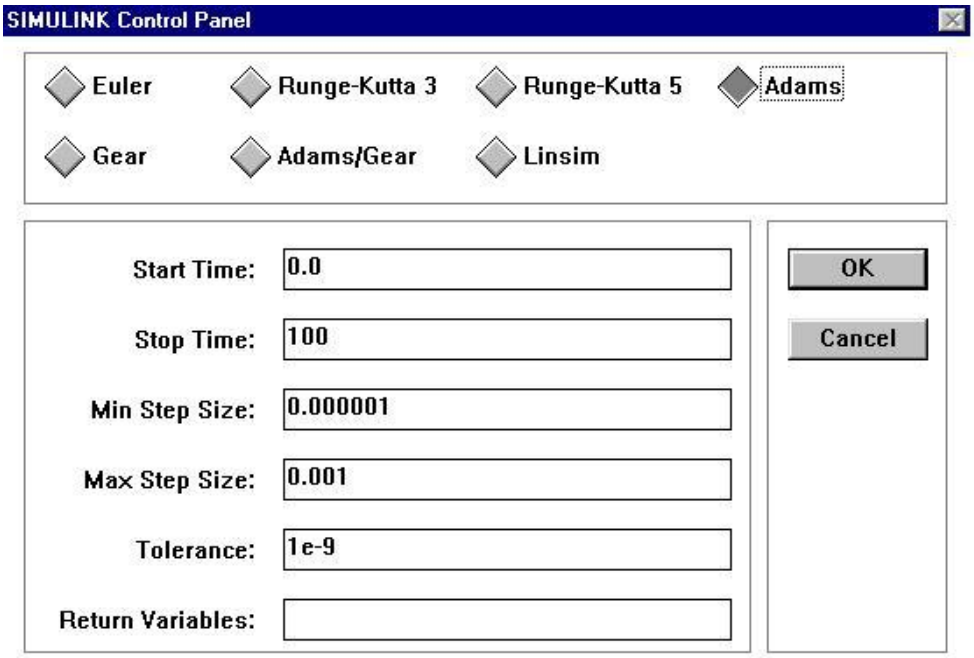


Figura 2-10 Panel de Control

Al ejecutar el programa seno.m creado mediante simulink, se puede observar la respuesta al hacer doble click en el osciloscopio.

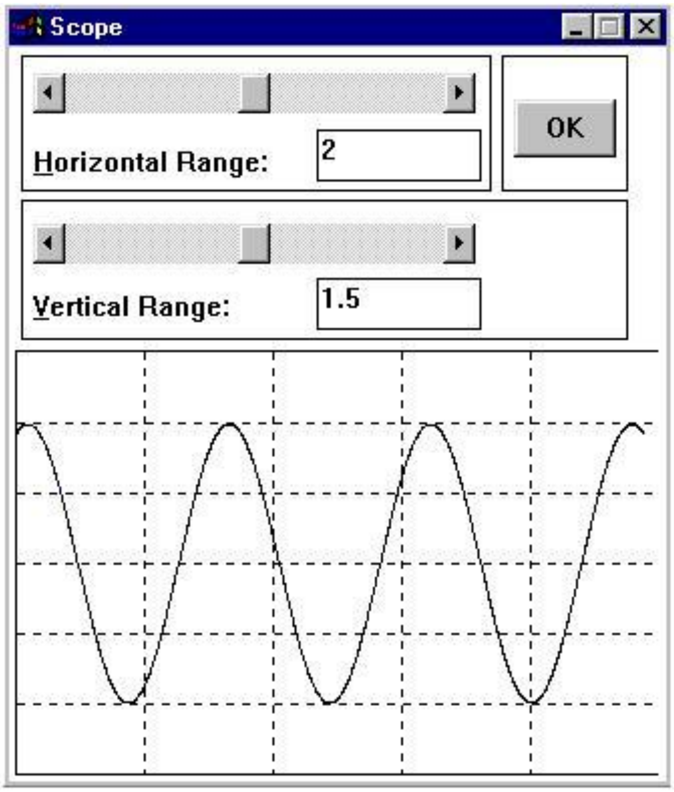


Figura 2-11 Ejemplo de Grafica

Existen numerosos bloques y funciones incorporados en las librerías de simulink que pueden ser empleados para simular cualquier sistema.

Por ejemplo, para implementar un sistema que emplea un controlador PID tenemos:

En este diagrama se tiene al bloque llamado PID que fue definido previamente y agrupado como uno solo. El contenido de dicho bloque se obtiene haciendo doble click sobre él.

Acelerador de Simulink

Para incrementar la velocidad de Simulink se debe instalar el acelerador "Accelerator". Este permite automáticamente generar una versión mejorada de los modelos los cuales correrán diez veces más rápido que el original. El acelerador puede ser usado sobre modelos continuos, discretos en el tiempo e híbridos.

El acelerador trabaja generando y compilando un código-C para un modelo dado. Una vez se completa la compilación, la simulación es ejecutada en la ventada de modelos de Simulink exactamente igual que antes sólo que más rápidamente. El propósito del acelerador es aumentar la velocidad de simulación.

Si el programa MatLab posee instalado el "Accelerator" podrá iniciarse la acción aceleradora seleccionando la opción simulation en el menú principal del Simulink y dentro de esta seleccionando la opción Accelerate. Esta acción es totalmente transparente en el sentido de que el incremento de la velocidad se presenta sin ningún otro requerimiento por parte del usuario.

Generador de código-C en Simulink

Una vez se ha creado un modelo dinámico en Simulink, se puede invocar el generador de código-C que permite convertir el diagrama de bloques implementado en un código C. Este puede ser útil para varios propósitos: puede ser usado para control en tiempo real, simulación en tiempo real o simulación acelerada en tiempo no real. Sus aplicaciones pueden ser control de movimiento, control de procesos, sistemas automotores, equipos médicos, robótica, etc.

El código-C es diseñado tal que puede ser ejecutado en tiempo real. No requiere ser escrito manualmente por un programador pues es creado a nivel de diagramas de bloques en Simulink. El código generado puede correr sobre un amplio rango de hardware ubicado en estaciones de trabajo, PC o microprocesadores. Este código es la forma en la que puede usare el Simulink para adquisición de datos.

Fundamento de la Tecnología Aplicada

Bluetooth

Origen del nombre y logo

Figura 2-12 Logo buetooth

El nombre procede del rey danés y noruego Harald Blåtand cuya traducción al inglés sería Harold Bluetooth (Diente Azul, aunque en lengua danesa significa 'de tez oscura') conocido por buen comunicador y por unificar las tribus noruegas, suecas y danesas.

De la misma manera, Bluetooth intenta unir diferentes tecnologías como las de las computadoras, los teléfonos móviles y el resto de periféricos.

El símbolo de Bluetooth es la unión de las runas nórdicas análogas a las letras H y B: ᚼ(Hagall) y ᛒ(Berkanan).

Usos y aplicaciones

Figura 2-13 Apple Mighty Mouse con tecnología Bluetooth

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basada en transceptores de bajo coste.

Gracias a este protocolo, los dispositivos que lo implementan pueden comunicarse entre ellos cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión lo permite. Estos dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" en referencia a su potencia de trasmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras.

Clase	Potencia máxima permitida	Potencia máxima permitida	Rango (aproximado)

	(mW)	(dBm)	
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2.5 mW	4 dBm	~10 metros
Clase 3	1 mW	0 dBm	~1 metro

Tabla 2-1 Clases de dispositivo y su compatibilidad

En la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Esto es así gracias a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte la mayor sensibilidad del dispositivo de clase 1 permite recibir la señal del otro pese a ser más débil.

Los dispositivos con Bluetooth también pueden clasificarse según su ancho de banda:

Versión	**Ancho de banda**

Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
UWB Bluetooth **(propuesto)**	53 - 480 Mbit/s

Tabla 2-2 Clases de dispositivo y su compatibilidad

Perfiles Bluetooth

Para utilizar Bluetooth, un dispositivo debe implementar alguno de los perfiles Bluetooth. Estos definen el uso del canal Bluetooth.

Lista de aplicaciones

Conexión sin cables vía OBEX.

- Transferencia de fichas de contactos, citas y recordatorios entre dispositivos vía OBEX.
- Reemplazo de la tradicional comunicación por cable entre equipos GPS y equipamiento médico.
- Controles remotos (tradicionalmente dominado por el infrarrojo).
- Enviar pequeñas publicidades desde anunciantes a dispositivos con Bluetooth. Un negocio podría enviar

publicidad a teléfonos móviles cuyo Bluetooth (los que lo posean) estuviera activado al pasar cerca.

- Las consolas Sony PlayStation 3 y Nintendo Wii incorporan Bluetooth, lo que les permite utilizar mandos inalámbricos.

Versiones

- Bluetooth v.1.1: en 1994, Ericsson inició un estudio para investigar la viabilidad de una nueva interfaz de bajo costo y consumo para la interconexión vía radio (eliminando así cables) entre dispositivos como teléfonos móviles y otros accesorios. El estudio partía de un largo proyecto que investigaba unos multicomunicadores conectados a una red celular, hasta que se llegó a un enlace de radio de corto alcance, llamado MC link. Conforme este proyecto avanzaba se fue haciendo claro que éste tipo de enlace podía ser utilizado ampliamente en un gran número de aplicaciones, ya que tenía como principal virtud que se basaba en un chip de radio.

- Bluetooth v.1.2: a diferencia de la 1.1, provee una solución inalámbrica complementaria para co-existir Bluetooth y Wi-Fi en el espectro de los 2.4 GHz, sin interferencia entre ellos. La versión 1.2 usa la técnica "Adaptive Frequency Hopping (AFH)", que ejecuta una transmisión más eficiente y un cifrado más seguro. Para mejorar las experiencias de los usuarios, la V1.2 ofrece una calidad de voz (Voice Quality - Enhanced Voice Processing) con menor ruido ambiental, y provee una más rápida configuración de la comunicación con los otros dispositivos bluetooth dentro del rango del alcance, como pueden ser PDAs, HIDs (Human Interface

Devices), computadoras portátiles, computadoras de escritorio, Headsets, impresoras y celulares.

- Bluetooth v.2.0: creada para ser una especificación separada, principalmente incorpora la técnica "Enhanced Data Rate" (EDR) que le permite mejorar las velocidades de transmisión en hasta 3Mbps a la vez que intenta solucionar algunos errores de la especificación 1.2.

- Bluetooth v.2.1: simplifica los pasos para crear la conexión entre dispositivos, además el consumo de potencia es 5 veces menor.

- Bluetooth v3.0 (mediados 2009): aumenta considerablemente la velocidad de transferencia. La idea es que el nuevo Bluetooth trabaje con WiFi, de tal manera que sea posible lograr mayor velocidad en los smartphones.

Futuro de Bluetooth

Ultra Wide Band Bluetooth

El 28 de marzo de 2006, el Bluetooth SIG anunció su intención de utilizar Ultra-Wideband/MB-OFDM como capa física para futuras versiones de Bluetooth.

La integración de UWB creará una versión de la tecnología Bluetooth con opción a grandes anchos de banda. Esta nueva versión permitirá alcanzar los requisitos de sincronización y transferencia de grandes cantidades de datos así como de contenidos de alta definición para dispositivos portátiles, proyectores multimedia, televisores y teléfonos VOIP.

Al mismo tiempo, la tecnología Bluetooth continuará satisfaciendo las necesidades de aplicaciones de muy bajo consumo como ratones, teclados o auriculares mono permitiendo a los dispositivos seleccionar la capa física más apropiada para sus requisitos.

Ultra Low Power Bluetooth

El 12 de junio de 2007, Nokia y el Bluetooth SIG anunciaron que Wibree formará parte de la especificación de Bluetooth como versión de muy bajo consumo. Sus aplicaciones son principalmente dispositivos sensores o mandos a distancia. Puede resultar interesante para equipamiento médico. La propuesta de Nokia es utilizar esta tecnología como enlace de bajo coste hasta un teléfono móvil que actúe de puerta de enlace hacia otras tecnologías como UMTS, Wi-Fi o incluso el mismo Bluetooth.

Información técnica

La especificación de Bluetooth define un canal de comunicación de máximo 720 kb/s (1 Mbps de capacidad bruta) con rango óptimo de 10 m (opcionalmente 100 m con repetidores).

La frecuencia de radio con la que trabaja está en el rango de 2,4 a 2,48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en Full Duplex con un máximo de 1600 saltos/s. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1Mhz; esto permite dar seguridad y robustez.

La potencia de salida para transmitir a una distancia máxima de 10 metros es de 0 dBm (1 mW), mientras que la versión de largo alcance transmite entre 20 y 30 dBm (entre 100 mW y 1 W).

Para lograr alcanzar el objetivo de bajo consumo y bajo costo, se ideó una solución que se puede implementar en un solo chip utilizando circuitos CMOS. De esta manera, se logró crear una solución de 9×9 mm y que consume aproximadamente 97% menos energía que un teléfono celular común.

El protocolo de banda base (canales simples por línea) combina conmutación de circuitos y paquetes. Para asegurar que los paquetes no lleguen fuera de orden, los slots pueden ser reservados por paquetes síncronos, un salto diferente de señal es usado para cada paquete. Por otro lado, la conmutación de circuitos puede ser asíncrona o síncrona. Tres canales de datos síncronos (voz), o un canal de datos síncrono y uno asíncrono, pueden ser soportados en un solo canal. Cada canal de voz puede soportar una tasa de transferencia de 64 kb/s en cada sentido, la cual es suficientemente adecuada para la transmisión de voz. Un canal asíncrono puede transmitir como mucho 721 kb/s en una dirección y 56 kb/s en la dirección opuesta, sin embargo, para una conexión asíncrona es posible soportar 432,6 kb/s en ambas direcciones si el enlace es simétrico.

Arquitectura hardware

El hardware que compone el dispositivo Bluetooth está compuesto por dos partes:

- **un dispositivo de radio**, encargado de modular y transmitir la señal
- **un controlador digital**, compuesto por una CPU, por un procesador de señales digitales (DSP - Digital Signal Processor) llamado Link Controller (o controlador de Enlace) y de los interfaces con el dispositivo anfitrión.

El LC o Link Controller está encargado de hacer el procesamiento de la banda base y del manejo de los protocolos ARQ y FEC de capa física. Además, se encarga de las funciones de transferencia (tanto asíncrona como síncrona), codificación de Audio y cifrado de datos.

El CPU del dispositivo se encarga de atender las instrucciones relacionadas con Bluetooth del dispositivo anfitrión, para así simplificar su operación. Para ello, sobre el CPU corre un software denominado Link Manager que tiene la función de comunicarse con otros dispositivos por medio del protocolo LMP.

Entre las tareas realizadas por el LC y el Link Manager, destacan las siguientes:

- Envío y Recepción de Datos.
- Empaginamiento y Peticiones.
- Determinación de Conexiones.
- Autenticación.
- Negociación y determinación de tipos de enlace.
- Determinación del tipo de cuerpo de cada paquete.
- Ubicación del dispositivo en modo sniff o hold.

CAPITULO 3

3. Diseño e Implementación del Proyecto

Componentes del Sistema

Entre los componentes generales de este sistema se encuentran los siguientes:

- **Kit Lego Mindstorm:** Con el kit armaremos el robot modelo para realizar la demostración de aplicación de nuestro proyecto.
- **Sensor de Luz:** Necesario para que el robot pueda hacer el recorrido del área pre-establecida.
- **Sensor de Tacto:** Ayuda en este proyecto al robot modelo a evadir los obstáculos que pudiera encontrar en el camino.
- **Sensor de sonido:** Utilizado para ir midiendo los valores de ruido en el medio.
- **Área de Tránsito:** El área pre-establecida para que el robot modelo pueda recorrer toda el área de trabajo de forma ordenada.
- **Conexión bluetooth:** Indispensable para el envío de datos medidos y mensajes de alarma.

Detalle del Sistema

Visto en forma sencillo este proyecto esta enfocado a ser utilizado como un sistema de alarma, para prevenir fallas en los equipos que intervienen en un proceso.

El funcionamiento del sistema se resume de la siguiente manera:

Se cuenta con un Robot, que posee movilidad gracias al uso de servomotores, el mismo que recorre el área donde se encuentran los equipos que deseamos monitorear, este recorrido esta previamente establecido a través de áreas de un color distinto por donde el robot se moverá a menos que encuentre un obstáculo.

A medida que el robot recorre la sección de trabajo ira realizando mediciones del nivel de ruido en el ambiente y enviara el dato medido de tal forma que al final podremos visualizarlo en un archivo de Excel para llevar un registro. Además tendrá dentro de su programación niveles umbral definidos tal que al rebasar estos umbrales genere un mensaje de alarmas además del dato de lectura y se detiene en la posición donde se halla el nivel de ruido que genera la alarma.

La recolección de datos permitirá verificar los cambios en el nivel de ruido a través del tiempo, programar un mantenimiento preventivo en caso de ser necesario.

Para la implementación de este proyecto se debe tener niveles de ruido bien definidos dentro del área que va a ser monitoreada, y estar seguros del buen funcionamiento de los sensores que vamos a usar, por tal motivo se incluye en el diseño de este proyecto una prueba de los sensores antes del inicio del monitoreo.

En el caso particular de este proyecto se utilizará un modelo de robot seguidor de área, el área de tránsito será de color amarillo de tal forma que con la ayuda del sensor de luz y tacto pueda recorrer este camino sin desviarse de forma innecesaria. Dentro del programa se considera que haya una rutina para que al momento de encontrar un obstáculo intente una acción evasiva que le permitirá regresar al camino pre-establecido y volver al programa principal, mientras continua el monitoreo y envío de datos.

Diseño del Proyecto

Como primer requerimiento se tiene el diseño y armado de un Robot modelo que cuente con movilidad además de los sensores considerados en el proyecto, así pues se presenta la siguiente propuesta que se utilizará en la presentación y demostración.

Robot Modelo

Figura 3-1: Robot Modelo

Construcción del Robot

En esta parte veremos una guía grafica de la construcción del Robot modelo para este proyecto, que como se había indicado tiene como base un diseño móvil. A continuación la construcción paso a paso:

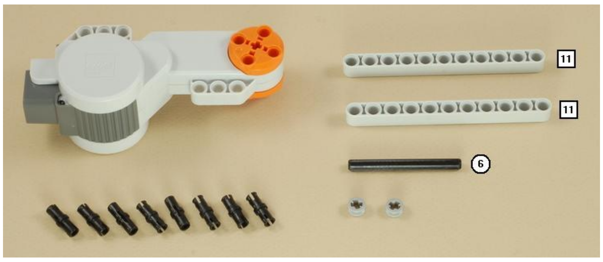


Figura 3-2: Robot Modelo Paso 1.1

Figura 3-3: Robot Modelo Paso 1.2

Figura 3-4: Robot Modelo Paso 1.3

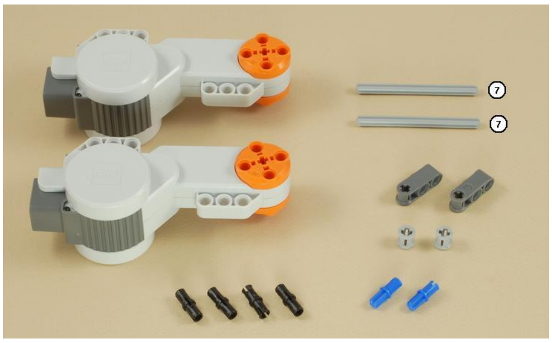


Figura 3-5: Robot Modelo Paso 2.1

Figura 3-6: Robot Modelo Paso 2.2

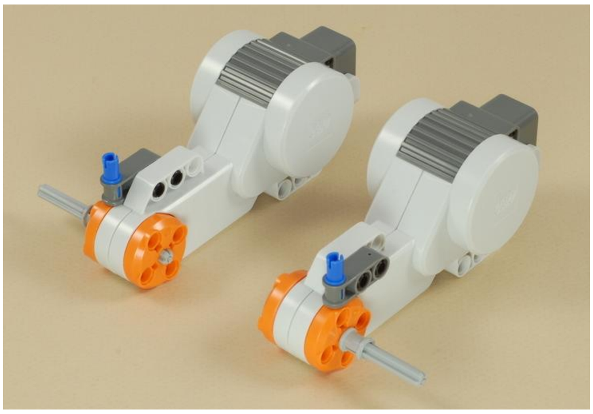


Figura 3-7: Robot Modelo Paso 2.3

Figura 3-8: Robot Modelo Paso 2.4

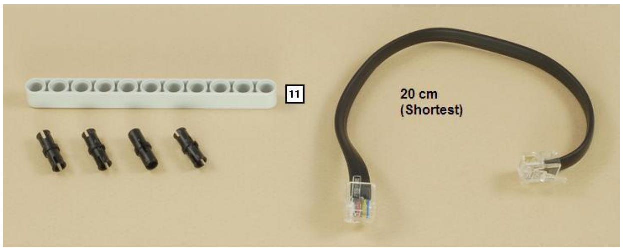


Figura 3-9: Robot Modelo Paso 3.1

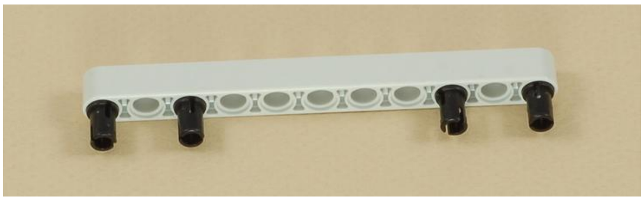


Figura 3-10: Robot Modelo Paso 3.2

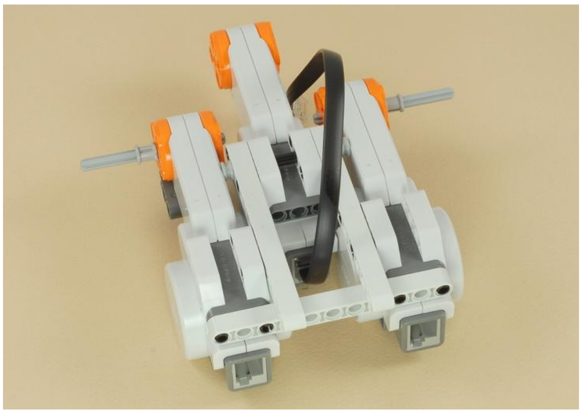


Figura 3-11: Robot Modelo Paso 3.3

Figura 3-12: Robot Modelo Paso 4.1

Figura 3-13: Robot Modelo Paso 4.2

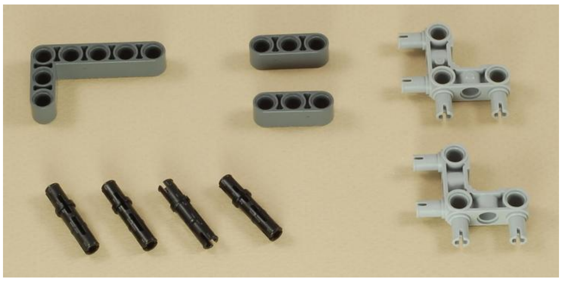


Figura 3-14: Robot Modelo Paso 5.1

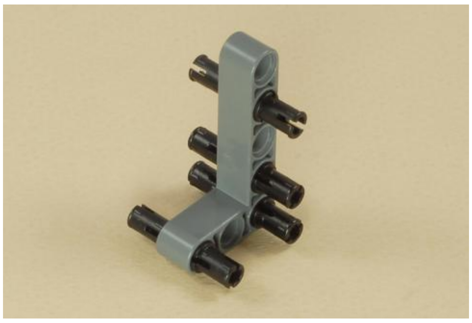


Figura 3-15: Robot Modelo Paso 5.2

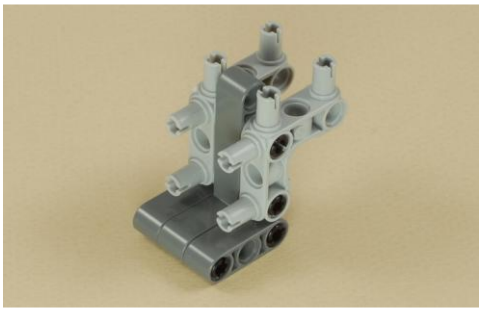


Figura 3-16: Robot Modelo Paso 5.3

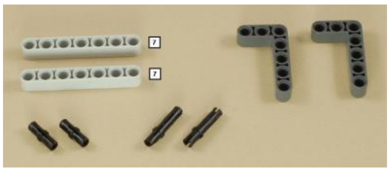


Figura 3-17: Robot Modelo Paso 6.1

Figura 3-18: Robot Modelo Paso 6.2

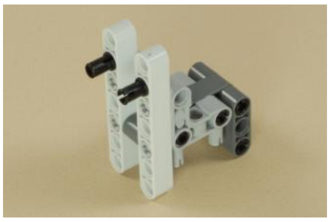


Figura 3-19: Robot Modelo Paso 6.3

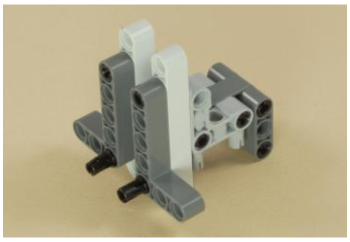


Figura 3-20: Robot Modelo Paso 6.4

Figura 3-21: Robot Modelo Paso 7.1

Figura 3-22: Robot Modelo Paso 7.2

Figura 3-23: Robot Modelo Paso 8.1

Figura 3-24: Robot Modelo Paso 8.2

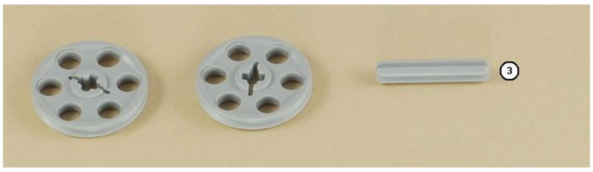


Figura 3-25: Robot Modelo Paso 9.1

Figura 3-26: Robot Modelo Paso 9.2

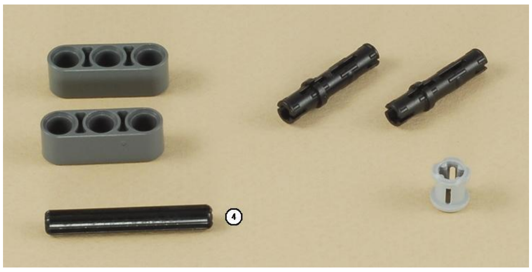


Figura 3-27: Robot Modelo Paso 10.1

Figura 3-28: Robot Modelo Paso 10.2

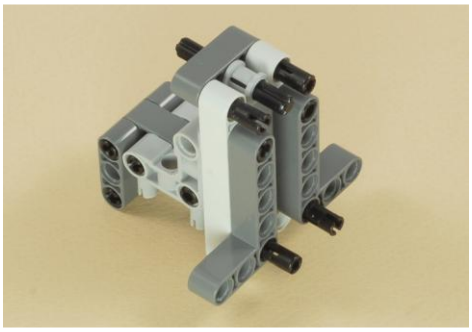


Figura 3-29: Robot Modelo Paso 10.3

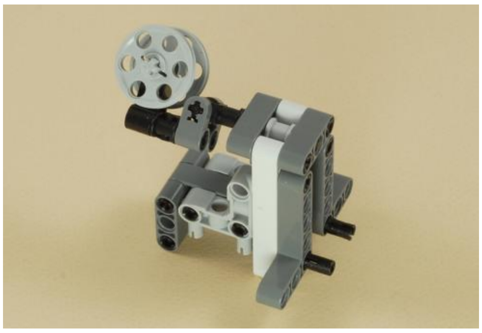


Figura 3-30: Robot Modelo Paso 10.4

Figura 3-31: Robot Modelo Paso 10.5

Figura 3-32: Robot Modelo Paso 11.1

Figura 3-33: Robot Modelo Paso 11.2

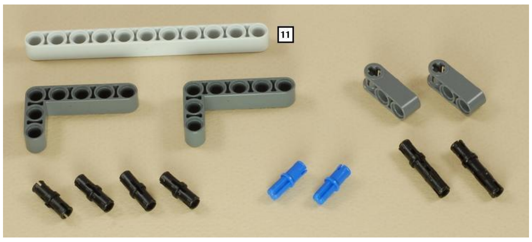


Figura 3-34: Robot Modelo Paso 12.1

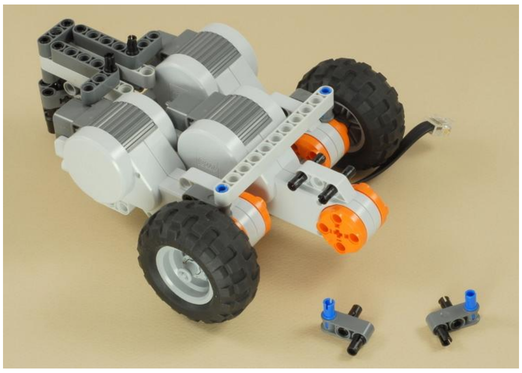


Figura 3-35: Robot Modelo Paso 12.2

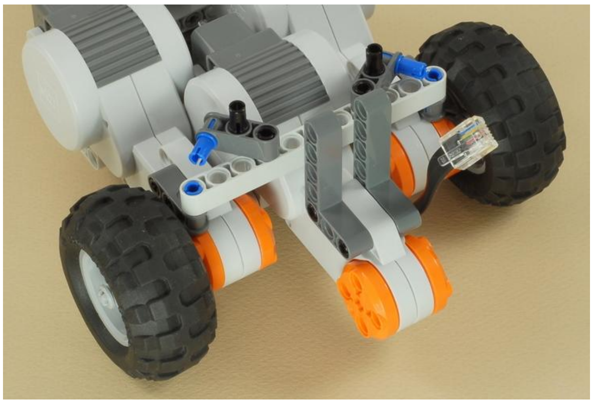


Figura 3-36: Robot Modelo Paso 12.3

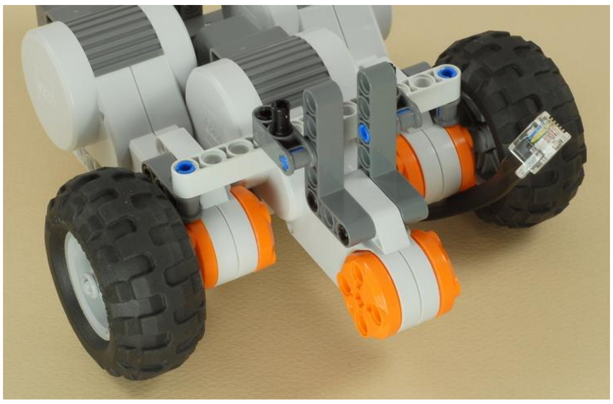


Figura 3-37: Robot Modelo Paso 12.4

Figura 3-38: Robot Modelo Paso 13.1

Figura 3-39: Robot Modelo Paso 13.2

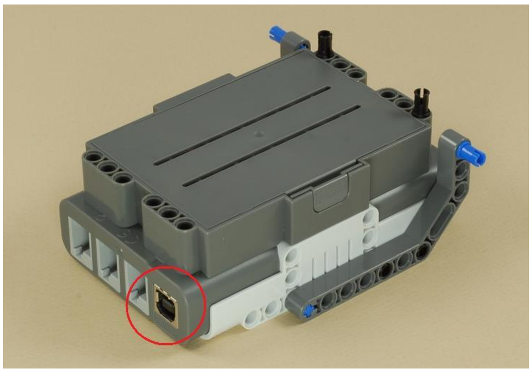


Figura 3-40: Robot Modelo Paso 13.3

Figura 3-41: Robot Modelo Paso 13.4

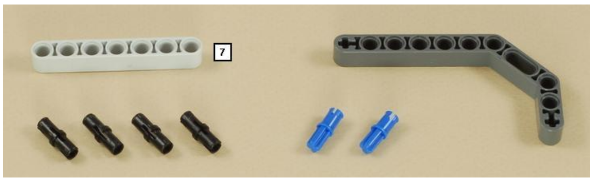


Figura 3-42: Robot Modelo Paso 14.1

Figura 3-43: Robot Modelo Paso 14.2

Figura 3-44: Robot Modelo Paso 14.3

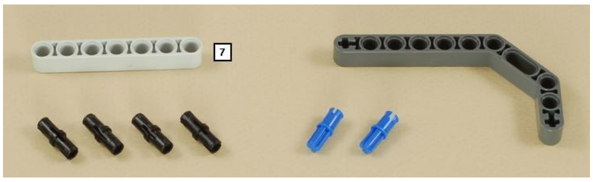


Figura 3-45: Robot Modelo Paso 15.1

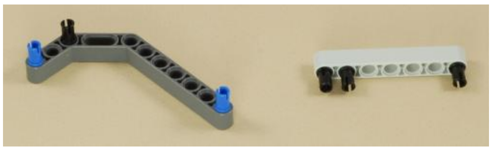


Figura 3-46: Robot Modelo Paso 15.2

Figura 3-47: Robot Modelo Paso 15.3

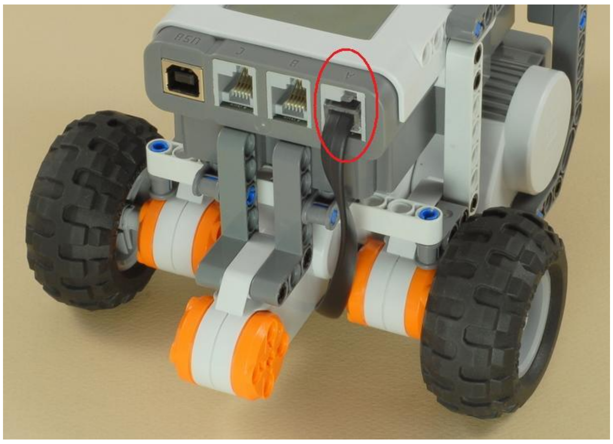


Figura 3-48: Robot Modelo Paso 15.4

Figura 3-49: Robot Modelo Paso 16

Figura 3-50: Robot Modelo Paso 17.1

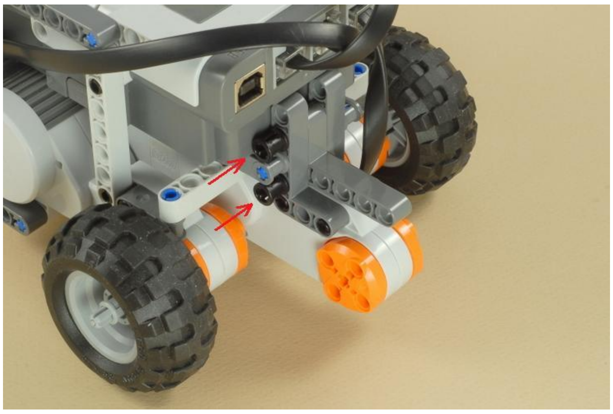


Figura 3-51: Robot Modelo Paso 17.2

Figura 3-52: Robot Modelo Paso 18.1

Figura 3-53: Robot Modelo Paso 18.2

Figura 3-54: Robot Modelo Terminado

Área de recorrido

El área de recorrido que seguirá el robot es de color amarillo delimitada por una línea de color negro para evitar que se salga del recorrido establecido, en las áreas esquineras donde debe realizar giros avanza hasta que encuentra el limite de color negro una vez ahí retrocede, mientras lo hace sensa el color si es el área amarilla gira hacia la derecha, si en su defecto es rojo el giro se realizara hacia la izquierda una vez q da el giro sensa el color del área si esta de nuevo en el área de Tránsito establecida continua de frente caso contrario vuelve a realizar otro giro y al encontrar el área de recorrido continua.

CAPITULO 4

4. SIMULACION Y PRUEBAS EXPERIMENTALES

4.1. Descripción del Procedimiento de Prueba

Figura 4-1: Robot

El ciclo de prueba inicia cuado el robot comienza su movimiento para lo cual utiliza un programado como un seguidor de área, que funciona de la siguiente manera:

Comienza el movimiento a través del área de color amarillo en los bordes esta delimitada por una línea negra para que reconozca el cambio de color a través del sensor de luz y no rebase esta área así se mantiene dentro del área amarilla, si al avanzar se encuentra con esta línea limite el robot da marcha de retro y sensa si el color es amarillo y hace un giro hacia la derecha y en el caso de ser azul gira hacia la izquierda, una vez que ha realizado el giro vuelve a sensar hasta confirmar que se encuentra dentro del área de transito y continua su recorrido, de esta manera además funciona cuando encuentra curvas en su recorrido.

El robot durante todo el tiempo se mantiene sensando los niveles de sonido, de los cuales se obtienen valores medidos entre 0 y 1023, los valores de umbral que se maneja para nuestro caso de prueba es un mínimo de 400, tal que si el valor medido es menor envía una alarma de “alarma por sonido bajo” y un máximo de 700, de tal forma que si rebasa este valor envía la alarma “alarma por sonido alto”. Cabe recalcar que cuando el robot detecta un valor de alarma se detiene en la posición en que lo detecta y envía los datos a un archivo EXCEL, pero hay que notar que durante todo el recorrido va graficando los valores del nivel de sonido a través de la herramienta de Plot de MATLAB.

Para el caso de encontrar un obstáculo en su recorrido este será percibido gracias a la ayuda del sensor de tacto, en ese momento el robot inicia una acción evasiva, esto es, retrocede y luego gira hacia la derecha, avanza aproximadamente un segundo y luego gira hacia la izquierda y sigue de frente, si ocurre que al intentar seguir su recorrido el sensor de tacto vuelve a activarse, indicando que la maniobra evasiva no fue efectiva, el robot se detiene y se envía una alarma “Touch” y permanece en la posición donde se genero la alarma.

Se tiene una marca cuando el robot ha terminado el ciclo de recorrido y así hace un alto para realizar el envío de datos cada ciclo de recorrido completado.

4.2. Datos Experimentales

Como se indicó en la parte de pruebas, el robot realiza un envío de datos a un archivo EXCEL cuando se detiene por una alarma o porque ha terminado el ciclo de recorrido. Los datos que se envía tendrán un formato similar al que se presentan a continuación,

el mismo que es parte de un archivo obtenido durante las pruebas de desarrollo de este proyecto y en el cual se muestra además en rojo la alarma mostrada.

a l a r m a p o r s o n i d o b a j o

0
40
78
37
51
78
43
70
89
39
69
61
38
87
31

4.3. Pruebas con Simulink

En la etapa de pruebas previa a la programación del robot, tratamos de trabajar con los ejemplos dados en la carpeta **samples** de **ecrobotnxt** instalado en Matlab para poder programar el robot con Simulink. Al compilar el programa y colocarlo vía USB al robot, este no presento ningún problema, pero al tratar de correr el programa TestBluetoothRx.mdl; el cual se trata de mover un servomotor enviando las revoluciones desde Matlab por medio del bluetooth; la comunicación no se efectuaba.

Intentamos varias configuraciones de la comunicación vía bluetooth, bilateral entre el bluetooth de la PC y el del robot nxt, unilateral colocando como emisor a la PC y receptor al Nxt y unilateral colocando como emisor al Nxt y como receptor a la PC.

Al colocar al bluetooth de la PC como emisor funciono la primera vez, pero al tratar de hacerlo nuevamente el programa no respondía. Lo intentamos varias veces pero al ver que no obteníamos una comunicación confiable decidimos buscar una manera alterna de comunicación vía bluetooth entre la PC y el Nxt.

Luego de buscar y aplicar varias pruebas decidimos trabajar el programa en Matlab obteniendo como resultado un retardo en la comunicación vía bluetooth, pero a diferencia de lo que ocurrió con Simulink el proceso es reproducible las veces que desee.

Esto lo logramos bajo la configuración de la comunicación vía bluetooth a través del comando COM_**MakeBTConfigFile** de Matlab, obteniendo una comunicación bilateral entre la PC (Matlab) y el Nxt.

Tomamos en cuenta el uso de Simulink en la etapa de graficación de nuestro sistema, pero debido a que Simulink funciona como un proceso de Matlab solo teníamos la opción de graficar una vez terminado todo el proceso, mientras que Matlab nos presenta la herramienta **PLOT** que nos permite graficar punto a punto y con la ayuda del comando **"hold on"** mantenemos los puntos anteriores simulando una gráfica continua en tiempo real.

CONCLUSIONES Y RECOMENDACIONES

1. En el presente proyecto de graduación hemos desarrollado un programa de control de robots para la detección de anomalías en el buen funcionamiento de maquinarias utilizadas en áreas industriales con la ayuda del software MATLAB, donde el robot realiza un monitoreo del nivel de ruido y compara con los niveles máximo y mínimo que tiene y llega a generar un archivo de los valores medidos para poder tener un histórico del comportamiento de este parámetro, así como enviar alarmas que den una alerta ante las posibles fallas.

Conclusiones

2. El uso de la herramienta MATLAB permite generar programas interesantes para el control de robots y sensores en aplicaciones de mayor escala que requieren obtención y envío de datos como es el caso de nuestro proyecto, donde estamos utilizando tres sensores: luz, tacto y sonido., los mismos que le permitirán al robot seguir el área definida para transitar, detectar y evadir objetos y realizar la medición requerida.

3. La transmisión vía bluetooth con SIMULINK no es aplicable cuando deseamos desarrollar un proceso repetitivo, en que el robot tendrá que recopilar y enviar datos, esto se concluye luego de que en las pruebas realizadas al colocar al bluetooth de la PC como emisor funciono la primera vez, pero al tratar de hacerlo nuevamente el programa no respondía

4. Al trabajar con programas desarrollados en MATLAB la transmisión vía bluetooth para procesos repetitivos es posible pero con un retardo que podemos considerar aceptable, esto lo logramos bajo la configuración de la comunicación vía bluetooth a través del comando COM_MakeBTConfigFile de Matlab, obteniendo una comunicación bilateral entre la PC (Matlab) y el Nxt.

5. Encontramos que MATLAB cuenta con herramientas como PLOT que nos permiten graficar los datos que obtienen los sensores punto a punto y con la ayuda del comando “hold on” mantenemos los puntos anteriores simulando una gráfica continua en tiempo real, permitiendo así un operador ir viendo si los niveles se mantienen dentro de los limites previstos sin necesidad de esperar el envío del archivo EXCEL por parte del robot.

Recomendaciones

Es muy importante considerar cuando hacemos el diseño del robot, la aplicación que tendrá, los sensores que va a usar y en que medio va a trabajar, porque:

1. En el caso de usar el sensor de sonido, este debe estar lo mas alejado posible de los servomotores para evitar que el ruido generado por ellos interfiera.

2. Para el caso de usar sensor de luz, este debe estar a una distancia apropiada de la superficie que va a estar sensando y además considerar que la luz que haya en el ambiente puede afectar la medida que se toma.

3. Con el sensor de tacto lo que se debe considerar es que este ubicado en el robot de tal forma que le permita sensar las superficies con suficiente distancia como para evitar daños y también realizar las acciones que requiera según lo que este sensando.

Respecto a los comandos utilizados en programas desarrollados en MATLAB hay que considerar:

4. La existencia de conflicto entre ciertos comandos cuando se los utiliza juntos en una rutina tal como es el caso del comando SetTurnRatio y el comando SyncToMotor.

5. Para los giros se recomienda usar el comando SpeedRegulation que rompe la sincronización de los motores antes de ejecutar la orden de giro, y además si se lo combina con el comando

NXT_ResetMotorPosition nos ayuda a resetear los motores y se asegura que no quede ninguna orden guardada al momento de reiniciar el proceso.

ANEXO 1

Preproyecto

Garra Paralizante

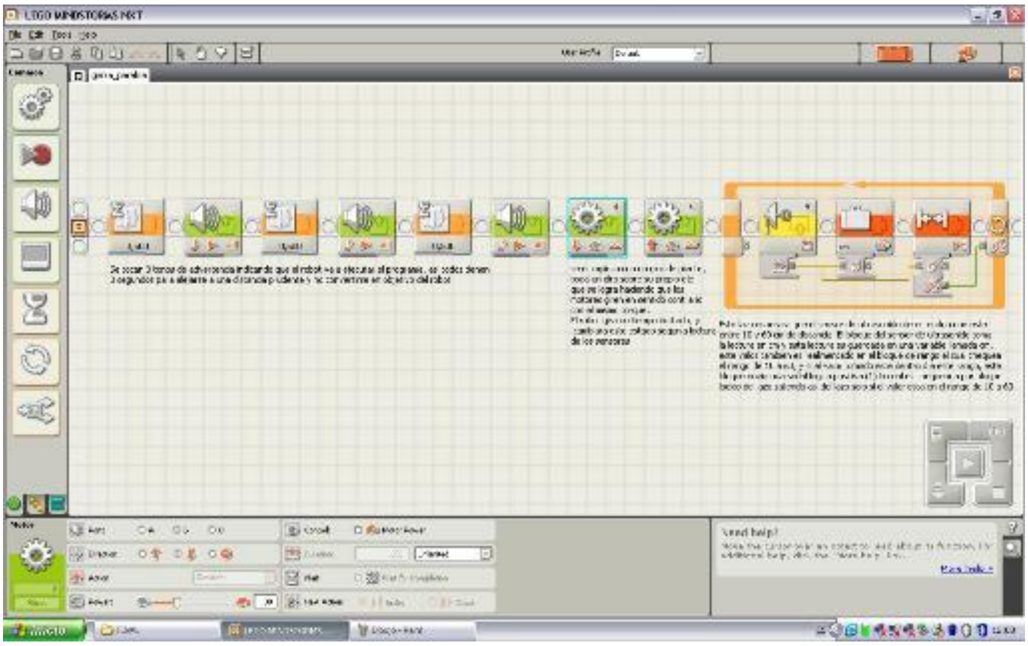
En nuestro robot usamos un sensor de ultrasonido y 3 servomotores además de las partes que vienen en el kit de lego Mindstorms para dar forma al móvil y la garra.

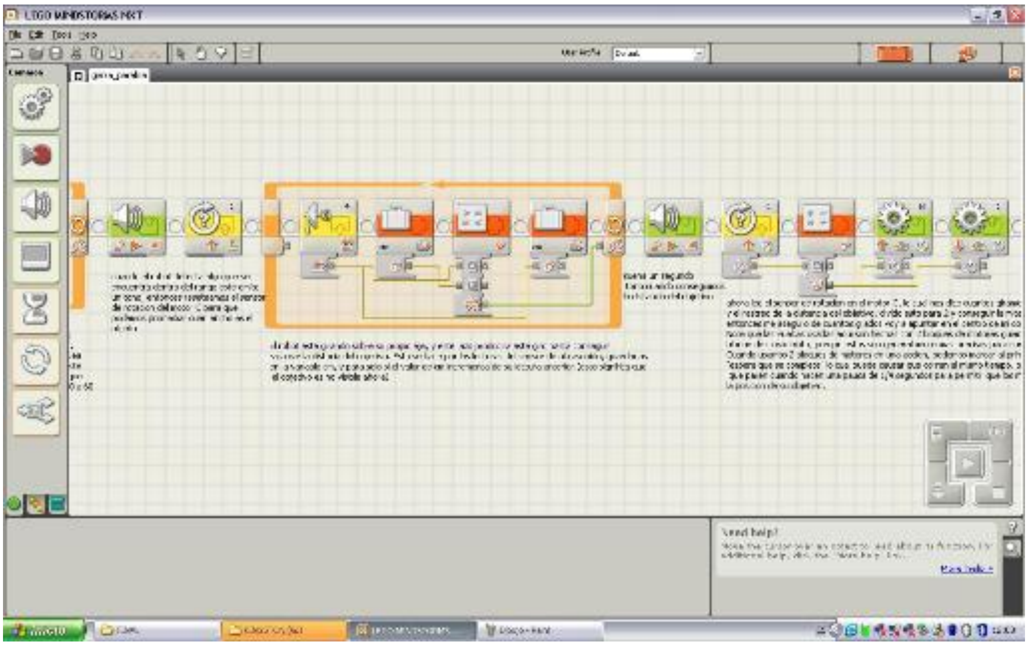
El robot empieza emitiendo 3 tonos uno cada segundo simbolizando una señal de advertencia para que quienes se encuentren cerca del robot se alejen ya que empezara a sensar objetivos cercanos a través del sensor de ultrasonido.

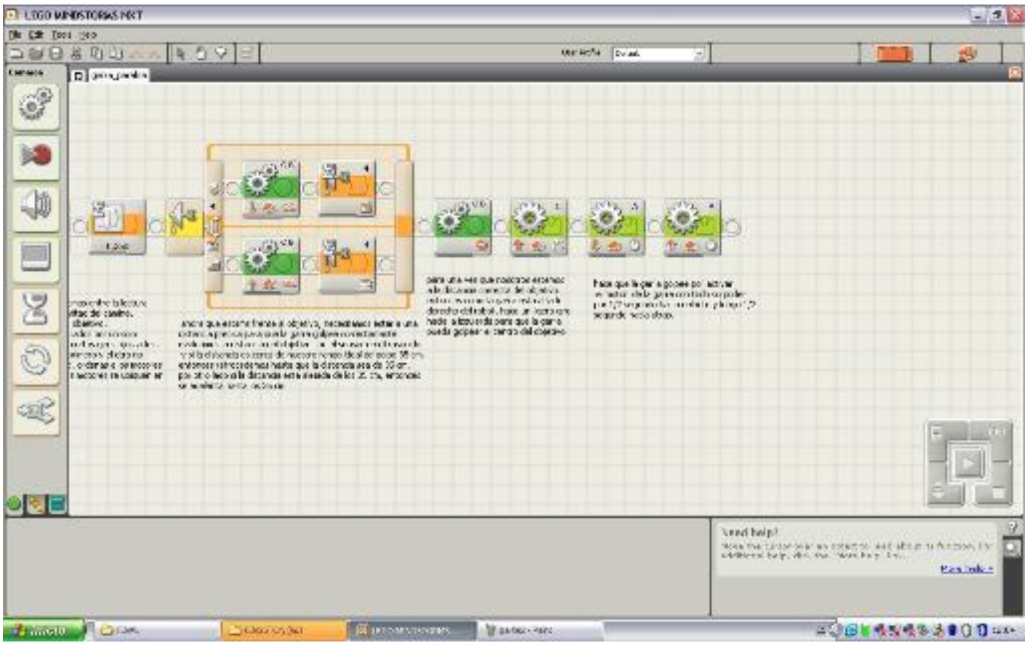
El funcionamiento empieza con un giro ilimitado en el propio eje; lo que conseguimos haciendo que los motores giren en sentido contrario y a tiempo sincronizado; y saldrá de este estado cuando nuestro robot sensa cualquier objetivo que se encuentre en un rango entre 10 y 60 cm. Al encontrar dicho objetivo, el robot emite un tono y da un pequeño giro para sensar y calcular a que distancia estimada se encuentra nuestro objetivo.

Al obtener este calculo se emite un segundo tono que indica que tenemos la distancia y leemos el movimiento hecho por los motores en el giro antes mencionado para calcular el centro del objetivo y poder dar un golpe preciso y paralizante idealmente en el centro del objetivo sensado.

Ahora que estamos frente al objetivo necesitamos estar a una distancia precisa para que mi garra pueda golpear correctamente y la distancia de mi garra es 35 cm así que según la distancia antes sensada me muevo hacia delante o hacia atrás, se da un pequeño giro hacia la izquierda para dar un impulso a la garra y se mueve el servomotor que sostiene la garra ½ segundo hacia delante y luego ½ segundo hacia atrás obteniendo así un golpe acertado y paralizante en el centro del objetivo.







ANEXO 2

Programa de Control Proyecto Final

```
%% cerrar todo lo que pudo haber esta prendido
COM_CloseNXT all
clear all
close all
%% seteos de variables

toque=0
contador=0
obst=0
cont=0

%% Set up ports (seteos de puertos)
portUS       = SENSOR_1;     % SENSOR DE TACTO
portUS2      = SENSOR_2;     % SENSOR DE SONIDO
portUS3      = SENSOR_3;     % SENSOR DE LUZ
portMotor    = MOTOR_A;      % MOTOR IZQUIERDA
portMotor2   = MOTOR_B;      % MOTOR DERECHA
MotorSpeed  = 100;


% %% Get USB handle
% COM_CloseNXT all
% h = COM_OpenNXT();
```

```
% COM_SetDefaultNXT(h);

%% Open Bluetooth connetion

handle = COM_OpenNXT('bluetooth.ini', 'check');
COM_SetDefaultNXT(handle);

%% Open sensors connetion
OpenSwitch(portUS);
OpenLight(portUS3,'ACTIVE');
OpenSound(portUS2, 'DB');

%% MAIN
SetMotor(portMotor);
          SetPower(75);
        SyncToMotor(portMotor2);
        SendMotorSettings();

while true

        NXT_ResetMotorPosition(MOTOR_A, true);
        NXT_ResetMotorPosition(MOTOR_B, true);

        light = GetLight(portUS3)
        toque = GetSwitch(portUS)
        cont=cont+1
        value(cont) = GetSound(portUS2)
        plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
        hold on


        while true

            if (value(cont)<75 || value(cont)>700 ) %valores minimo y
maximo normal

               break

            end

            if (toque == 1)
                     break
                end;
            cont=cont+1
            value(cont) = GetSound(portUS2)
            plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
            hold on
            light = GetLight(portUS3);
            toque = GetSwitch(portUS);

                  if (value(cont)<75 || value(cont)>700)

                     break

                  end

                 if (toque == 1)
```

```
                         break
                    end;

                 if (light<300)% si encontro marca de fin del camino
el robot se detiene y
                              % envia los reportes de los datos
sensados

                      break

                 end

                 if (light<500 ) %si topo linea negra me
detengo,retrocedo
                               %y vuelvo a sensar para decidir giro
                   cont=cont+1
                   value(cont) = GetSound(portUS2)
                   plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
                   hold on
                   if (value(cont)<75 || value(cont)>700)

                     break

                   end
                   StopMotor(portMotor, 'brake');
                   StopMotor(portMotor2, 'brake');
                   pause (1);
                   SetMotor(portMotor);
                    SyncToMotor(portMotor2);
                    SetPower(-75);
                        SendMotorSettings(); %orden de retroceder los
motores

                        pause (1.5)

                 else
                     break
                 end
                    cont=cont+1
                    value(cont) = GetSound(portUS2)
                    plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
                    hold on
                   if (value(cont)<75 || value(cont)>700)

                     break

                  end
                    StopMotor(portMotor, 'brake');
                    StopMotor(MOTOR_B, 'brake');
                     pause (1)

                      light = GetLight(portUS3);%senso2 para decidir
giro

                 if (light>640 ) %si detecta zona amarilla, gira
derecha y continua
```

```
    SetMotor(portMotor);
        SetPower(75);
        SpeedRegulation('on');
      SendMotorSettings(); %orden de giro a la derecha
                   pause(0.7)
                while true
                    cont=cont+1
                   value(cont) = GetSound(portUS2)
                   plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
                   hold on
                  if (value(cont)<75 || value(cont)>700)

                   break

                  end

                   StopMotor(portMotor, 'brake');
                  StopMotor(portMotor2, 'brake');
                  pause (1)
                  light = GetLight(portUS3) %senso para confirmar
giro correcto
                                           %si luego de girar
detecta
                                           %zona amarilla continua
                  if (light>640 )
                   break
                  else % si no detecta zona amarilla gira un poco
mas a la derecha


                  SetMotor(portMotor2);
                      SetPower(75);
                      SpeedRegulation('on');
                    SendMotorSettings();
                   cont=cont+1
                    value(cont) = GetSound(portUS2)
                    plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
                    hold on

                      if (value(cont)<75 || value(cont)>700)

                          break

                      end

                   pause(0.5)
                  end
                end
                break

               else %si detecta otro color menor que amarillo en la
escala
                   %gira a la izquierda

                  SetMotor(portMotor2);
                      SetPower(75);
                      SpeedRegulation('on');
```

```
                      SendMotorSettings(); %orden de giro a la
izquierda

                    cont=cont+1
                    value(cont) = GetSound(portUS2)
                    plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
                    hold on
                     if (value(cont)<75 || value(cont)>700)

                         break

                     end

                     pause(0.7)
                  while true
                     StopMotor(portMotor, 'brake');
                    StopMotor(portMotor2, 'brake');
                    pause(1)
                    light = GetLight(portUS3)

                    if (light<650 )% se percata que aun este en la
marca de giro
                     break
                    else %si no estoy en la marca de giro giro un poco
mas

                    SetMotor(portMotor2);
                        SetPower(75);
                        SpeedRegulation('on');
                      SendMotorSettings();
                      cont=cont+1
                    value(cont) = GetSound(portUS2)
                    plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
                    hold on
                    if (value(cont)<75 || value(cont)>700)

                     break

                    end

                     pause(0.75)
                    end
                  end

                  break

                 end

        end

       cont=cont+1
       value(cont) = GetSound(portUS2)
       plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
       hold on
       if (value(cont)<75 || value(cont)>700)
```

```
            break
        end

        contador = contador+1

while (toque == 1) % si se encuentra con algun obstaculo lo rodea por
la derecha
                 % y se vuelve a encontrarlo en el mismo ciclo se
detiene y
                 % envia una alarma de obstaculo.

            if(obst==2)
     %si el robot se encontro con un obstaculo en el camino hace la
     %maniobra para rodearlo pero si lo vuelve a encontrar en el mismo
     %ciclo se detiene y envia un mensaje de alarma por obstaculo

              break
             end

            cont=cont+1
            value(cont) = GetSound(portUS2)
            plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
            hold on
            if (value(cont)<110 || value(cont)>700)

               break
            end

            StopMotor(portMotor, 'brake');
            StopMotor(portMotor2, 'brake');

            pause(1)

            SetMotor(portMotor);
            SyncToMotor(portMotor2);
            SetPower(-75);
              SendMotorSettings();
              cont=cont+1
              value(cont) = GetSound(portUS2)
              plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
              hold on
         if (value(cont)<75 || value(cont)>700)

                  break
          end

              pause(1)

              StopMotor(portMotor, 'brake');
            StopMotor(portMotor2, 'brake');

            pause (1);
            NXT_ResetMotorPosition(MOTOR_A, true);
      NXT_ResetMotorPosition(MOTOR_B, true);

              SetMotor(portMotor);
```

```
        SetPower(75);
        SpeedRegulation('on');
      SendMotorSettings();
      pause(0.65);

     cont=cont+1
     value(cont) = GetSound(portUS2) % senso sonido aun cuando me
encuenro con un obstaculo
      plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
      hold on
      if (value(cont)<75 || value(cont)>700)

            break
      end


      StopMotor(portMotor, 'brake');
            StopMotor(portMotor2, 'brake');

        pause (1)

        NXT_ResetMotorPosition(MOTOR_A, true);
      NXT_ResetMotorPosition(MOTOR_B, true);

       SetMotor(portMotor);
       SyncToMotor(portMotor2);
       SetPower(75);
       SendMotorSettings();
       pause (0.5)

       cont=cont+1
       value(cont) = GetSound(portUS2)
       plot (cont,value(cont),'--
rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','g','MarkerS
ize',5)
      hold on
      if (value(cont)<75 || value(cont)>700)

            break
       end


       StopMotor(portMotor, 'brake');
       StopMotor(portMotor2, 'brake');

       pause (1)

       NXT_ResetMotorPosition(MOTOR_A, true);
       NXT_ResetMotorPosition(MOTOR_B, true);

        SetMotor(portMotor2);
        SetPower(75);
        SpeedRegulation('on');
      SendMotorSettings();
      pause(0.80)

      StopMotor(portMotor, 'brake');
       StopMotor(portMotor2, 'brake');
       pause(1)
```

```
      toque = 0

      NXT_ResetMotorPosition(MOTOR_A, true);
       NXT_ResetMotorPosition(MOTOR_B, true);
       toque = GetSwitch(portUS)
       SetMotor(portMotor);
            SyncToMotor(portMotor2);
            SetPower(75);
              SendMotorSettings();
              pause(1)

        toque = GetSwitch(portUS)


      StopMotor(portMotor, 'brake');
       StopMotor(portMotor2, 'brake');
       pause(0.5)
       NXT_ResetMotorPosition(MOTOR_A, true);
       NXT_ResetMotorPosition(MOTOR_B, true);

       light = GetLight(portUS3);

       if (toque==1||light<500)

           obst=2

           break
       end

end

if(obst==2)
     break
  end

if (value(cont)<75 || value(cont)>700)
    %si el sonido baja de este nivel hay problemas el robot se detiene
y
    %envia el reporte de los datos sensado en un archivo excel ademas
de
    %la grafica de los datos del hasta el momento.


    break


end
if (light<300)% si encontro marca de fin del camino el robot se
detiene y
              % envia los reportes de los datos sensados

    break

end

        SetMotor(portMotor);
          SetPower(75);
        SyncToMotor(portMotor2);
        SendMotorSettings();
end
```

```
StopMotor(portMotor, 'brake');
 StopMotor(portMotor2, 'brake');
 NXT_ResetMotorPosition(MOTOR_A, true);
 NXT_ResetMotorPosition(MOTOR_B, true);
%%  ENVIO DE REPORTES Y ALARMAS
 if(obst==2)
 %se envia mensaje de error por obstaculo
 alarma='alarma touch          '
 value = value'
   xlswrite('prueba final.xls', alarma, 'Valores Sensados', 'A1')
    xlswrite('prueba final.xls', value, 'Valores Sensados', 'B4')
obst=0
end

if (value(cont)<75)
    %si el sonido baja de este nivel hay problemas el robot se detiene
y
    %envia el reporte de los datos sensado en un archivo excel ademas
de
    %la grafica de los datos del hasta el momento.
    alarma='alarma sonido bajo '
    value = value'
    xlswrite('prueba final.xls', alarma, 'Valores Sensados', 'A1')
    xlswrite('prueba final.xls', value, 'Valores Sensados', 'B4')


end
if (value(cont)>700)
    %si el sonido sube de este nivel hay problemas el robot se detiene
y
    %envia el reporte de los datos sensado en un archivo excel ademas
de
    %la grafica de los datos del hasta el momento.
    alarma='alarma sonido alto '
    value = value'
    xlswrite('prueba final.xls', alarma, 'Valores Sensados', 'A1')
    xlswrite('prueba final.xls', value, 'Valores Sensados', 'B4')

end
if (light<300)% si encontro marca de fin del camino el robot se
detiene y
              % envia los reportes de los datos sensados

    alarma='Fin del senso      '
    value = value'
    xlswrite('prueba final.xls', alarma, 'Valores Sensados', 'A1')
    xlswrite('prueba final.xls', value, 'Valores Sensados', 'B4')

 end
 %% Clean up
CloseSensor(portUS3);
CloseSensor(portUS);
CloseSensor(portUS2);

%% CIERRE DE COMUNICACION
COM_CloseNXT(handle);
```

BIBLIOGRAFIA

Libros

1. Floyd Kelly James, LEGO MINDSTORNMS NXTG programing guide, editorial Technology in Action Press, Julio 2007.

2. Gasperi Michael and Hurbain Philippe, EXTREME NXT Extending the LEGO MINDSTORMS NXT to the Next Level, editorial Technology in Action Press, segunda edición Octubre 2009.

Sitios web

1. Medspain.com, Junio del 2009, página Html
http://www.medspain.com/colaboraciones/ruidoindustrial.htm

2. Loginstore.com, Julio 14 del 2009, página Html
http://www.loginstore.com/blog/2009/07/14/lego-mindstorms-nxt/

3. Rbtnxt.com, Enero del 2009, página Html
http://rbtnxt.blogspot.com/2009/01/el-sensor-de-sonido.html

4. Lrobotikas.net, página Html
http://lrobotikas.net/mediawiki/index.php?title=Programaci%C3%B3n_NXT

5. Compelect.com, Enero del 2005, página Html
http://www.compelect.com.co/otros/diamatlab/2005/Roboticamovil.pdf

6. Petercorke.com, Agosto 17 del 2009, página Html
7.
http://petercorke.com/Robotics_Toolbox.html

8. Aruvisa.org, Agosto 22 del 2009, página Html
http://www.aruvisa.org/node/9

9. Mindstorms.com, Agosto 22 del 2009, página Html
http://mindstorms.lego.com/eng/overview/Touch_Sensor.aspx

10. Angelfire.com, Junio del 2009, página Html

http://www.angelfire.com/la/hmolina/matlab7.html

11. Winkipedia.org, Agosto 24 del 2009, página Html
http://es.wikipedia.org/wiki/Tacto

12. Winkipedia.org, Agosto 24 del 2009, página Html
http://es.wikipedia.org/wiki/Bluetooth