ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

"Uso de Matlab y Simulink para el control de robots y la observación de sensores de luz y ultrasónico"

INFORME DE MATERIA DE GRADUACIÓN

Previa la obtención del Título de:

INGENIERO EN ELECTRICIDAD

ESPECIALIDAD EN ELECTRÓNICA Y AUTOMATIZACIÓN INDUSTRIAL

Presentado por:

Mariano de Jesús Malavé Lindao

Manuel Rogelio Nevárez Toledo

Pedro Fabián Vallejo Mero

GUAYAQUIL – ECUADOR

AÑO 2009

AGRADECIMIENTO

A Dios.

A los amigos.

A todas las personas que contribuyeron en el desarrollo de este trabajo.

A todos quienes apuestan por el desarrollo tecnológico en Ecuador.

DEDICATORIA

A Dios que siempre nos ha acompañado, siendo su amor la fuente de energía para alcanzar nuestras metas.

A nuestros padres, por su comprensión y ayuda incondicional, quienes siempre nos inculcaron perseverancia con valores éticos, permitiéndonos iniciar nuestra vida profesional, y a nuestros hermanos que siempre han apoyado nuestras decisiones y retos.

TRIBUNAL DE GRADUACION

Ing. Carlos Valdivieso

Profesor de Materia de Graduación

Ing. Hugo Villavicencio V.

Delegado del Decano

Ing. Jorge Aragundi

Sub-Decano FIEC

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".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

____

Mariano de Jesús Malavé Lindao

____

Manuel Rogelio Nevárez Toledo

____

Pedro Fabián Vallejo Mero

RESUMEN

El objetivo principal es introducirse en el campo de la robótica, utilizando el kit de Lego Mindstorms NXT y herramientas de softwares como Matlab y Simulink. El proyecto que a continuación se presenta consiste en el diseño y construcción de un Brazo Robótico con Banda Transportadora, este proyecto está basado en el principio de una envasadora de bebidas.

El brazo tiene dos etapas de secuencia, una para detectar que los recipientes que van a circular por la banda están con el nivel correcto y la segunda cuando tiene un nivel incorrecto. El sensor de luz cumple la función de detectar la presencia del recipiente; y el sensor de tacto se usa para detener la secuencia del brazo robótico. En la programación de Matlab se crea un algoritmo para obtener la adquisición de datos del sensor ultrasónico en tiempo real.

Se utiliza el servo HITEC-HS311 para mover la banda transportadora, el cual es controlado por el PIC-16F628A. Además se utiliza dos sensores infrarrojos como entradas en el PIC, con la función de hacer dos pausas al servo, la primera para que el sensor ultrasónico lea el nivel del recipiente y la segunda para que el brazo agarre el recipiente y cumpla con la secuencia.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El presente proyecto tiene como finalidad el diseño y construcción de un Brazo Robótico con Banda Transportadora. La cual es una de las muchas aplicaciones que se puede realizar con Lego en el campo de la robótica. La implementación del proyecto se realizará con las piezas de Lego NXT, que incluyen cuatro sensores (ultrasonido, luz, tacto y sonido) y tres servomotores. El Brazo Robótico será controlado por Lego Mindstorms NXT y para el circuito de control de la Banda Transportadora se utilizará el PIC 16F628A.

En el primer capítulo, se da un detalle general del proyecto con una descripción de los sensores a utilizar y la función que desempeñan cada uno, de igual manera se define los servomotores utilizados en cada puerto del NXT. Así mismo se menciona la aplicación en la que puede ser utilizada este proyecto en el campo industrial.

En el segundo capítulo, se describe las herramientas de hardware utilizadas, que incluyen equipos y materiales adicionales en la construcción de la Banda Transportadora. Además se detalla MATLAB, que es la principal herramienta de programación del Lego NXT para control del Brazo Robótico, por lo que se da una descripción de las funciones para desarrollar este proyecto. Las herramientas adicionales CYGWIN, GNUARM, NEXTTOOL, NXTOSEK son parte de estas herramientas que trabajan con Matlab. De igual manera se describe las herramientas de software utilizadas para la programación del PIC.

En el tercer capítulo, se describe el prototipo inicial, del cual surgió la idea del proyecto final, él cual es una versión mejorada del primero con funciones más específicas de los sensores de luz y ultrasonido. Por cuanto se describen todos los algoritmos para leer cada uno de los sensores utilizados, de igual forma el algoritmo para la adquisición de datos de sensor de ultrasonido; y, por último el algoritmo de la secuencia del Brazo. También se describe el diseño del circuito controlador para la Banda.

En el cuarto y último capítulo encontramos todos los diagramas electrónicos para el diseño del circuito controlador de la Banda. Todos los valores de las pruebas efectuadas para la configuración correspondiente de los sensores de luz y ultrasonido son descritos.

CAPÍTULO 1

1. DESCRIPCIÓN GENERAL DEL PROYECTO

1.1. Antecedentes

El campo de la tecnología avanza a una velocidad impresionante, esto conlleva a la investigación y desarrollo de nuevas tecnologías. En el mercado existen herramientas como Lego Mindstorms, Vex Robotics y software que nos simplifican el trabajo al momento de diseñar y construir prototipos. Para el campo de la investigación y desarrollo de prototipos ya están a nuestro alcance herramientas de hardware y software que nos permiten el control automático y modelamiento de sistemas en tiempo real, teniendo una predicción de lo que sucede en el medio a través del uso de sensores.

Para desarrollar el siguiente proyecto utilizamos Lego Mindstorms, con la ayuda de Matlab controlaremos el equipo. El manejo de sensores, motores y microcontroladores, es una de las aplicaciones con la cual complementamos el proyecto.

1.2. Descripción del Proyecto

El proyecto consiste en la construcción de una banda transportadora que contiene un brazo robótico, en ella circularán recipientes llenados a diferentes niveles de producto, estos niveles serán medidos con un sensor de ultrasonido. El brazo robótico clasificará los recipientes según el nivel medido.

El prototipo será construido y controlado con Lego Mindstorms NXT y su programación se la realizará con Matlab y Simulink; el brazo tendrá 2 grados de libertad, utilizando 2 servomotores para mover la base de brazo y un motor que controlará el gipper o garra. Además contará con un sensor de luz para detectar la presencia de los recipientes.

La banda transportadora será controlada por un microcontrolador independiente, el que tendrá como dispositivo de entrada un sensor de luz infrarrojo, como dispositivo de salida un servomotor y un potenciómetro para controlar la velocidad de la banda.

1.3. Aplicaciones

Este proceso es el utilizado en el control de calidad de algunas envasadoras de bebidas, la mayor parte de los envases que contienen niveles inferiores al establecido por el fabricante, son rechazadas, produciendo perdidas en la producción.

Nuestro sistema propone que los envases que no completan el llenado, puedan ser retornados al la línea de producción para que cumpla los requerimientos del nivel de producto.

CAPÍTULO 2

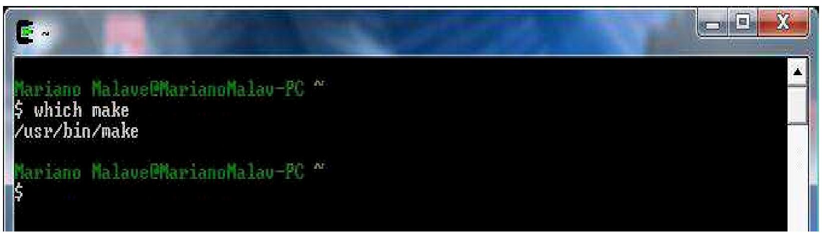
2. FUNDAMENTO TEÓRICO

2.1. Requerimientos para aplicación del Proyecto

Para el desarrollo del proyecto se usaron varias herramientas de hardware y software, el Kit de Lego Mindstorms NXT tiene su software de programación, que funciona en base a LabView, las instrucciones son muy sencillas y es utilizado para la enseñanza a niños mayores de 10 años de edad, en nuestro caso usaremos herramientas de programación alternativas, de esta forma trabajaremos un en entorno de diseño e ingeniería.

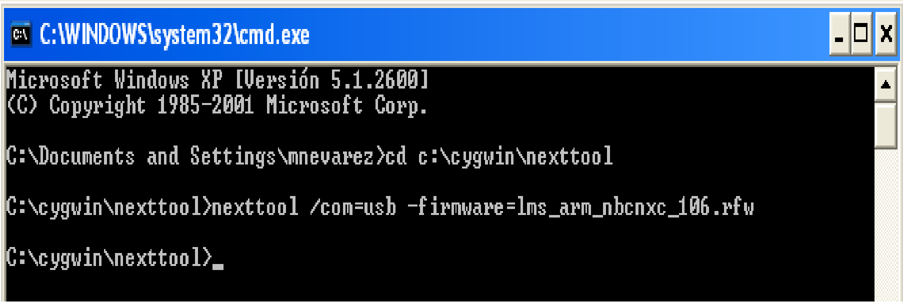
En las herramientas de software utilizamos el programa **Cygwin** el que nos servirá para ejecutar instrucciones en un entorno similar LINUX, es básicamente un simulador de UNIX que ejecuta los programas GNU ARM y Nexttool.

Fig. 2.1 Ventana de Cygwin



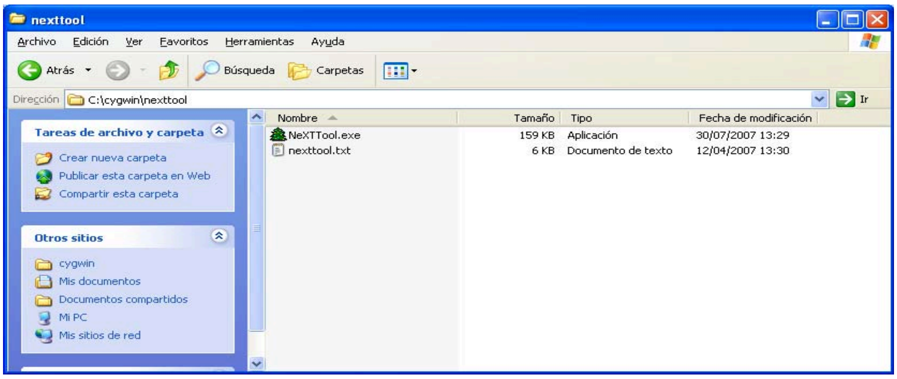
GNU ARM es un compilador, que convierte los archivos creados desde matlab (archivos *.m) o cualquier otro programador que use lenguaje C o C++. Los archivos creados por GNU tienen extensión *.rxe, los cuales son compatibles con el controlador de lego NXT.

Fig 2.2 Ventana del Command Prompt de Windows



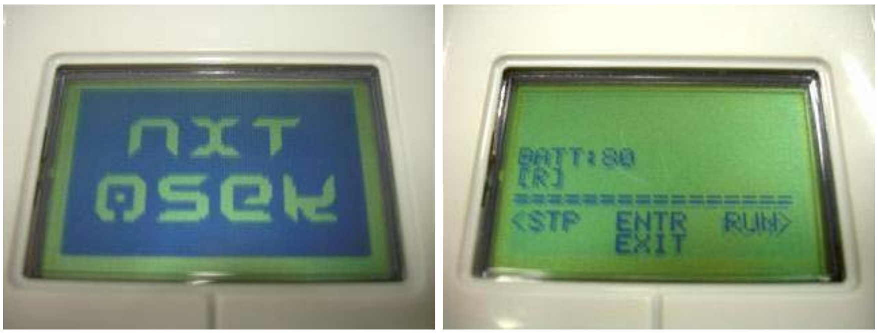
Para cargar los programas en el NXT necesitamos de NeXTTool.exe, que es un software ejecutable desde Cygwin o DOS, el cual también nos permitirá hacer actualizaciones de firmware. Podemos utilizar el cmd de Windows (Command Prompt) para ejecutar las instrucciones de Nexttool.

Fig. 2.3 Archivo ejecutable de NextTool



Teniendo claro cómo funcionan estas herramientas de software, necesitamos ejemplos de programas para realizar pruebas en el NXT, utilizaremos **NXTOsek**. Es un paquete de ejemplos que nos permite cambiar el entorno de Lego Mindstorms en el NXT, modificando su apariencia y la forma de visualización de los programas en el display.

Fig. 2.4 Display de NXT Osek



La mayor parte de sus ejemplos pueden ser cargados a través de Cygwin, pero sus códigos no pueden ser manipulados.

Uno de sus ejemplos mas importantes es **ECRobot** (Embedded Coder Robot NXT), el cual funciona con Matlab y Simulink, su código de programas se puede manipular usando el work space de matlab y visualizar gráficamente en Simulink. En esta nueva etapa dejamos de lado al Cygwin, Nexttool y GNU ARM para utilizar Matlab y Simulink.

Primero necesitamos instalar la librería “RWTHMindstormsNXT” en el toolbox de Matlab y los ejemplos “RWTH-MindstormsNXTExamples” que se los puede descargar en la página oficial de Matlab. Esta librería adicionará bloques de Lego a Simulink. Ahora la programación del NXT se la puede hacer desde Simulink, tomando en cuenta que Matlab llama indirectamente a Cygwin, Nexttool y GNU ARM para compilar, cargar y ejecutar los programas creados desde SimulinK.

2.2. Herramientas de Hardware utilizadas.

En esta sección se detallan los equipos utilizados para la construcción del brazo robótico y la banda transportadora, para el primero se uso Lego Mindstorms, mientras para la segunda parte se usó materiales y componentes apropiados para la implementación de prototipos electrónicos.

2.2.1. Lego Mindstroms

Lego Mindstroms es un juego de robótica para niños fabricado por la empresa Lego, el cual posee elementos básicos de las teorías robóticas, como la unión de piezas y la programación de acciones, en forma interactiva. Este robot fue comercializado por primera vez en septiembre de 1998.

Comercialmente se publicita como *Robotic Invention System*, en español Sistema de Invención Robotizado (RIS). También se vende como herramienta educacional, lo que originalmente se pensó en una sociedad entre Lego y el MIT. La versión educativa se llama *Lego Mindstorms for Schools*, en español Lego

Mindstorms para la escuela y viene con un software de programación basado en la GUI de Robolab.

Lego Mindstorms puede ser usado para construir un modelo de sistema integrado con partes electromecánicas controladas por computador. Prácticamente todo puede ser representado con las piezas tal como en la vida real, como un elevador o robots industriales.

2.2.1.1. NXT

El bloque NXT es una versión mejorada a partir de Lego Mindstorms RCX, que generalmente se considera la predecesora y precursora de los bloques programables de Lego. Debido a la comercialización de los bloques programables, Lego vendió la generación NXT en dos versiones: Retail Version y Education Base Set. Una ventaja de la versión Educacional es que se incluía las baterías recargables y el cargador, pero esta misma versión debía comprar el software según el tipo de licencia: Personal, Sala de clases, Sitio.

Además, Lego dispuso de varios kits para desarrolladores según las características de los programas que estuvieran desarrollando,

- *Software Developer Kit* (SDK), que incluía los controladores del puerto de USB, archivos ejecutables y referencia a los bytecodes.
- *Hardware Developer Kit* (HDK), incluía la documentación y esquemas para los sensores de NXT.

- *Bluetooth Developer Kit* (BDK), documentos de los protocolos usados para la comunicación Bluetooth.

2.2.1.2. Microcontrolador

El microcontrolador que posee es un ARM7 de 32 bits, que incluye 256 Kb de memoria Flash y 64 Kb de RAM externa, la cual a diferencia del bloque RCX, posee mayores capacidades de ejecución de programas, evitando que los procesos inherentes de varios paquetes de datos colisionen y produzcan errores y un posible error en la ejecución del software. Su presentación es similar al Hitachi H8 ya que se encuentra en el circuito impreso del bloque, junto a la memoria FLASH.

2.2.1.3. Entradas y salidas

En el bloque de NXT existen cuatro entradas para los sensores, que a diferencia del bloque RCX, éstos poseen 6 vías de entradas, haciéndolos incompatibles con las entradas de RCX, sin embargo, el empaque de NXT incluye el adaptador para que los sensores de RCX sean compatibles con NXT.

Las salidas de energía aún son tres localizadas en la parte posterior del bloque, haciendo que la conexión para los motores y partes móviles sean de más fácil acceso.

2.2.1.4. Comunicaciones

El bloque de NXT puede comunicarse con el computador mediante la interfaz de USB que posee, la cual ya viene en la versión 2.0. Además, para comunicarse con otros robots en las cercanías posee una interfaz Bluetooth que es compatible con al Clase II v 2.0. Esta conectividad con *Bluetooth* no tan sólo permite conectarse con otros bloques, sino también con computadores, palms, teléfonos móviles, y otros aparatos con esta interfaz de comunicación.

Dentro de las posibilidades de conexión se encuentran

- Conectar hasta tres dispositivos distintos,
- Buscar y conectarse a otros dispositivos que posean Bluetooth.
- Recordar dispositivos con los cuales se ha conectado anteriormente para conectarse más rápidamente,
- Establecer el bloque NXT como visible o invisible para el resto de los dispositivos.

2.2.1.5. Sensor de luz

Un **sensor fotoeléctrico** es un dispositivo electrónico que responde al cambio en la intensidad de la luz. Estos sensores requieren de un componente emisor que genera la luz, y un componente receptor que "ve" la luz generada por el emisor. Todos los diferentes modos de medición se basan en este principio de funcionamiento. Están diseñados especialmente para la detección, clasificación y posicionado de objetos; la detección de formas, colores y diferencias de superficie, incluso bajo condiciones ambientales extremas.

Los sensores de luz se usan para detectar el nivel de luz y producir una señal de salida representativa respecto a la cantidad de luz detectada. Un sensor de luz incluye un transductor fotoeléctrico para convertir la luz a una señal eléctrica y puede incluir electrónica para condicionamiento de la señal, compensación para sensibilidades cruzadas como la temperatura y formateo de la señal de salida.

El sensor de luz más común es el LDR -Light Dependant Resistor o Resistor dependiente de la luz-.Un LDR es básicamente un resistor que cambia su resistencia cuando cambia la intensidad de la luz.

2.2.1.6. Fuentes de luz

Hoy en día la mayoría de los sensores fotoeléctricos utilizan LEDs como fuentes de luz. Un LED es un semiconductor, eléctricamente similar a un diodo, pero con la característica de que emite luz cuando una corriente circula por él en forma directa.

Los LEDs pueden ser construidos para que emitan en verde, azul, amarillo, rojo, infrarrojo, etc. Los colores más comúnmente usados en aplicaciones de sensado son rojo e infrarrojo, pero en aplicaciones donde se necesite detectar contraste, la elección del color de emisión es fundamental, siendo el color más utilizado el verde. Los fototransistores son los componentes más ampliamente usados como receptores de luz, debido a que ofrecen la mejor relación entre la sensibilidad a la luz y la velocidad de respuesta, comparado con los componentes fotorresistivos, además responden bien ante luz visible e infrarroja. Las fotocélulas son usadas cuando no es necesaria una gran sensibilidad, y se utiliza una fuente de luz visible. Por otra parte los fotodiodos donde se requiere una extrema velocidad de respuesta.

Tabla 2.1.1 Fuentes de luz habituales

Color	Rango	Características
INFRARROJO	890…950 nm	No visible, son relativamente inmunes a la luz ambiente artificial. Generalmente se utilizan para detección en distancias largas y ambientes con presencia de polvo.
ROJO	660…700 nm	Al ser visible es más sencilla la alineación. Puede ser afectado por luz ambiente intensa, y es de uso general en aplicaciones industriales.
VERDE	560…565 nm	Al ser visible es más sencilla la alineación. Puede ser afectado por luz ambiente intensa, generalmente se utiliza esta fuente de luz para detección de marcas.

2.2.1.7. Sensor de Luz de Lego Mindstorms

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 oscuridad, midiendo la intensidad de la luz le permite a nuestro robot "ver" en blanco y negro.

Fig. 2.1.1 Sensor de luz NXT

El sensor de luz permite tomar una muestra de luz mediante un bloque modificado que un extremo trae un conductor eléctrico y por el otro una cámara oscura que capta las luces. Esta cámara es capaz de captar luces entre los rangos de 0,6 a 760 lux. Este valor lo considera como un porcentaje, el cual es procesado por el bloque lógico, obteniendo un porcentaje aproximado de luminosidad.

El bloque RCX calcula con la fórmula $Luz = 146 - RAW / 7$ para determinar el porcentaje obtenido por la lectura de la luz, tomando una muestra cada 2,9 ms, siendo leído en 100 us. el valor que se lee a partir del sensor.

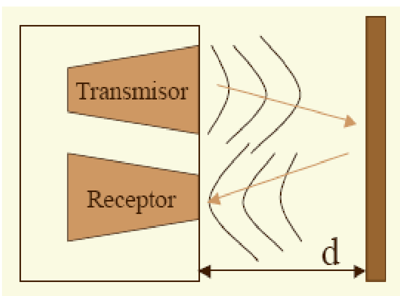
Debido a que este sensor capta grados de luminosidad, no es capaz de distinguir colores, sólo captando la existencia del blanco (claridad), negro (oscuridad) y los tonos de grises que corresponden a los distintos porcentajes de luz existentes en el medio.

2.2.1.8. Sensor de Ultrasonido

Los ultrasonidos son antes que nada sonido, exactamente igual que los que oímos normalmente, salvo que tienen una frecuencia mayor que la máxima audible por el oído humano. Ésta comienza desde unos 16 Hz y tiene un límite superior de aproximadamente 20 KHz, mientras que nosotros vamos a utilizar sonido con una frecuencia de 40 KHz. A este tipo de sonidos es a lo que llamamos Ultrasonidos.

El funcionamiento básico de los ultrasonidos como medidores de distancia se muestra de una manera muy clara en el siguiente esquema, donde se tiene un receptor que emite un pulso de ultrasonido que rebota sobre un determinado objeto y la reflexión de ese pulso es detectada por un receptor de ultrasonidos:

Fig. 2.1.2 Funcionamiento del sensor ultrasónico



La mayoría de los sensores de ultrasonido de bajo coste se basan en la emisión de un pulso de ultrasonido cuyo lóbulo, o campo de acción, es de forma cónica. Midiendo el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda sonora, mediante la fórmula:

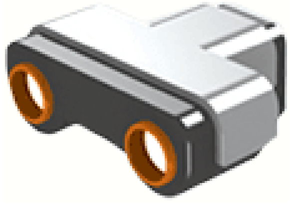
$$d = \frac{1}{2} V \cdot t$$

donde V es la velocidad del sonido en el aire y t es el tiempo transcurrido entre la emisión y recepción del pulso.

2.2.1.9. Sensor de Ultrasonido de Lego Mindstorm

Este sensor le permite a nuestro robot ver y detectar obstáculos así como medir distancias.

Fig. 2.1.3 Sensor ultrasónico del NXT



El sensor Ultrasónico sólo se incluye en el empaque de Lego Mindstorms NXT, y su principal función detectar las distancias y el movimiento de un objeto que se interponga en el camino del robot, mediante el principio de la detección ultrasónica. Este sensor es capaz de detectar objetos que se encuentren desde 0 a 255 cm, con una precisión relativa de +/- 3 cm

Mediante el principio del eco, el sensor es capaz de recibir la información de los distintos objetos que se encuentren en el campo de detección. El sensor funciona mejor cuando las señales ultrasónicas que recibe, provienen de objetos que sean grandes, planos o de superficies duras. Los objetos pequeños, curvos o suaves, como pelotas, pueden ser muy difíciles de detectar. Si en el cuarto se encuentra más de un sensor ultrasónico, los dispositivos pueden interferir entre ellos, resultando en detecciones pobres.

2.2.1.10. Sensor de Tacto

El sensor de contacto permite detectar si el bloque que lo posee ha colisionado o no con algún objeto que se encuentre en su trayectoria inmediata. Al tocar una superficie, una pequeña cabeza externa se contrae, permitiendo que una pieza dentro del bloque cierre un circuito eléctrico comience a circular energía, provocando una variación de energía de 0 a 5 V

.

Fig. 2.1.4 Sensor de Tacto del NXT

En este caso, si la presión supera una medida estándar de 450, mostrado en la pantalla de LCD, se considera que el sensor está presionado, de otro modo, se considera que está sin presión.

2.2.1.11. Sensor de Sonido

El sensor solo detecta la “cantidad” de sonido y no ningún tipo de tono o modulación, pero aun así hay muchas aplicaciones ingeniosas que se le pueden dar.

Fig. 2.1.5 Sensor de Sonido del NXT

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 podria incluir frecuencias que no podemos escuchar pero que el sensor de sonido capta.

Poner el sensor de sonido en modo dBA es más complicado de lo que debería hacer. Lo que tenemos que hacer es conectar una variable lógica puesta en TRUE al conector dBA del bloque del sensor de sonido como se ve en esta pantalla (click para agrandar).

Los decibeles se miden en una escala logarítmica medio complicada, por lo cual este sensor por defecto nos regresa, como ya dije, 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 más fácil para probar que lectura nos da este sensor es conectarlo a algún puerto de nuestro bloque programable y seleccionar la opción de “View” y dentro ya sea “Sound Sensor dB” o “Sound Sensor dBA” esto nos indicara la lectura continua que esta teniendo el sensor.

2.2.1.12. Servomotor de Lego Mindstorms NXT

Los motores de la serie Lego Robotics han sido de tres tipos, los cuales son independientes al bloque, lo que entrega movilidad al sistema dinámico según las necesidades de construcción

Fig. 2.1.6 Servomotor del NXT

En la tabla de medición, el motor estándar es más veloz que el de 9 volts, pero este último posee más fuerza para mover el robot, ya que pueden levantar cerca de 240 piezas de 8x8, pero es más lento y a la vez más preciso. El motor Micro es sólo para funciones menores debido a su escaso torque y la mínima velocidad de rotación.

Tabla 2.1.2 Parámetros del Motor del NXT

Motor	Velocidad normal (RPM)	Torque (kg/cm)	Velocidad estándar (RPM)
Estándar	3240	1,760	40
9 voltios	370	3.840	15
Micro	36	0,128	36
Tabla correspondiente a las mediciones para los distintos motores desmontables de Lego Robotics			

Los motores desmontables son alimentados mediante cables que poseen conductores eléctricos que transmiten la energía a los inductores. Como son motores paso a paso, el sentido de conexión no entrega la misma dirección de movimiento.

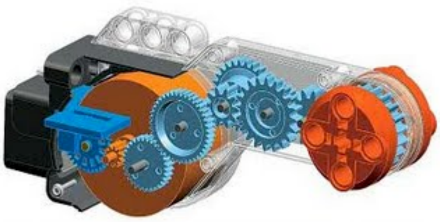
El modelo NXT usa servo motores, los cuales permiten la detección de giros de la rueda, indicando los giros completos o medios giros, que es controlado por el software

Estos motores se conectan al bloque programable a través de los puertos A, B y C. Los tres motores pueden estar conectados al bloque programable y usándose al mismo tiempo.

Los servos además de incluir un motor eléctrico convencional también incluyen un **sensor de posición**. Este sensor nos permite saber a qué velocidad se está moviendo nuestro motor, y corregirla si es necesario. Además podemos saber exactamente cuántos grados a girado el motor en todo momento. Con esto tenemos un **control muy preciso del** movimiento de nuestro robot.

El sensor de posición además de servir para controlar la velocidad y avance de nuestro robot también es útil en sí mismo y nos permite usar los motores del NXT como sensores de movimiento.

Fig. 2.1.7 Diseño interno del servomotor de lego NXT



El motor internamente tiene un tren de engranes para subir la torca del motor. Esto es lo que lo hace un poquito más grande que un motor normal.

2.2.2. Equipos adicionales utilizados

El Kit de Lego NXT solo permite conectar 3 motores y 4 sensores en sus puertos, para tener un control mas completo del proceso se añadió 2 sensores de luz infrarroja y un servomotor adicional, los cuales son controlados por un microcontrolador. Todo esto estará sincronizado con el brazo robótico y la banda transportadora.

2.2.2.1. Pic16F628A

Es un microcontrolador CMOS FLASH de 8 bits de arquitectura RISC capaz de operar con frecuencias de reloj hasta de 20 MHz, fácil de programar. Posee internamente un oscilador de 4 MHz y un circuito de Power-On Reset que eliminan la necesidad de componentes externos y expanden a 16 el número de pines que pueden ser utilizados como líneas I/O (entrada/salida; Input/ Output) de propósito general. Proporciona una memoria de datos EEPROM de 128x8 (128 Bytes), una memoria de programa FLASH de 2024x14 (2K con 14 bits por localidad), una memoria de datos RAM de propósito general de 224x8, un módulo CCP (captura/comparación/PWM), un USART, 3 comparadores análogos, una referencia de voltaje programable y tres temporizadores. Estas y otras características lo hacen ideal en aplicaciones automotrices, industriales, y de electrónica de consumo, así como en equipos e instrumentos programables de todo tipo.

Fig. 2.1.8 Configuración de pines del PIC16F628A

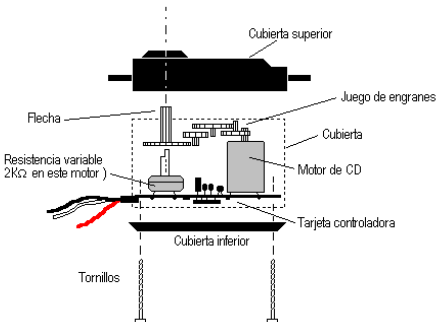
2.2.2.2. Servomotor Hitec HS311

Los servos son un tipo especial de motor de D.C. que se caracterizan por su capacidad para posicionarse de forma inmediata en cualquier posición dentro de su intervalo de operación. Para ello, el servomotor espera un tren de pulsos que se corresponde con el movimiento a realizar. Están generalmente formados por un amplificador, un motor, un sistema reductor formado por ruedas dentadas y un circuito de realimentación, todo en una misma caja de pequeñas dimensiones. El resultado es un servo de posición con un margen de operación de 180° aproximadamente.

Fig. 2.1.9 Servomotor HITEC HS311

El motor del servo tiene algunos circuitos de control y un potenciómetro conectado al eje central del motor. Este potenciómetro permite a la circuitería de control, supervisar el ángulo actual del servo motor. El eje del servo es capaz de girar de 0 a 180 grados. Normalmente, en algunos llegan a los 210 grados, pero varía según el fabricante.

Fig. 2.1.10 Diseño interno de un servomotor HITEC



Un servo normal no es mecánicamente capaz de retornar a su lugar, si hay un mayor peso que el sugerido por las especificaciones del fabricante.

El voltaje aplicado al motor es proporcional a la distancia que éste necesita viajar. Así, si el eje necesita regresar una distancia grande, el motor regresará a toda velocidad. Si este necesita regresar sólo una pequeña cantidad, el motor girará a menor velocidad. A esto se le denomina **control proporcional**.

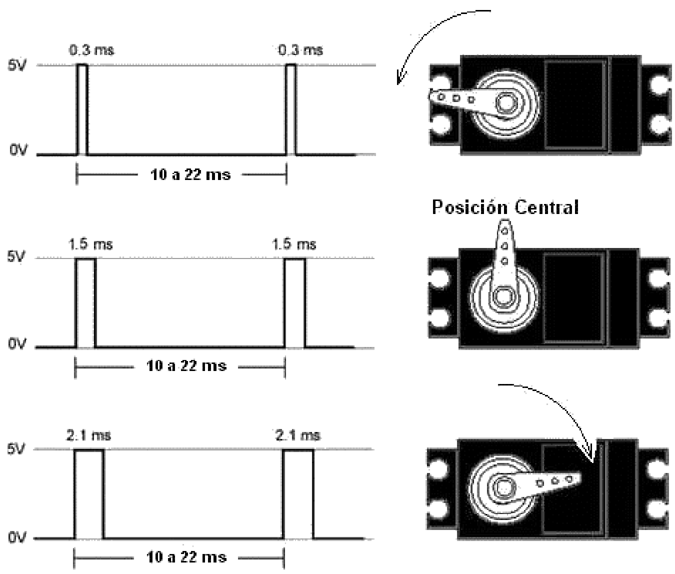
2.2.2.3. Control de posición por PWM

La **modulación por anchura de pulso**, PWM (*Pulse Width* **Modulation), es una de los sistemas más empleados para el** control de servos. Este sistema consiste en generar una onda cuadrada en la que se varía el tiempo que el pulso está a nivel alto, manteniendo el mismo período (normalmente), con el objetivo de modificar la posición del servo según se desee.

Para la generación de una onda PWM en un microcontrolador, lo más habitual es usar un *timer* y un comparador (interrupciones asociadas), de modo que el microcontrolador quede libre para realizar otras tareas, y la generación de la señal sea automática y más efectiva. El mecanismo consiste en programar el *timer* con el ancho del pulso (el período de la señal) y al comparador con el valor de duración del pulso a nivel alto. Cuando se produce una interrupción de *overflow* del *timer*, la subrutina de interrupción debe poner la señal PWM a nivel alto y cuando se produzca la interrupción del comparador, ésta debe poner la señal PWM a nivel bajo. En la actualidad, muchos microcontroladores, como el 68HC08, disponen de hardware específico para realizar esta tarea, eso sí, consumiendo los recursos antes mencionados (*timer* y comparador).

El sistema de control de un servo se limita a indicar en que posición se debe situar. Esto se lleva a cabo mediante una serie de pulsos tal que la duración del pulso indica el ángulo de giro del motor. Los valores más generales se corresponden con pulsos de entre 1 ms y 2 ms de anchura, que dejarían al motor en ambos extremos (0º y 180º). El valor 1.5 ms indicaría la posición central o neutra (90º), mientras que otros valores del pulso lo dejan en posiciones intermedias. Estos valores suelen ser los recomendados, sin embargo, es posible emplear pulsos menores de 1 ms o mayores de 2 ms, pudiéndose conseguir ángulos mayores de 180°. Si se sobrepasan los límites de movimiento del servo, éste comenzará a emitir un zumbido, indicando que se debe cambiar la longitud del pulso.

Fig. 2.1.11.A Diagrama de pulso para controlar un servomotor



El período entre pulso y pulso (tiempo de OFF) no es crítico, e incluso puede ser distinto entre uno y otro pulso. Se suelen emplear valores ~ 20 ms (entre 10 ms y 30 ms). Si el intervalo entre pulso y pulso es inferior al mínimo, puede interferir con la temporización interna del servo, causando un zumbido, y la vibración del eje de salida. Si es mayor que el máximo, entonces el servo pasará a estado dormido entre pulsos. Esto provoca que se mueva con intervalos pequeños.

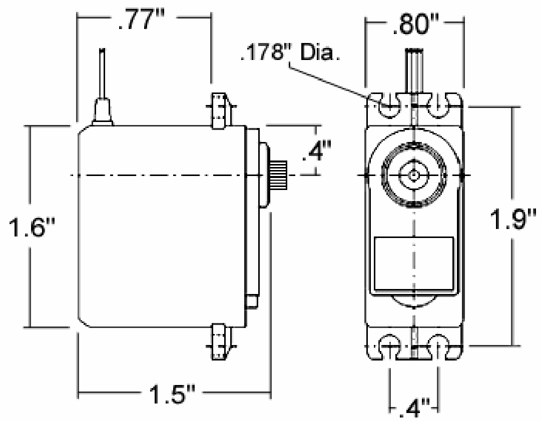
Es importante destacar que para que un servo se mantenga en la misma posición durante un cierto tiempo, es necesario enviarle continuamente el pulso correspondiente. De este modo, si existe alguna fuerza que le obligue a abandonar esta posición, intentará resistirse. Si se deja de enviar pulsos (o el intervalo entre pulsos es mayor que el máximo) entonces el servo perderá fuerza y dejará de intentar mantener su posición, de modo que cualquier fuerza externa podría desplazarlo.

Para controlar la banda del proyecto, utilizaremos el servomotor HITEC HS311, modificado para que gire a 360 grados, sus características de funcionamiento se mencionan a continuación.

Tabla 2.1.3 Parámetros del servo HITEC HS311

Parámetros	**Valor**
Pulse requerido	3 to 5 V
Voltaje de operación	4.8 - 6./0 V
Velocidad de operación (6.0V)	0.15 seg/60 grados
Torque (6.0V)	49.60 oz/in (4.5 kg/cm)
Corriente consumida (6.0V)	7.7 mA/idle y 180 mA/60
Bearing Type	Top/Resin Bushing
Gear Type	Nylon
360 Modificable	Si
Longitud del Conector	11.81” (300 mm)
Peso	1.52 oz. (43 g)

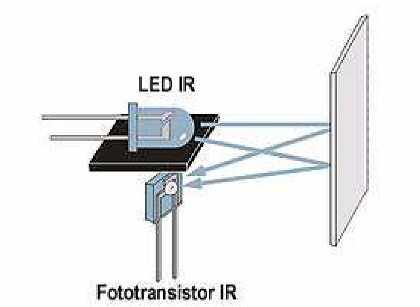
Fig. 2.1.11.B Dimensiones de un servomotor



2.2.2.4. Sensor de luz Infrarrojo

Están diseñados especialmente para la detección, clasificación y posicionado de objetos; la detección de formas, colores diferencias de superficie, incluso bajo condiciones ambientales extremas. En su interior contiene dos elementos, uno es un diodo emisor de luz infrarroja y el otro un fototransistor, este último mide la cantidad de luz infrarroja en algún objeto, esta medición depende de la claridad del objeto a sensar.

Fig. 2.1.12 Funcionamiento de un sensor de luz

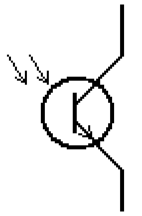


La IR es un tipo de luz que los seres humanos no podemos ver. Es conveniente cuando no desea que la gente vea su sensor (por ej., en los sistemas de seguridad), también la luz natural no afecta al funcionamiento del sensor.

2.2.2.5. El Fototransistor

Se llama **fototransistor** a un transistor sensible a la luz, normalmente a los infrarrojos. La luz incide sobre la región de base, generando portadores en ella. Esta carga de base lleva el transistor al estado de conducción. El fototransistor es más sensible que el fotodiodo por el efecto de ganancia propio del transistor.

Fig. 2.1.13 El Fototransistor, simbología y encapsulado



Un fototransistor es igual a un transistor común, con la diferencia que el primero puede trabajar de 2 formas:

- Como transistor normal con la corriente de base Ib (modo común).
- Como fototransistor, cuando la luz que incide en este elemento hace las veces de corriente de base. Ip (modo de iluminación).
- Puede utilizarse de las dos en formas simultáneamente, aunque el fototransistor se utiliza principalmente con el pin de la base sin conectar.

En el mercado se encuentran fototransistores tanto con conexión de base como sin ella y tanto en cápsulas plásticas como metálicas (TO-72, TO-5) provistas de una lente.

Se han utilizado en lectores de cinta y tarjetas perforadas, lápices ópticos, etc. Para comunicaciones con fibra óptica se prefiere usar detectores con fotodiodos p-i-n. También se pueden utilizar en la detección de objetos cercanos cuando forman parte de un sensor de proximidad.

Se utilizan ampliamente encapsulados conjuntamente con un LED, formando interruptores ópticos (*opto-switch*), que detectan la interrupción del haz de luz por un objeto. Existen en dos versiones: de transmisión y de reflexión.

Para obtener un circuito equivalente de un fototransistor, basta agregar a un transistor común un fotodiodo

2.3. Herramientas de Software

El proyecto consta de dos etapas, la primera cosiste en armar un prototipo y programarlo utilizando el software de programación de Lego Mindstorms, mientras que en la segunda etapa se utilizan las herramientas de Matlab para la programación y adquisición de datos en tiempo real. En esta segunda etapa se incluye la programación del controlador de la banda transportadora, en el que se usó Mikrobasic

2.3.1. Programación de Lego Mindstorms NXT

La programación del Lego Mindstorms se realiza mediante el software que se adjunta en el empaque original, el cual trae el firmware del robot y un programa que emula un árbol de decisiones, para los cuales, el usuario debe programar las acciones a seguir por el robot. El software se encuentra dividido por cada tipo de robot que se puede construir, y que viene recomendado en el empaque.

Una de las principales características de este software de programación, es su entorno visual, el cual emula la construcción por bloques, dando la posibilidad a cualquier usuario aprendiz acostumbrarse rápidamente a la programación de bloque.

Este lenguaje permite las instrucciones secuenciales, instrucciones de ciclos e instrucciones de decisiones, éstas últimas, basadas en los datos reportados por los sensores que se puede añadir al robot.

2.3.2. Lenguajes alternativos de programación

El bloque del Lego Mindstorms como un producto de hardware y software integrado, puede ser programado con varias interfaces, pero todos logrando el mismo fin. Esto se puede realizar mediante la torre de comunicación y utilizando las herramientas correctas para poder acceder al firmware básico de Lego.

Algunas personas han podido ingresar por medio de interfaces rudimentarias a obtener el código básico de la memoria ROM que posee el Lego y así poder tener acceso a programación mediante assembler para poder controlar por ellos mismos el bloque.

Algunos de frameworks más conocidos con el BrickOS, LejOS y Not Quite C

2.3.2.1. BrickOS (o LegOS)[]

BrickOS es una librería de instrucciones y programas que permiten al programador ingresar de forma directa a la BIOS del bloque y allí instalar un micro sistema operativo con su respectivo núcleo operativo y librerías necesarias para enlazar todos los recursos que dispone el bloque. Para ser instalado debe sobreescribir el área donde se encuentra el framework original, pero con este cambio, el bloque puede ser programado en C, C++ y assembler.

BrickOS está soportado en la mayoría de las distribuciones de Linux y en Windows (por CYGWIN), usando el compilador que trae integrado linux (gcc o gcc++), generando el mapa de bytecodes para controlar las acciones del bloque.

En un inicio, este conjunto de programas se llamaba **LegOS**, pero la empresa Lego solicitó un cambio de nombre debido a la semejanza que existía entre ambos nombres.

2.3.2.2. LejOS

LejOS a diferencia de BrickOS, no instala un sistema operativo en reemplazo del firmware del bloque RCX, sino que instala una máquina virtual de Java, lo cual permite el bloque sea programable en el lenguaje Java, por lo cual no dependen de un compilador o un sistema operativo para ser reemplazado. Sin embargo, la transparencia de procesos para el programador es más baja debido a la programación orientada a objetos que restringe LejOS, haciendo que el programa de BrickOS se más utilizado por la transparencia de procesos tanto internos como externos. Esto último repercute en que el programador, utilizando BrickOS, pierde la conciencia de los movimientos que se realiza en forma interna en el bloque, por lo que hace imposible añadir mejores capacidades de programación.

2.3.2.3. Problemas de la adaptación

Un problema generado por el cambio del framework a otro lenguaje es el retardo que pueda existir entre las instrucciones, debido a la emulación de las instrucciones que el conjunto de programas le entrega al bloque. Este retardo fue registrado por Dick Swan y tras algunas pruebas de rendimiento y emulación en software permitió descubrir que el retardo medio para la ejecución de cualquier instrucción, con o sin motor encendido es de 1,75 mseg.

La prueba que realizó fue realizar muchas tareas en la misma cantidad de tiempo, notando la relación lineal de las instrucciones ejecutadas, por lo cual, a mayor cantidad de instrucciones, mayor el tiempo de espera para ejecutar la instrucción.

2.3.3. Matlab y Simulink

Es un ambiente de cómputo, de alta ejecución numérica y de visualización. MATLAB integra el análisis numérico, calculo de matrices, procesamiento de señales, diseño de sistemas de potencia, Mapeo y tratamiento de imágenes, instrumentación y adquisición de datos, identificación de sistemas, graficación entre otras aplicaciones en un ambiente sencillo de utilizar, donde los problemas y sus soluciones son expresadas justamente como están escritas; a diferencia de la programación tradicional. Permite resolver problemas en una fracción de tiempo.

MATLAB, también cuenta con varias familias de soluciones para aplicaciones especificas llamadas toolboxes, que son colecciones de funciones utilizadas para resolver alguna clase particular de problema.

Simulink es una herramienta para el modelaje, análisis y simulación de una amplia variedad de sistemas físicos y matemáticos. Como una extensión de MatLab, Simulink adiciona muchas características específicas a los sistemas dinámicos, mientras conserva toda la funcionalidad de propósito general de MatLab. Simulink tiene dos fases de uso: la definición del modelo y el análisis del modelo. Simulink usa un ambiente gráfico lo que hace sencillo la creación de los modelos de sistemas. 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.

2.3.4. Cygwin, Nexttool GNU ARM y NXTOSEK

A continuación se detallan el funcionamiento de las herramientas instaladas en Matlab.

2.3.4.1. CYGWIN

Cygwin es un entorno de Linux como para Windows. Se compone de dos partes:

- Una DLL (cygwin1.dll), que actúa como una capa de emulación de la API de Linux proporciona importantes funcionalidades de la API de Linux.
- Una colección de herramientas que ofrecen Linux apariencia.

Cygwin se vale de una aplicación gráfica (el setup.exe) para añadir, quitar y actualizar paquetes. La DLL de Cygwin actualmente trabaja con todos los recientes, estrenados comercialmente x86 de 32 bits y 64 bits de Windows, con la excepción de Windows CE.

Cygwin no es una forma de ejecutar aplicaciones de Linux nativo en Windows. Tiene que reconstruir la aplicación de la fuente si desea que se ejecute en Windows.

2.3.4.2. GNU ARM

GNU fue iniciado con el objetivo de crear un sistema operativo completamente libre, está basado en una arquitectura que ha demostrado ser técnicamente estable. El sistema **GNU** fue diseñado para ser totalmente compatible con UNIX.

La cadena de herramientas consiste en el compilador (GCC) y un depurador (para Windows y Linux, GDB sólo para MacOS). La configuración de GNU y el sistema de compilación se compone de varias herramientas diferentes. La configuración de GNU y el sistema de compilación requiere de varios archivos de soporte técnico que deberá incluirse en su distribución.

2.3.5. RWTHMindstormNXT y Embedded Coder Robot NXT

RWTHMindstormNXT es un toolbox que ha sido desarrollado para el control de LEGO ® MINDSTORMS ® robots NXT con MATLAB a través de una conexión inalámbrica Bluetooth o vía USB. Este software es un producto libre de código abierto y está sujeto a la Licencia Pública General GNU (GPL).

Esta toolbox abre posibilidades ilimitadas para proporcionar a los robots de inteligencia artificial y otras mejoras, utilizar las funciones de MATLAB múltiples y cálculos para el procesamiento de señales digitales. Embedded Coder Robot NXT es la librería que proporciona los bloques de Mindstorm NXT para desarrollar un modelo en Simulink, posee una capacidad de modelado para la estrategia de control NXT, dinámica y capacidad de simular y visualizar estos componentes en un modelo 3-D virtual de realidad gráfica.

ECRobot NXT también prevé la Real-Time Workshop Embedded Codificador de implementación de destino con una base de código abierto OSEK RTOS NXT con el hardware real. Esto significa que usted puede experimentar plenamente diseño basado en modelos como el modelado, simulación, el código generación, objetivo construir y probar en una NXT real para el LEGO Mindstorms NXT.

2.3.6. MikroBasic

Es una herramienta de desarrollo que se permite realizar proyectos para microcontroladores PIC. Proporciona una solución fácil para aplicaciones para sistemas embebidos, sin comprometer el rendimiento o el control, desarrolla rápidamente y desplegar aplicaciones complejas.

Originalmente concebido como una herramienta fácil de usar, BASIC se generalizó en microcomputadoras en la década de 1980, y sigue siendo popular hoy en día. BASIC nombre, acuñado en el clásico, la tradición de la informática para producir un acrónimo de Niza, *representa* todo *principiante fines Código de Instrucciones Simbólicas.* Sigue siendo considerado por muchos usuarios de PC a ser el lenguaje de programación fácil de usar. Hoy en día, esta reputación se está desplazando al mundo de los microcontroladores. Basic permite un desarrollo mucho más rápido y más fácil de las solicitudes de consentimiento fundamentado previo en comparación con el lenguaje del microchip Ensamblador MPASM. Al escribir el código de MCU, con frecuencia los programadores de hacer frente a las mismas cuestiones, tales como la comunicación serial, la impresión en la pantalla LCD, la generación de señales PWM, etc. Con el fin de facilitar la programación, se basa en una serie de rutinas built-in de la biblioteca destinados para resolver estos problemas.

En cuanto a la ejecución y el tamaño del programa en cuestión, MPASM tiene una pequeña ventaja en relación con BASIC. Por esta razón hay una opción de combinar BASIC y el código ensamblador. El ensamblador se utiliza comúnmente para las partes del programa en el que el tiempo de ejecución es crítico o que los comandos se ejecuten gran número de veces. Los microcontroladores modernos, como los PIC, ejecutan instrucciones en un solo ciclo de reloj. Si el reloj del microcontrolador es 4MHz, una instrucción en lenguaje ensamblador requiere 250ns x 4 = 1us. Como cada comando básico es técnicamente una secuencia de instrucciones en ensamblador, el tiempo exacto necesario para su ejecución se puede calcular simplemente sumando los tiempos de ejecución de las instrucciones.

2.3.7. Proteus

Es un paquete de software para el diseño de circuitos electrónicos que incluye captura de los esquemas, simulación analógica y digital combinadas y diseño de circuitos impresos. Proteus es un entorno integrado diseñado para la realización completa de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño, simulación, depuración y construcción. El paquete está compuesto por dos programas: ISIS, para la captura y simulación de circuitos; y ARES, para el diseño de PCB's. También permite simular y depurar el funcionamiento de todo el sistema ejecutando el software paso a paso, insertando puntos de ruptura (breakpoints, que también pueden ser generados por el hardware), mirando el contenido de registros y posiciones de memoria, etc. y comprobando si la respuesta del hardware es la correcta. También se simulan herramientas electrónicas, como osciloscopios, analizadores lógicos, voltímetros, etc.

2.3.8. Programador de Microcontroladores PIC Kit 2

El PICkit 2 es un programador fabricado por Microchip© para programar toda su línea de microcontroladores PIC's® desde los PIC10, PIC12, PIC14, PIC16, PIC18, PIC24, dsPIC30 y dsPIC33 (todos los microcontroladores con memoria Flash sin excepción). El programador fue diseñado para programar los microcontroladores en circuito (ICSP) lo que significa que puede programar los microcontroladores montados directamente en tu aplicación y/o protoboard sin necesidad de tener que sacarlo y meterlo cada vez que se modifica el programa.

CAPÍTULO 3

3. IMPLEMENTACIÓN DEL PROYECTO

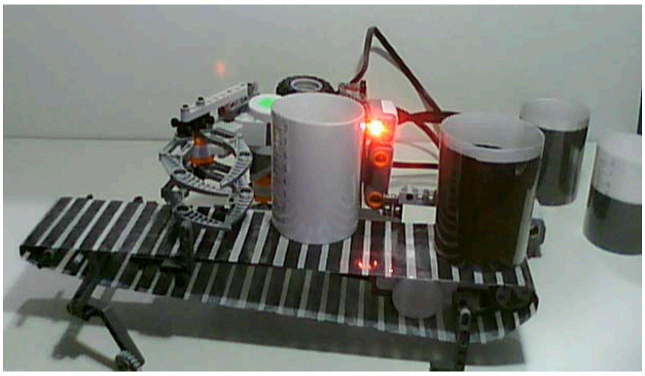
3.1. Prototipo Inicial

Utilizando el principio del llenado de botellas, diseñamos un robot capaz de seleccionar botellas completamente llenas de otras que no contengan el nivel exacto de líquido, esto puede ser utilizado en el control de calidad para un proceso de llenado de botellas. La programación se realizó con las herramientas de Lego Mindstorms, la cual funciona bajo la plataforma de Labview.

3.1.1. Construcción

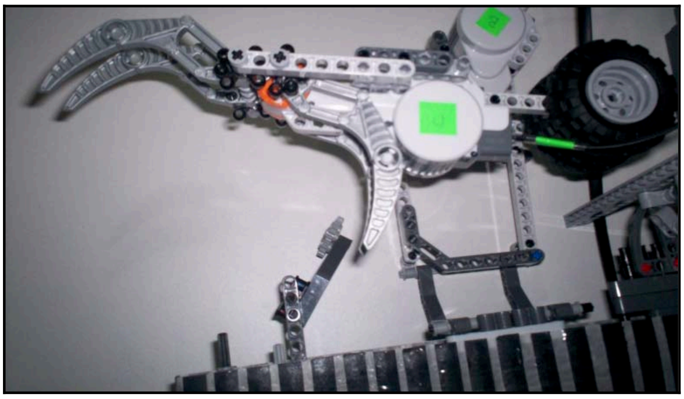
La primera versión de la banda consistía en un brazo selector de envases con diferentes niveles de líquido, estos niveles eran representados con franjas de colores en unos recipientes transparentes, es decir el sensor de luz se lo utilizó para medir estos dos colores (banco y negro). Una escala impresa en los recipientes indicaba el rango de niveles en una escala de 0 a 8cm. Con el sensor ultrasónico detectamos la presencia del recipiente para que el sensor de luz cumpla el objetivo mencionado. Una vez realizadas estas funciones de los sensores, el brazo actúa para coger el recipiente.

Fig. 3.1.1 Pruebas de lectura utilizando sensor infrarrojo



La banda transportadora es movida por un servomotor, aquí se utilizan un sensor de luz y otro de ultrasonido. Para simular el proceso usamos unos cilindros con cartulina de color negro y blanco para representar el nivel de líquido dentro de las botellas. El sensor de ultrasonido (Puerto 2) lo utilizamos para detectar la presencia de las botellas en un lugar de la banda transportadora y en la misma posición ubicamos al sensor de luz (Puerto 1), este nos permite medir el nivel de líquido establecido por el operador.

Fig. 3.1.2 Garra controlada por servomotor



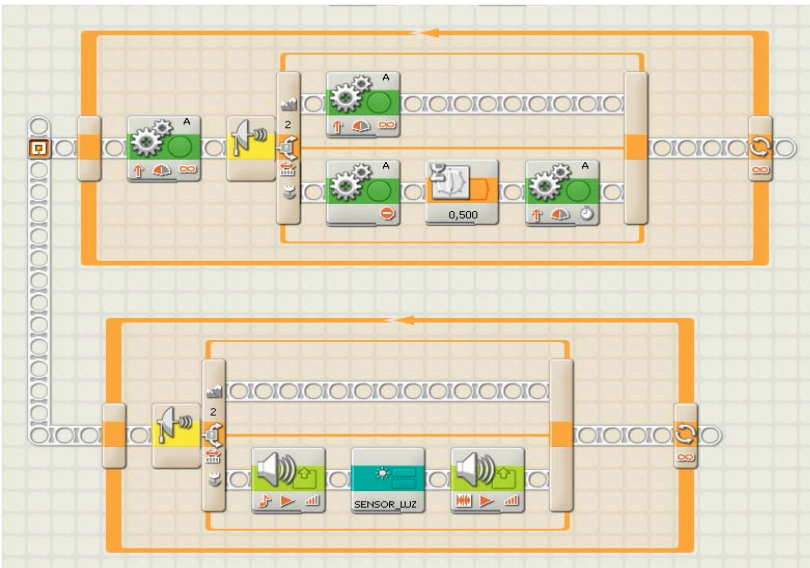
El brazo robótico solo tiene 2 motores, uno controla el gripper y el otro mueve el eje del brazo, mientras que la banda transportadora es controlada por un tercer motor

3.1.2. Programación:

Para programar el equipo usamos el programa de LEGO MINDSTORM NXT, el lazo superior de programa controla al motor de la banda (Motor A), al notar la presencia de un objeto, detiene la banda por 500ms.

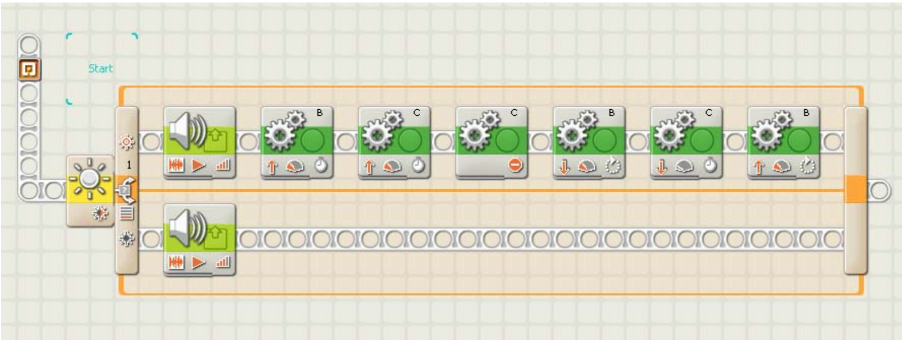
El segundo lazo, controla al brazo, el primer sonido indica la presencia de un objeto y el segundo sonido avisa al operador que ha finalizado el proceso.

Fig. 3.1.3 Programación en Lego Mindstorms de la banda Transportadora



Creamos un bloque llamado SENSOR_LUZ, aquí verificamos el nivel de líquido en la botella, si el nivel es el requerido, simplemente la banda continua girando, en caso de detecta un nivel inferior, activamos en brazo robótico controlado por 2 motores (Motor B y C)

Fig. 3.1.4 Programación en Lego Mindstorms del Brazo Robótico



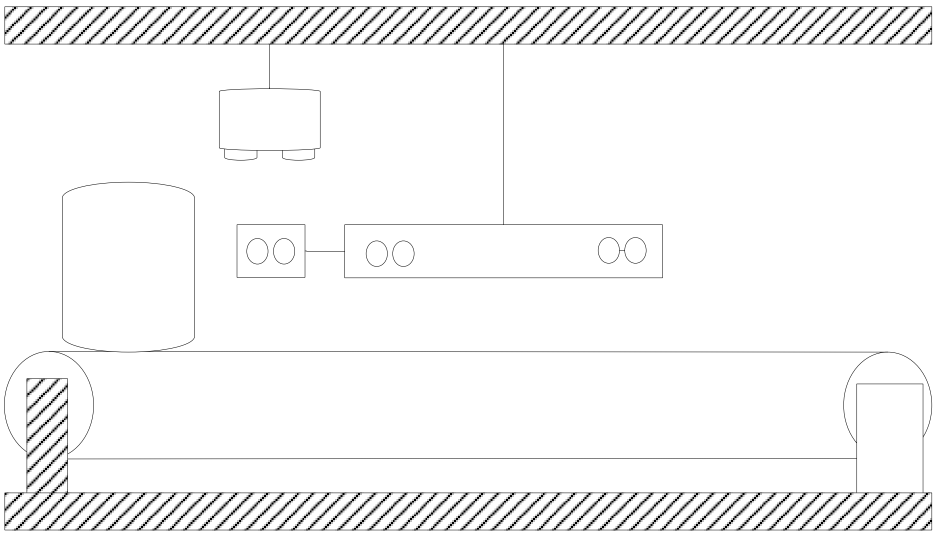
3.2. Descripción del Proyecto Final

Luego de la construcción y programación del prototipo, notamos ciertos errores en el diseño del mismo, ya que el sensor de ultrasonido lo utilizábamos para medir presencia de objetos y no para su función básica que es medir distancias, entonces se decidió hacer un cambie en a ubicación de los sensores.

El sensor de ultrasonido se lo colocó en la parte superior de la banda transportadora, de esta forma lograremos medir la cantidad de producto que se encuentra dentro de los recipientes, de esta forma aprovechamos la funciona del sensor, utilizándolo como sensor de nivel, mientras que el sensor de luz solo detectará presencia de objetos claros en la posición presentada en la figura 3.2.1. También adicionaremos 2 sensores de luz infrarroja a lo largo de la banda para monitorear la presencia de los recipientes en diferentes puntos, estos sensores serán las entradas del circuito controlador de la banda; en el primer punto la banda se detendrá para hacer la mediciones de nivel, mientras que en el segundo punto la banda se detendrá para que actué el brazo robótico en la selección de los recipientes.

El brazo colocara los recientes con nivel inferior de producto en la zona de rechazo, y los recipientes con el nivel correcto los depositará en la zona de producto aceptado.

Fig. 3.2.1: Esquema, ubicación de los sensores y motores.



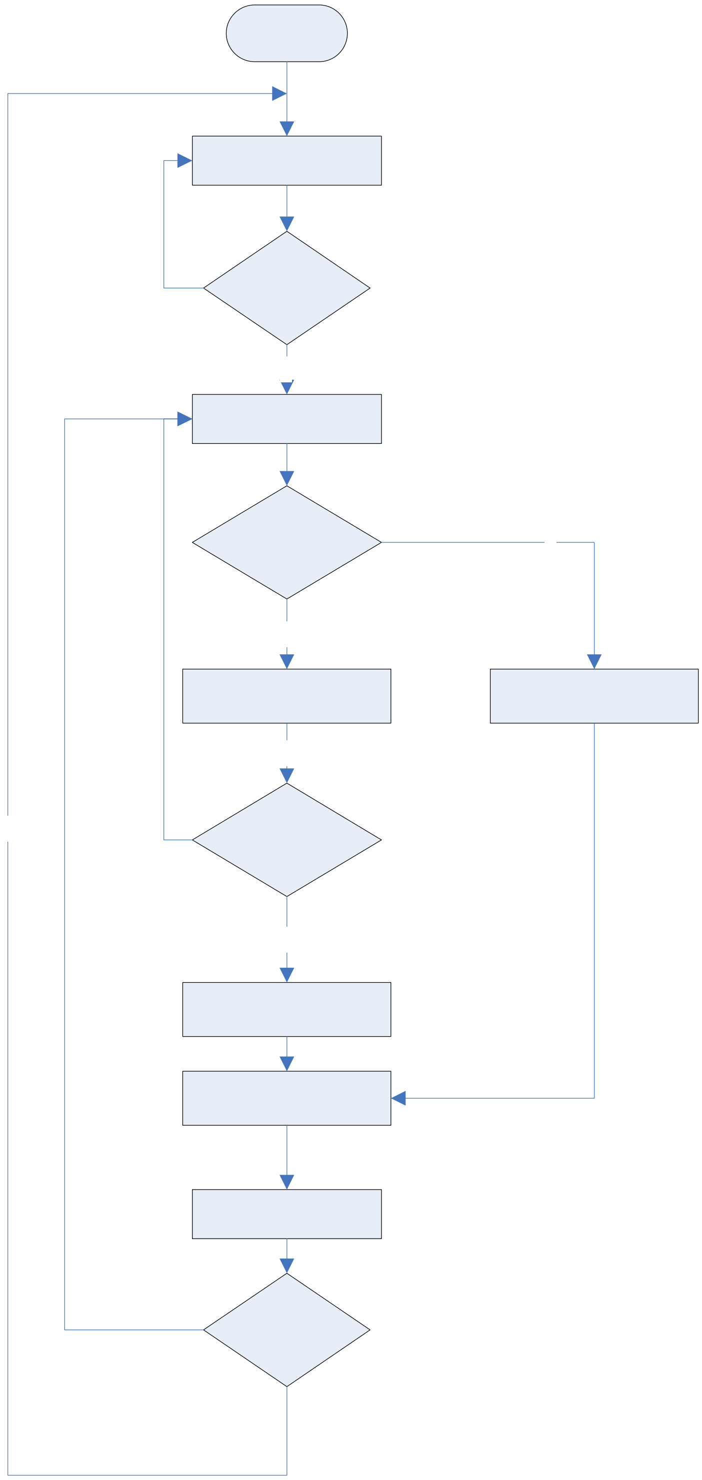
3.3. Programación en Matlab y Simulink

A continuación se detallamos un diagrama de flujo general para el desarrollo del proyecto con la secuencia de la misma, en donde identificamos en que momento actúan los sensores tanto de luz como ultrasonido. Además detallamos los códigos *.m para la programación de la secuencia del brazo robótico y para la adquisición de datos del sensores utilizados. Y por último los bloques realizados para el modelo de nuestro proyecto en Simulink. La comunicación a utilizar entre la computadora y el NXT del proyecto es USB, la cual también la utilizamos para la adquisición de datos correspondientes.

3.3.1. Diagrama de flujo

En el diagrama observamos que la secuencia empieza por el sensor de luz, el cual verifica que cumpla con el valor seteado, si este es el correcto indica que hay un recipiente en la banda transportadora.

Una vez detectado la presencia del objeto, el sensor de ultrasonido lee el nivel del mismo; si este cumple con el nivel establecido el brazo robótico realiza la primera secuencia, caso contrario si esté es incorrecto realiza la segunda secuencia. Esto se realiza dentro de un lazo cerrado hasta que el sensor de tacto detecte el valor de uno, con lo cual se detiene todo el proceso.



3.3.2. Código del archivo *.m

Todos los algoritmos utilizados para la programación de la secuencia del brazo robótico se detallan a continuación:

3.3.2.1. Algoritmo para leer el sensor de luz

Mediante este algoritmo podemos leer instantáneamente el sensor de luz, el cual sirve para realizar las respectivas calibraciones con el objetivo de setear el valor preciso para la aplicación a realizar; valor que ese utiliza para detectar la presencia del recipiente. Una vez detectado el brazo robótico empieza el proceso.

```
COM_CloseNXT all
clear all
close all

portLight     = SENSOR_3;
h = COM_OpenNXT();
COM_SetDefaultNXT(h);

OpenLight (portLight, 'active');
light = GetLight(portLight)
```

```
CloseSensor(portLight)
COM_CloseNXT(h);
```

3.3.2.2. Algoritmo para leer el sensor de ultrasonido

Con este algoritmo leemos instantáneamente el sensor de ultrasonido, para setear el valor para el nivel del recipiente. Para lo cual fijamos dos niveles: el nivel máximo, el cual será al valor definido para que el recipiente este lleno con el nivel correcto; y el nivel mínimo, el cual es el valor incorrecto. Una vez comprobado el nivel continuara el proceso del brazo robótico.

```
COM_CloseNXT all
clear all
close all

portUS      = SENSOR_4;
h = COM_OpenNXT();
COM_SetDefaultNXT(h);
 OpenUltrasonic(portUS);
distancia = GetUltrasonic (portUS)
CloseSensor(portUS);
COM_CloseNXT(h);
```

3.3.2.3. Algoritmo para obtener la adquisición de datos

Durante el tiempo en que se desarrolla el proceso del brazo robótico se está sensando continuamente el sensor de ultrasonido en tiempo real, y estos valores se muestran gráficamente en la pantalla de MatLab, para lo cual usamos el siguiente algoritmo.

```
COM_CloseNXT all
clear all
close all
format compact

portUS      = SENSOR_4;
h = COM_OpenNXT();
COM_SetDefaultNXT(h);
figure('name', 'Next Generation Ultrasound')
set(gca, 'Color', 'black');
hold on

 OpenUltrasonic(portUS, 'snapshot')
n         = 8;
count     = 200;
plotcols  = 8;
```

```
outOfRange = 160;

colors = flipud(hot(8));

data = zeros(1, n);

allX = (1:count+1)';


for i = 1 : count

USMakeSnapshot(portUS)

pause(0.05);

echos = USGetSnapshotResults(portUS);

echos(echos == 255) = outOfRange;

echos = [echos(1); diff(echos)];

data = vertcat(data, echos');

x = allX(1:i+1)';


clf

hold on

set (gca, 'Color', 'black');


axis ([0 count 0 outOfRange])

for j = plotcols : -1 : 1

area (x, data(:, j) , 'FaceColor', colors(j, :))

end
```

```
end

CloseSensor (portUS)
COM_CloseNXT (h);
```

3.3.2.4. Algoritmo para leer Sensor de Tacto

El sensor de tacto lo utilizamos para detener el proceso del brazo robótico, es decir detiene la secuencia del programa en Matlab.

```
COM_CloseNXT all
clear all
close all
port = SENSOR_1;

h = COM_OpenNXT();
COM_SetDefaultNXT(h);

OpenSwitch(port);
GetSwitch(port);

CloseSensor(port);
COM_CloseNXT(h);
```

3.3.2.5. Algoritmo para escuchar sonido

Para escuchar una señal auditivita que indica que el brazo robótico está listo para agarrar otro recipiente se emite un sonido mediante el algoritmo siguiente. Con lo cual sabremos que se a terminado el proceso del brazo y está en la espera de un nuevo recipiente.

```
COM_CloseNXT all
clear all
close all
handle = COM_OpenNXT();
COM_SetDefaultNXT(handle);
NXT_PlayTone(262,350);
pause (0.2);
NXT_PlayTone(440,50);
pause (0.2);
NXT_PlayTone(330,50);
pause (0.2);
COM_CloseNXT(handle);
```

3.3.2.6. Algoritmo para la secuencia del brazo robótico

Toda la secuencia de los servomotores para movilizar el brazo robótico, con todos los parámetros de tiempo para pausa y parada, y puertos a utilizarse en los servos se realiza con el algoritmo siguiente.

```
                    %Inicio de Programa

COM_CloseNXT all
clear all
close all
format compact

%Inicilización de puertos
puertoSW = SENSOR_1;
puertoLH = SENSOR_3;
puertoUS = SENSOR_4;
puerto1 = MOTOR_A;
puerto2 = MOTOR_B;
puerto3 = MOTOR_C;

 %Comunicación entre Matlab y NXT
h = COM_OpenNXT();
COM_SetDefaultNXT(h);


%Formato para gráfica del sensor  ultrasonico
figure('name', 'Next Generation Ultrasound')
set(gca, 'Color', 'black');
hold on


 %Mensaje en pantalla de Matlab de inicio de secuencia del brazo robótico
disp ('Connectando......')

 %Inicio de lazo cerrado que entra con condición verdadera
while true

%Inicialización de variables para graficar el sensor de      ultrasonido
   n          = 8;
   count      = 100;
   plotcols   = 8;
   outOfRange = 16;

   colors = flipud(hot(8));

   data = zeros(1, n);
   allX = (1:count+1)';

   %Lectura de Sensor de Ultrasonido
```

```
OpenUltrasonic(puertoUS);

for i = 1 : count
USMakeSnapshot(puertoUS);
pause(0.05);
echos = USGetSnapshotResults(puertoUS);
echos(echos == 255) = outOfRange;

echos = [echos(1); diff(echos)];

data = vertcat(data, echos');
x = allX(1:i+1);


clf
hold on
set(gca, 'Color', 'black');

axis([0 count 0 outOfRange])

for j = plotcols : -1 : 1
    area(x, data(:, j) , 'FaceColor', colors(j, :))
end


%Lectura de sensor de luz
OpenLight(puertoLH,'ACTIVE');
light = GetLight (puertoLH);


if light > 400



OpenUltrasonic(puertoUS);
pause (1);

%Captura de lectura de sensor de ultrasonido
distancia = GetUltrasonic(puertoUS);
pause (1);

%Condición de lectura de ultrasonido menor que 8
if distancia  < 8
```

```
  %Mensaje en pantalla que recipiente tiene el nivel ok
   disp ('Nivel de Recipiente OK')
   pause (2);

%Servo B se mueve 6000 grados para bajar brazo a nivel de  la banda
   SetMotor(puerto2);
   SetPower(-100);
   SetAngleLimit (6000);
   SendMotorSettings();
   WaitForMotor (puerto2,10);

  % Servo C se mueve 40 grados para agarrar recipiente
   SetMotor(puerto3);
   SetPower(20);
   SetAngleLimit (40);
   SendMotorSettings();
   WaitForMotor (puerto3,2);

   %Servo B se mueve 6000 para regresar a estado inicial
   SetMotor(puerto2);
   SetPower(100);
   SetAngleLimit (6000);
   SendMotorSettings();
   WaitForMotor (puerto2,10);

%Servo A se mueve 4900 grados  para mover base de servo y dejar recipiente
con nivel correcto
   SetMotor(puerto1);
   SetPower(-100);
   SetAngleLimit (4900);
   SendMotorSettings();
   WaitForMotor (puerto1,11);

   %Servo C se mueve 40 grados para dejar recipiente
   SetMotor(puerto3);
   SetPower(-20);
   SetAngleLimit (40);
   SendMotorSettings();
   WaitForMotor (puerto3,2);

%Servo A se mueve 4900 grados para regresar a estado inicial
   SetMotor(puerto1);
   SetPower(100);
   SetAngleLimit (4900);
```

```
   SendMotorSettings();
   WaitForMotor (puerto1,11);


%Sonido para indicar que el brazo está listo para coger otro recipiente
   NXT_PlayTone(262,350);
   pause (0.2);
   NXT_PlayTone(440,50);
   pause (0.2);
   NXT_PlayTone(330,50);
   pause (0.2);




    else

   %Mensaje en pantalla que recipiente tiene el nivel incorrecto
   disp ('Nivel de Recipiente VACIO')

%Servo B se mueve 6000 grados para bajar brazo a nivel de  la banda
   SetMotor(puerto2);
   SetPower(-100);
   SetAngleLimit (6000);
   SendMotorSettings();
   WaitForMotor (puerto2,10);

  % Servo C se mueve 40 grados para agarrar recipiente
   SetMotor(puerto3);
   SetPower(20);
   SetAngleLimit (40);
   SendMotorSettings();
   WaitForMotor (puerto3,2);

   %Servo B se mueve 6000 para regresar a estado inicial
   SetMotor(puerto2);
   SetPower(100);
   SetAngleLimit (6000);
   SendMotorSettings();
   WaitForMotor (puerto2,10);

%Servo A se mueve 9600 grados  para mover base de servo y dejar recipiente
incorrecto
   SetMotor(puerto1);
```

```
    SetPower(-100);
    SetAngleLimit (9600);
    SendMotorSettings();
    WaitForMotor (puerto1,17);


    %Servo C se mueve 40 grados para dejar recipiente
    SetMotor(puerto3);
    SetPower(-20);
    SetAngleLimit (40);
    SendMotorSettings();
    WaitForMotor (puerto3,2);

%Servo A se mueve 96000 grados para regresar a estado inicial
    SetMotor(puerto1);
    SetPower(100);
    SetAngleLimit (9600);
    SendMotorSettings();
    WaitForMotor (puerto1,17);

%Sonido para indicar que el brazo está listo para coger otro recipiente
    NXT_PlayTone(262,350);
    pause (0.2);
    NXT_PlayTone(440,50);
    pause (0.2);
    NXT_PlayTone(330,50);
    pause (0.2);

     end
   else
   end
   end

%Lectura de sensor de tacto para terminar secuencia del brazo
 OpenSwitch(puertoSW);
 GetSwitch(puertoSW);
 switchState = GetSwitch(puertoSW);

if switchState == 1
     NXT_PlayTone(800,450);
    disp ('Finalizado......')
   break
  end
 end
```

```
%Exportaciòn de datos Sensor_US a excel
xlswrite ('DATOS_US',data,'Sheet1','A2');

 %Fin de comunicación de puertos
CloseSensor(puertoSW);
CloseSensor(puertoLH);
CloseSensor(puertoUS);
COM_CloseNXT(h);
```

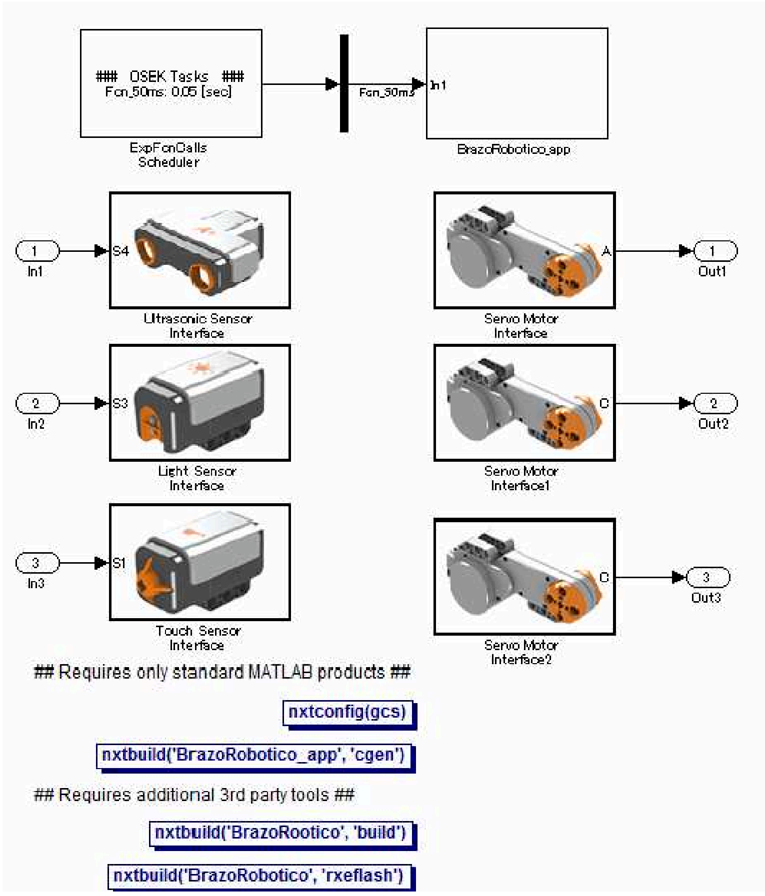
3.3.3. Bloques en Simulink

En la librería ECRobot NXT Blockset de Simulink, encontramos todos los bloques de interface de los tres servomotores (MOTOR_A, MOTOR_B, MOTOR_C) y de los cuatros sensores (SENSOR_SOUND, SENSOR TOUCH, SENSOR LIGHT, SENSOR ULTRASONIC) para la comunicación entre el NXT y la PC; de igual manera se encuentran los bloque de lectura y escritura para los servomotores y sensores. Adicionalmente están los bloques de comunicación para USB o BLUETOOTH.

3.3.3.1. Bloques de interface

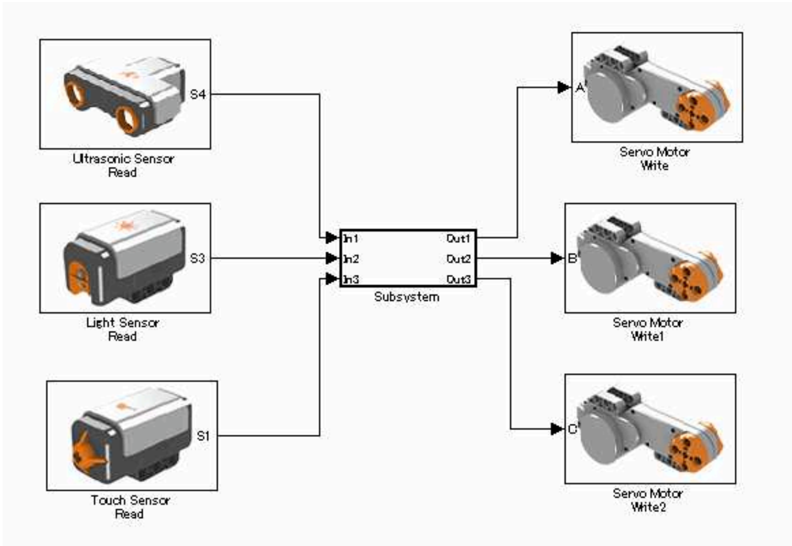
En este modelo se utiliza el bloque **ExpFcnCalls Scheduler** para llamar los subsistemas de cada servomotor de interface. En el bloque **Subsystem** están los tres subsistemas que se creo para controlar cada servomotor. El bloque de utilidades lo usamos para crear lo siguiente:

nxt('BrazoRobotico_app','cgen') el cual genera las interfaces en el bloque **ExpFcnCalls Scheduler;** nxtbuild ('BrazoRobotico_app','build') genera el archivo *.rxe, el cual es el ejecutable para guardar nuestro modelo de simulink en el NXT; y, nxtbuild (‘BrazoRobotico_app’,’rxeflash’, el cual carga el archivo *.rxe para ejecutar el modelo del brazo robotico.



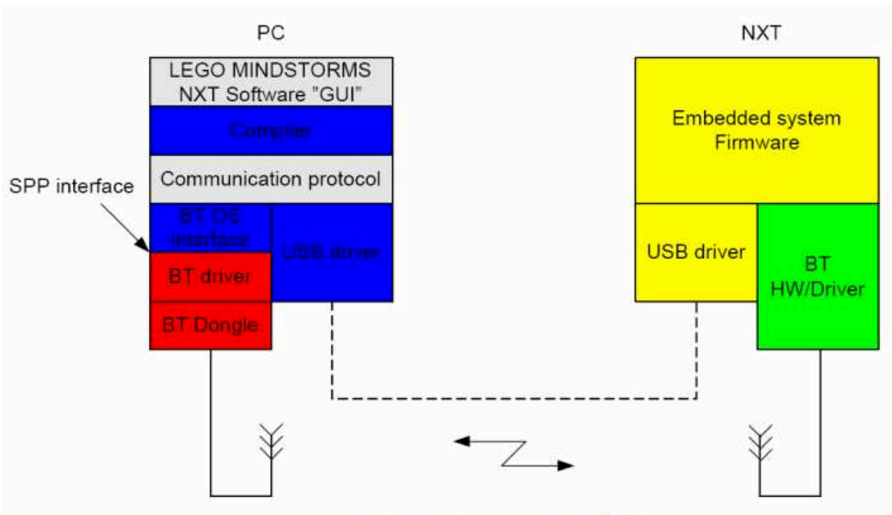
3.3.3.2. Bloques lectura y escritura del modelo

Esta es la pantalla del controlador del modelo, en la cual los valores obtenidos por los sensores son enviados al bloque subsytem, dentro de este bloque se ejecutan las secuencias para el control de los servomotores dependiendo de las variables establecidas.



3.3.4. Comunicación USB

El protocolo utilizado entre la PC y el NXT es USB de tercer tipo; esta interface es usada para la transferencia de datos con una alta velocidad (2.0), con una tasa de transferencia de hasta 480 Mbps (60 MB/s).



Communication Block Diagram (LEGO® ©)

3.4. Diseño electrónico del controlador de la Banda.

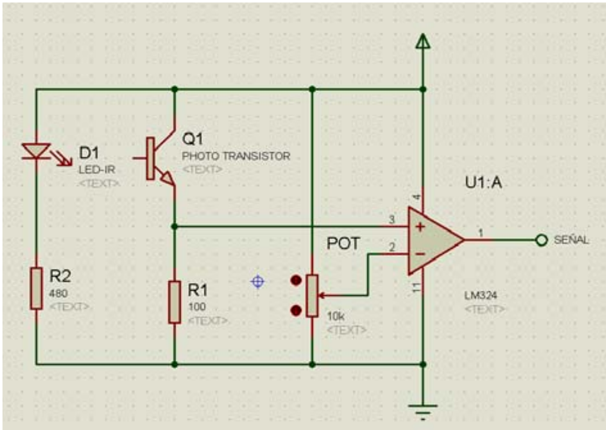
La banda consiste en transportar los envases de producto, a lo largo de la banda se colocarán 2 sensores de luz infrarroja reflectivos, el primero hará que la banda se detenga en el punto A para sensar los niveles de producto utilizando el ultrasonido de Lego Mindstorms, luego de esto se accionará automáticamente la banda hasta que el segundo sensor detecte la presencia del envase en el punto B, aquí el brazo robótico seleccionará los envases según el nivel de producto en su interior, colocándolos en diferentes compartimientos.

3.4.1. Circuito del Sensor de luz

Para obtener señales analógicas en el microcontrolador, proveniente de los sensores de luz, utilizaremos un amplificador operacional, para nuestro caso elegimos el LM324, el cual contiene 4 Opams (Amplificadores Operacionales) en su interior.

Utilizaremos los Opams como un circuito comparador, en el pin+ conectaremos el fototransistor y en el pin- colocaremos un potenciómetro para la calibración de los sensores, los que se fijaran a un nivel de voltaje que depende del objeto a sensar. Las salidas del comparador variarán entre 0 y 5Vdc.

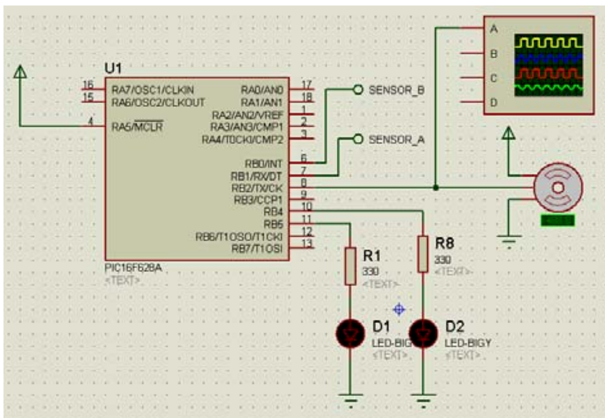
Fig. 3.4.1: Circuito acondicionador de un sensor infrarrojo.



3.4.2. Circuito del Microcontrolador

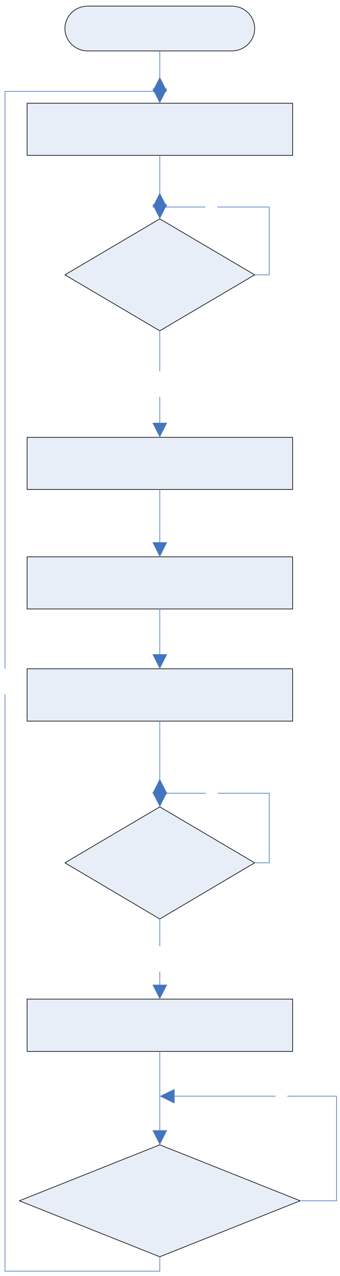
Para controlar la banda utilizamos el PIC16F628A, al tener muy pocas entradas y salidas que manejar se eligió este microcontrolador, también por su facilidad al momento de armar el circuito ya que este PIC posee un oscilador interno.

Fig. 3.4.2: Circuito controlador de la Banda transportadora



En el puerto RB0-RB1 se conectaron los sensores B y A respectivamente, en el puerto RB4-RB5 están conectados dos leds indicadores y en el puerto RB2 se conecta el servomotor que controla el movimiento de la banda, este motor fue previamente modificado para que pueda trabajar a 360 grados y su control se lo realiza por modulación de ancho de pulso PWM.

3.4.3. Diagrama de Flujo del microcontrolador



3.4.4. Código de programa del microcontrolador

```
program servomotor
dim in as byte

dim Vel as byte
dim cont1 as word

Main:

INTCON = 0
TRISA = 1
TRISB = 0
TRISB.0 = 1
TRISB.1 = 1
PORTA = 0
PORTB = 0
PORTA = 0


PORTB.4 = 1
Delay_ms(500)
PORTB.4 = 0
Delay_ms(500)

PORTB.4 = 1
Delay_ms(500)
PORTB.4 = 0
Delay_ms(500)

PORTB.4 = 1
Delay_ms(500)
PORTB.4 = 0
Delay_ms(500)


 While true
  if  (PORTB.0 = 0 ) then

      PORTB.4 = 1
      PORTB.2 = 1
      Delay_us(900)
```

```
    PORTB.2 = 0
    Delay_us(1100)


    if  (PORTB.1 = 1 ) then

       PORTB.5 = 1
       PORTB.2 = 0
       Delay_ms(2000)


            for cont1 = 0 to 700

            PORTB.2 = 1
            Delay_us(900)
            PORTB.2 = 0
            Delay_us(1100)

            next cont1

      else
      PORTB.5 = 0
    end if


  else

  PORTB.4 = 0
  Delay_ms(4000)

  PORTB.5 = 0


 end if
wend

 PORTB.2 = 0
 PORTB.4 = 1
 Delay_ms(2000)
 PORTB.4 = 0

End.
```

CAPÍTULO 4

4. SIMULACIÓN Y PRUEBAS

4.1. Configuración y Funcionamiento del equipo

El proyecto final está equipado con 5 sensores distribuidos en diferentes posiciones, los sensores del Lego NXT presentan señales digitales, en el caso del sensor de ultrasonido 8 bits de resolución y el sensor de Luz 10 bits; los sensores adicionales presentan valores de voltaje que varían entre 3.5 y 4.5 Vdc, en este capítulo se presentan los valores configurados para la toma de dediciones en el circuito controlador.

La construcción del brazo robótico se la hizo con Lego Mindstorms, utilizando 2 servomotores para su movimiento (Puerto A y B), con 2 grados de libertad, el tercer motor (Puerto C) se lo usó para el control de la garra, un sensor de luz (Puerto 3) para identificar la presencia del recipiente y un sensor de ultrasonido (Puerto 4) mide los niveles de producto dentro del recipiente.

Para el diseño de la banda transportadora utilizamos un circuito controlador mediante el PIC16F628A, el servomotor (Puerto RB2), 2 sensores de luz infrarrojo (puerto RB0 y RB1) y un sensor de tacto (Puerto RB3). La electrónica adicional fue diseñada según los requerimientos del proyecto, debido a que el kit de Lego Mindstorms solo trae tres sevomotores.

Fig. 4.1.1 Prueba de funcionamiento del nuevo Brazo robótico

Fig. 4.1.2 Vista lateral de la banda transportadora

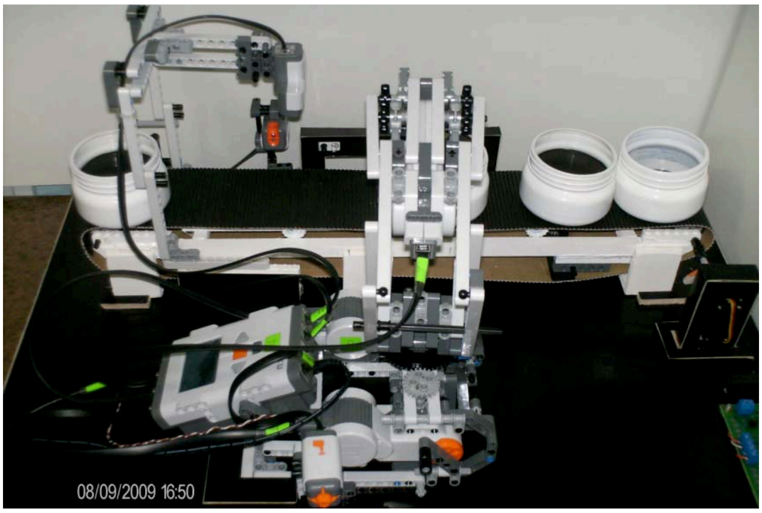


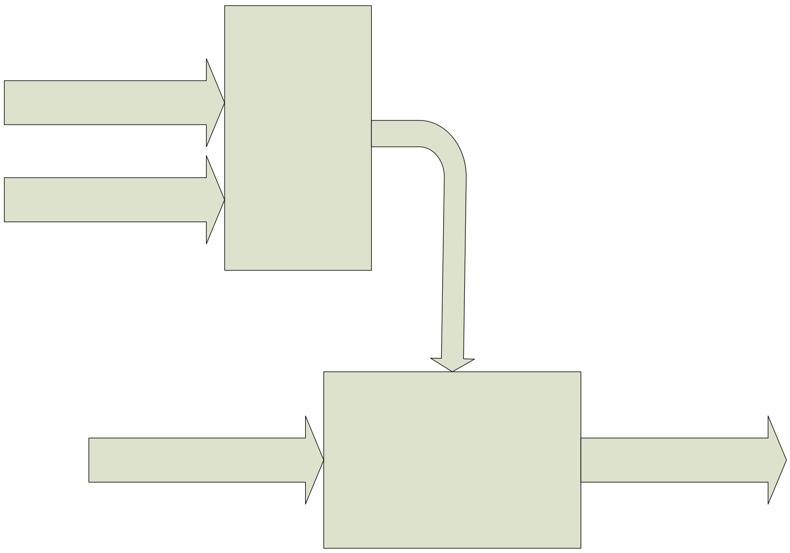
Fig. 4.1.3 Sensores de luz de en la banda transportadora

4.1.1. Tarjetas electrónicas (PCBs)

Para el diseño de las tarjetas se utilizó el software PROTEL 99 SE, la programación se la desarrolló en lenguaje C usando el compilador de mikroBASIC, para grabar el microcontrolador se utilizó el quemador de Microchip llamado PICkit 2, el cual me permite una grabación en el circuito sin necesidad de quitar el microcontrolador de la tarjeta controladora.

Básicamente el control de la banda contiene 3 tarjetas electrónicas independientes: Tarjeta Controladoras, Sensor Infrarrojo y fuente de Poder. Los elementos de entrada son 2 sensores infrarrojos y un sensor de tacto, mientras que a la salida del controlado se conecta un servomotor HITEC HS311 modificado para que trabaje a 360 grados.

Fig. 4.1.4 Esquema de conexión entre las tarjetas electrónicas



4.1.1.1. Circuito Controlador

El controlador está montado sobre la tarjeta universal Ctrl Pics V1.1, la cual me permite programar y probar los microcontroladores de microchip 16F84A, 16F628A, 16F876 y 16F88X. Para nuestro proyecto se usó el PIC16F628A, el la figura 4.1.5 se observan los componentes de la tarjeta, para controlar la banda solo se usaron los componentes:

Tabla 4.1.1 Listado de materiales tarjeta Ctrl. Pics V1.1

Nomenclatura	**Tipo**	**Descripción**
BT1	Bornera doble	Alimentación 5VDC
R1	Resistencia	33KOhm
U1	Zócalo 18 pines	Pic16F628A
J1	Jumper	Selector
R2, R3	Resistencia	220Ohm
D2, D3	Diodo Led	Puerto RB5, RB6
J5	Bornera 4 pines	Puerto RB0, RB1, RB2 y RB3
J8	3 Espadines	Servomotor RB2

Fig. 4.1.5 Componentes de la tarjeta universal Ctrl. Pics V1.1

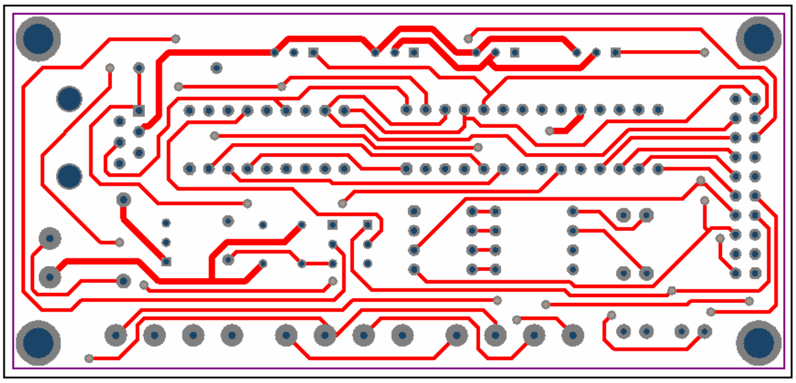


Fig. 4.1.6: Pista superior tarjeta universal Ctrl Pics V1.1

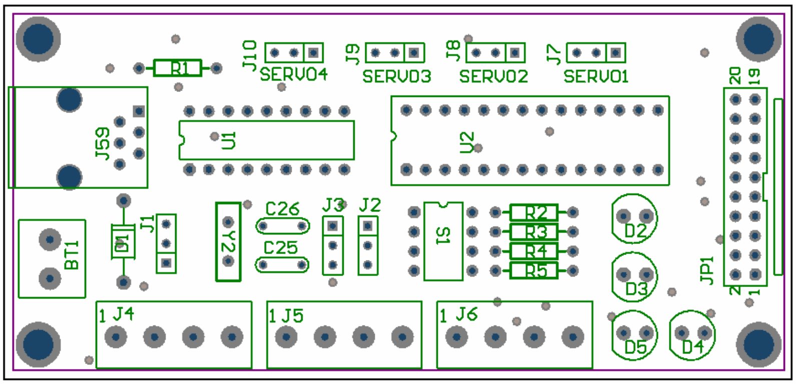
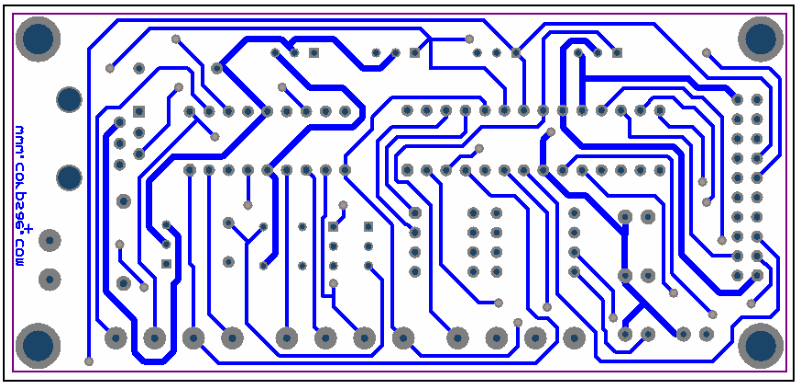


Fig. 4.1.7: Pista inferir tarjeta universal Ctrl Pics V1.1



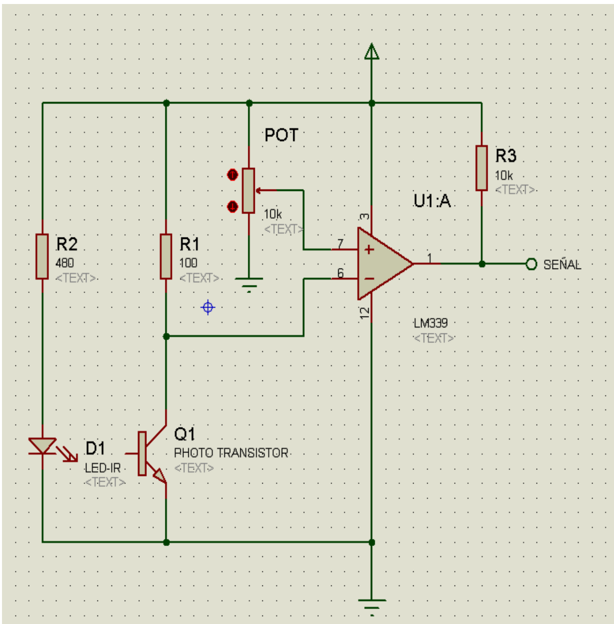
El esquemático de la tarjeta Ctrl Pics V1.1 se adjunta en los Anexos.

4.1.1.2. Circuito Sensor Infrarrojo

Esta tarjeta está diseñada para manejar hasta 4 sensores infrarrojos, en su interior contiene un amplificador operacional, el LM339N, este integrado trabaja como un comparador. La señal proveniente de Los fototransistores varía entre 2 y 4.7 voltios dependiendo del brillo de los objetos a medir, al comparador se conecta un potenciómetro por cada sensor, para calibrar el punto medio, en nuestro caso será 3.8 V.

El comparador muestra en su salida un voltaje de 0 o 5 V, si se encuentra un objeto cerca presenta 5V y caso contrario el voltaje será 0 V.

Fig. 4.1.8: Circuito del Sensor Infrarrojo



La banda transportadora contiene 2 sensores infrarrojos, el primero detiene el motor para que el sensor ultrasónico del brazo mida la el nivel de producto dentro de los envases, mientras que el segundo sensor detiene la banda para que el brazo tome los recipientes y los seleccione según el nivel de producto.

La tarjeta utiliza da es Sensor Luz V4.0, en la que solo utilizaremos una parte de sus componentes para cumplir la tarea propuesta, el listado de elementos se adjunta a continuación:

Tabla 4.1.2 Listado de materiales tarjeta Sensor Luz V4.0

Nomenclatura	**Tipo**	**Descripción**
R27, R28, R31 y R30	Resistencia	10 KOhm
R23 y R22	Resistencia	470 Ohm
R34 y R35	Potenciómetro	10 KOhm
J4	Sip-4	Salidas del Opam
J3	Bornera doble	Alimentación 5Vdc
D2 y D3	Bornera doble	Diodos Infrarrojos
Q6 y Q7	Bornera doble	Fototransistores
U2	Zócalo 14 pines	LM339N

Fig. 4.1.9 Componentes de la tarjeta Sensor Luz V4.0

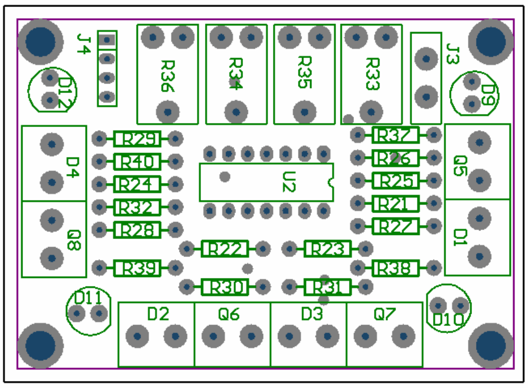


Fig. 4.1.10: Pista superior de la tarjeta Sensor Luz V4.0

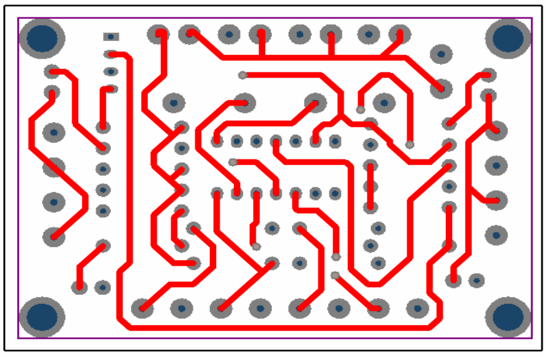
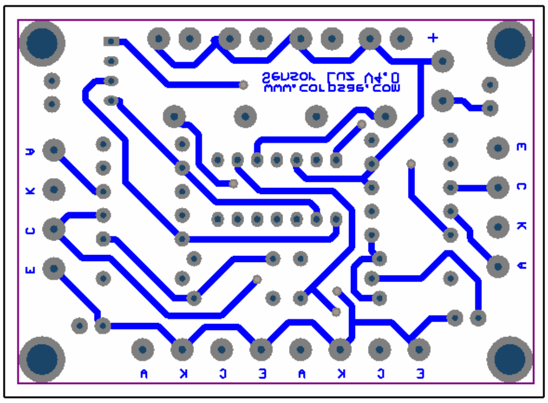


Fig. 4.1.11: Pista inferior de la tarjeta Sensor Luz V4.0



El esquemático de la tarjeta Sensor Luz V4.0se adjunta en los Anexos.

4.1.1.3. Circuito de la Fuente de Poder.

Para la alimentación de las tarjetas anteriormente mencionadas, utilizaremos un circuito regulador de voltaje, el que se puede alimentar usando un adaptador de 120Vac/7.5-12Vdc. Esta fuente contiene un regulador L7805, el cual acepta entradas no reguladas de 8 a 18 voltios y produce una salida regulada de +5Vdc, con una exactitud del 5% (0.25 volt).

Contiene un circuito limitador de corriente y una protección contra sobre-calentamiento, para alimentar esta tarjeta se puede utilizar un transformador simple con dos salidas en el secundario, aquí necesitamos un rectificador de onda completa lo cual se consigue con 4 diodos, todas las fuentes usan en la salida un condensador de 0.01µf para desviar los ruidos parásitos a tierra.

Fig. 4.1.12 Circuito de una fuente de poder con regulador de voltaje

La tarjeta electrónica Fuente V4 contiene los siguientes componentes:

Tabla 4.1.3 Listado de materiales tarjeta Fuente V4

Nomenclatura	Tipo	Descripción
BT2	Regulador de Voltaje	L7805
J4	Jack Adaptador	
C6	Capacitor electrolítico	470uF
C5	Capacitor electrolítico	100uF
S1	Interruptor y Fusible	1A
C1 y C2	Capacitor cerámico	
D3	Diodo rectificador	1N4007
R1	Resistencia	470Ohm
J1 y J2	Bornera doble	Voltaje Regulado
D1	Diodo Led	

Fig. 4.1.13 Componentes de la tarjeta Fuente V4

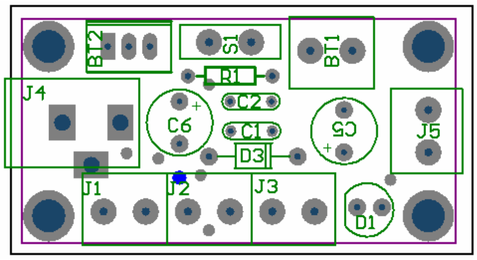


Fig. 4.1.14: Pista superior de la tarjeta Fuente V4

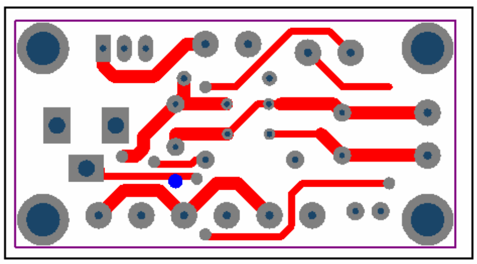
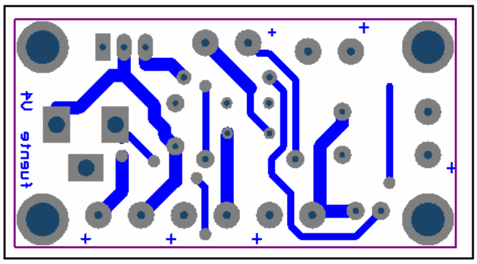


Fig. 4.1.15: Pista inferior de la tarjeta Fuente V4



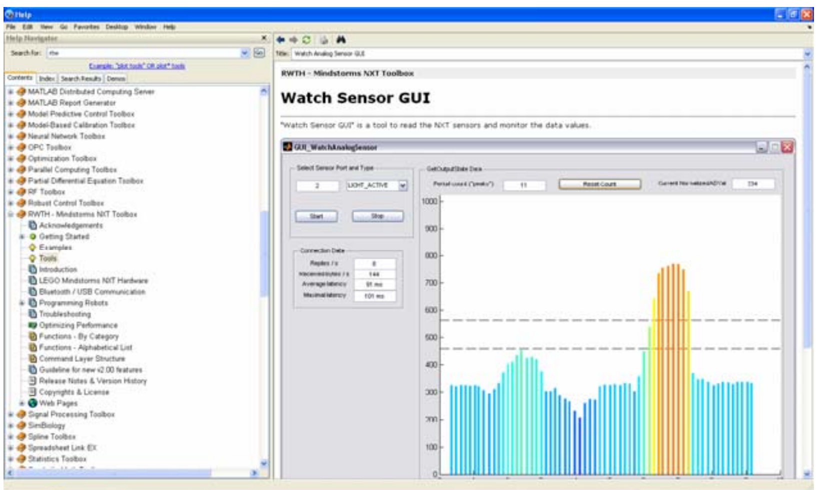
4.2. Seteo y Configuración de los sensores.

Para la calibración de los sensores de luz y ultrasónico se usó la herramienta Watch Sensor GUI, la cual permite fijar los parámetros para el brazo robótico.

4.2.1. Calibración de Sensores del NXT

Dentro del toolbox de Matlab RWTH - Mindstorms NXT se encuentra la herramienta Watch Sensor GUI, esto nos permite una visualización gráfica de los valores analógicos medidos por los sensores de Lego NXT. Podemos ingresar a esta herramienta mediante la ayuda de Matlab.

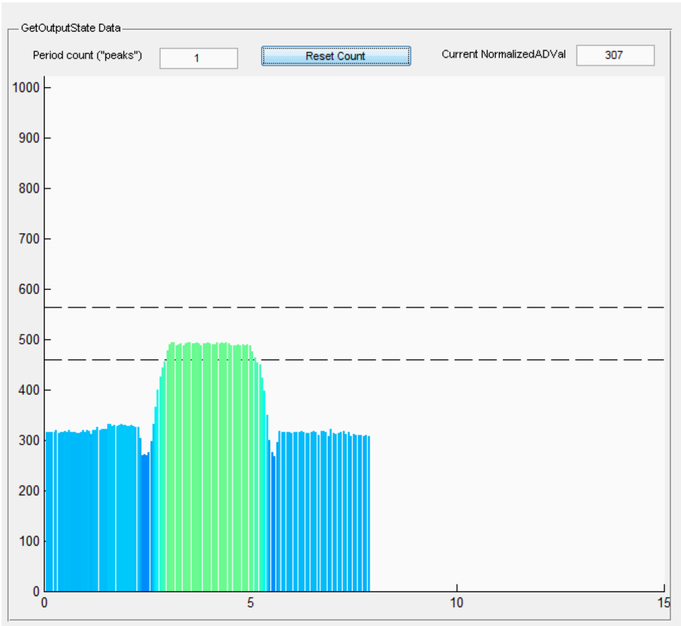
Fig. 4.2.1 Help de Matlab en la librería RWTH - Mindstorms NXT



El proyecto contiene 2 sensores, uno infrarrojo y otro de Ultrasonido. El sensor infrarrojo se lo usa para notar la presencia de un objeto en la banda, en este punto se medirán los niveles de producto dentro de los envases usando el sensor ultrasónico.

El sensor de luz tiene una resolución de 10 bits, es decir presenta valores de 0 a 1124, se lo ha conectado en el puerto 3 del NXT. En la figura 4.2.2 se presentan valores inferiores a 350 bits, todo lo que está por debajo de este rango significa que no hay presencia de algún objeto frente al sensor. Todos los valores que están sobre los 420 bits representan presencia de un objeto claro en frente del sensor.

Fig. 4.2.2 Rango de Valores de sensor de luz



El sensor Ultrasónico tiene 8 bits de resolución, es decir que presenta valores entre 0 a 256, para el proyecto utilizaremos envases con tres niveles de producto, vacío, medio y lleno respectivamente. El sensor está clocado a una altura de 9.5cm. de la banda transportadora.

Tabla 4.2.1: Valores medidos en el sensor de Ultrasonido

Niveles	Valor Medido (Bits)
Cero	11
Vacío	9
Medio	8
Lleno	7

La superficie de color blanco en las graficas mostradas representan los valores medidos de los sensores en diferentes condiciones de nivel.

Fig. 4.2.3 Rango de Valores, Sensor Ultrasónico en nivel de producto Vacío

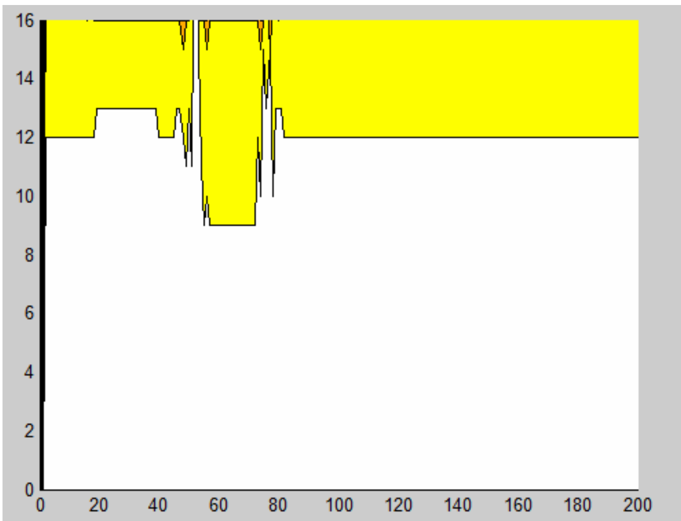


Fig. 4.2.4 Rango de Valores, Sensor Ultrasónico en nivel de producto Intermedio

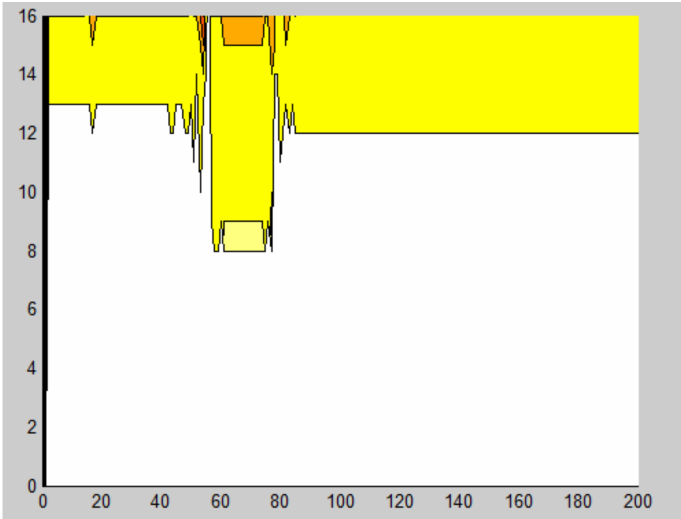
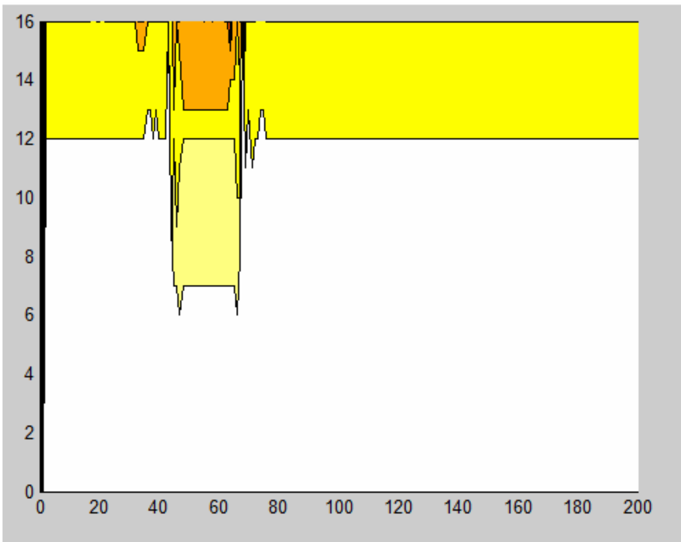


Fig. 4.2.5 Rango de Valores, Sensor Ultrasónico en nivel de producto Lleno



4.2.2. Calibración de Sensores de la banda transportadora

Para el control de banda transportadora se utilizaron 2 diodos emisores de luz infrarroja y 2 fototransistores, con ello se diseño el circuito de sensores de luz, mencionado en la sección 3.4.1. Al polarizar los fototransistores con una resistencia de 10KOhm se obtuvieron los valores presentados en la tabla 4.2.2, al incidir sobre ellos una la luz infrarroja de los diodos emisores.

Tabla 4.2.2: Valores medidos en el sensor de Ultrasonido

Sensor	Sin presencia de Objeto (Vdc)	Con presencia de Objeto (Vdc)
A	4.86	3.67
B	4.88	3.71

Como utilizamos circuitos comparadores para obtener señales digitales de 0 o 5V a la salida de cada sensor, posicionamos al potenciómetro de calibración en 4.5 Vdc, cualquier valor de voltaje inferior al calibrado activará al comparador mostrando señales digitales en alto.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1.- El desarrollo de este proyecto es un Brazo Robótico con una Banda Transportadora, aplicación del campo de la robótica con el cual aportamos una solución de diseño para mejorar la calidad de una línea de producción en una industria envasadora de bebidas, a su vez mejorar el orden de almacenamiento de los envases que tiene el nivel correcto y rechazar los envases que tienen un nivel fuera del rango.

2.- Con el principio del prototipo inicial del proyecto, se determinó la aplicación de los sensores con las siguientes funciones: el sensor de luz detecta la presencia del recipiente, el sensor de ultrasonido mide el nivel del recipiente y el sensor de tacto detiene todo el proceso del Brazo Robótico una vez terminada la secuencia del mismo.

3.- Se usó Matlab para desarrollar la programación de la secuencia del Brazo Robótico y de igual manera para obtener la adquisición de datos del sensor ultrasónico en tiempo real. Con el comando xlswrite de Matlab exportamos los datos medidos del sensor ultrasónico durante toda la secuencia del Brazo a una hoja de Excel.

4.- En la implementación del proyecto se utilizó los tres servomotores que trae Lego Mindstorms NXT para el Brazo Robótico, por lo que se adicionó el servomotor HITEC HS311 para controlar la Banda Transportadora donde circularán los recipientes. Se usó este servomotor porque tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación y de mantenerse estable en dicha posición, la cual nos facilita en los instantes que se detiene la Banda para que el sensor de ultrasonido mida el nivel del recipiente.

5.- Para la alimentación de la tarjeta de control de la Banda Trasportadora, se diseñó una fuente independiente de voltaje de 5 VDC, utilizando el regulador 7805 para tener una salida constante de 5 V. Con este circuito también se consiguió tener la corriente necesaria de 180 mA para mover el servomotor debido que la alimentación que proporciona Lego Mindstorms NXT no es la requerida.

Recomendaciones

1.- Programar la secuencia del brazo robótico en Matlab, utilizando los comandos de rotación para mover los servomotores, los cuales dan un valor exacto de giro para el servo, ya que si es realizado con comandos de tiempo presenta margen de error en cuanto al posicionamiento del brazo.

2.- Para calibrar el sensor de luz, tomar en cuenta que ésta es afectada por la luz del ambiente ya que el sensor del NXT funciona con luz visible roja, por lo que puede dar error al momento de la calibración, por tanto se recomienda trabajar con sensores de luz infrarroja ya que estos no son afectados por la luz del ambiente.

3.- Con los conocimientos adquiridos en el desarrollo del proyecto se puede motivar el desarrollo y la investigación de las aplicaciones del Lego NXT para la implementación de futuros proyectos que integren nuevas tendencias tecnológicas para encontrar soluciones a las mismas.

ANEXOS

ANEXO A:

Instrucciones para instalar Embedded Coder Robot NXT

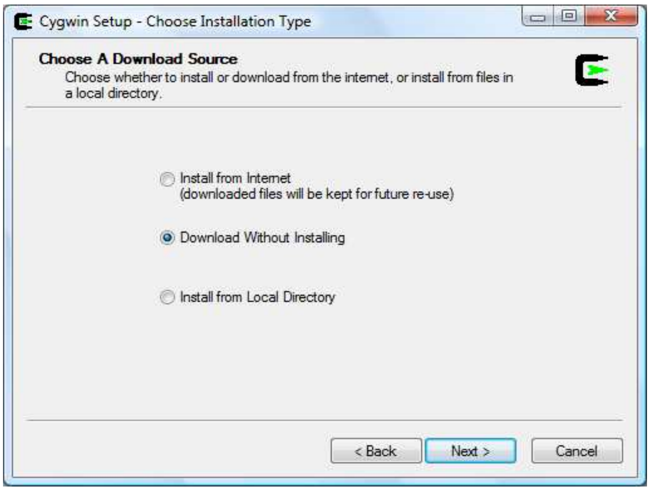
Pre-requisitos:

a) Instalar Matlab R2008 en la unidad C, de la siguiente manera: C:\MATLAB
b) MATLAB debe tener instalado Simulink, Real-Time Workshop Real-Time Workshop Embedded Coder.

Instalar las siguientes Herramientas auxiliares:

1. **Instalar Cygwin:**

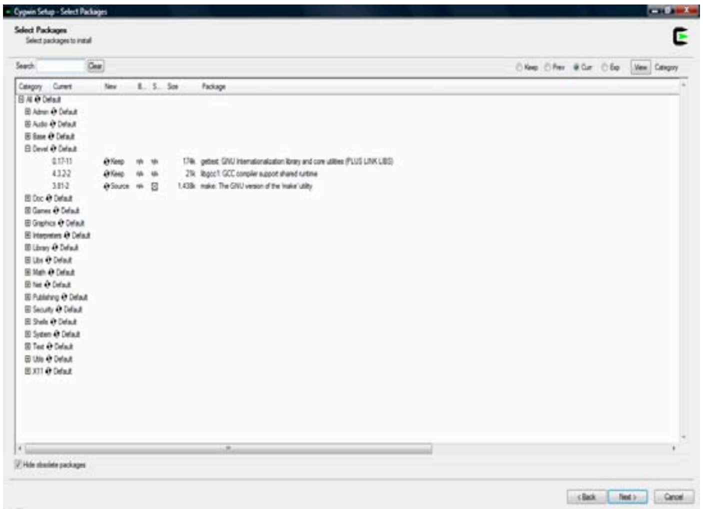
a) Descargar setup.exe versión 1.7 de http://www.cygwin.com/



b) Ejecutamos el setup.exe descargado
c) Descargamos los paths sin instalarlos desde internet
d) Seleccionamos la ruta donde vamos a guardar los paths
e) Seleccionamos la url: http//cygwin elite-system.org (es mas rápido)
f) Ahora instalamos los paths que hemos descargados en la carpeta : http%3ª%2f%2fcygwin.elite-system.org%2f

g) Seleccionamos la opción siguiente para instalar el software cygwin

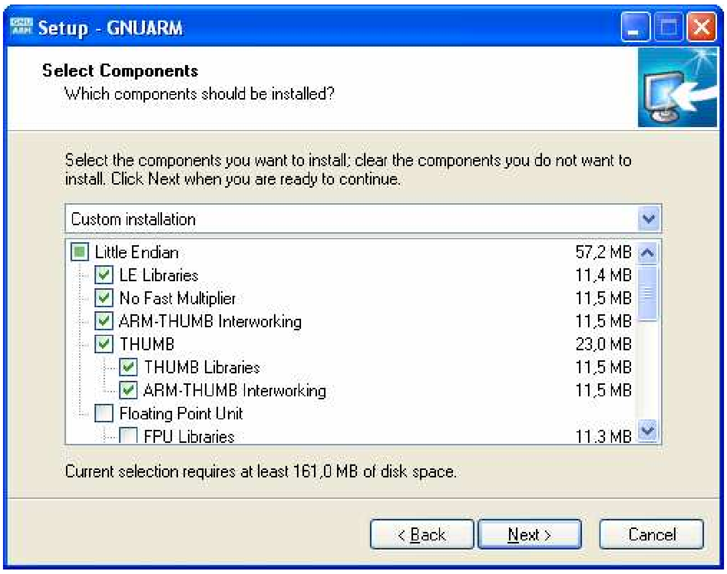
h) Escogemos make 3.81-2 que se encuentra dentro de Devel Default



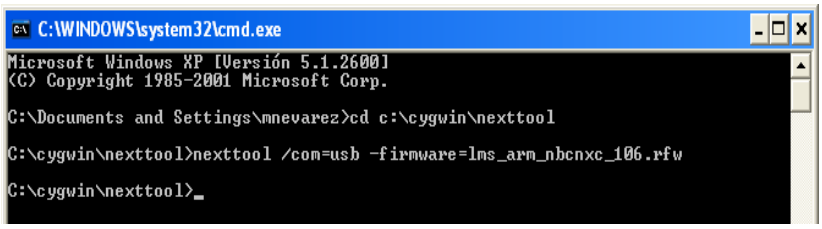
i) Instalado cygwin lo ejecutamos y realizamos las siguientes pruebas para comprobar que hemos instalado perfectamente. Para lo cual le preguntamos cygwin donde tiene el programa "make":
teclea "which make", te ha de responder algo así como "/usr/bin/make"
$ which make
/usr/bin/make (Si no da una localización, esta mal instalado).

2. Instalar GNU ARM:

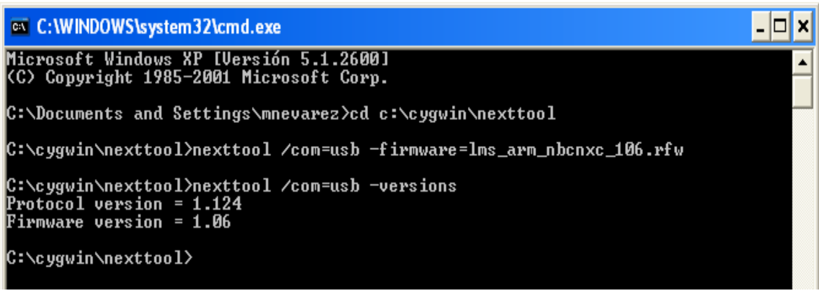
a) Instalarlo dentro de la carpeta de Cygwin (C :\Cygwin\GNUARM)
b) Desmarque la instalación DLL de Cygwin porque ya fue instalado anteriormente.



c) Instalar Nexttool dentro de la carpeta de Cygwin (C :\Cygwin\GNUARM)
d) Descomprimir el archivo lms_arm_jch.zip
e) Copiar el firmware dentro de la carpeta de nexttool
f) Actualizar el firmware utilizando el cmd (comand promt)



g) Verficar si se instaló



3. Instalar NXT USB Driver

a) Si se tiene instalado el Software Mindstorm , ya no es necesario instalar.
b) De lo contrario, descárguelo de la página http://mindstorms.lego.com/Support/Updates/
c) Instale el driver

4. Instalar EcRobot

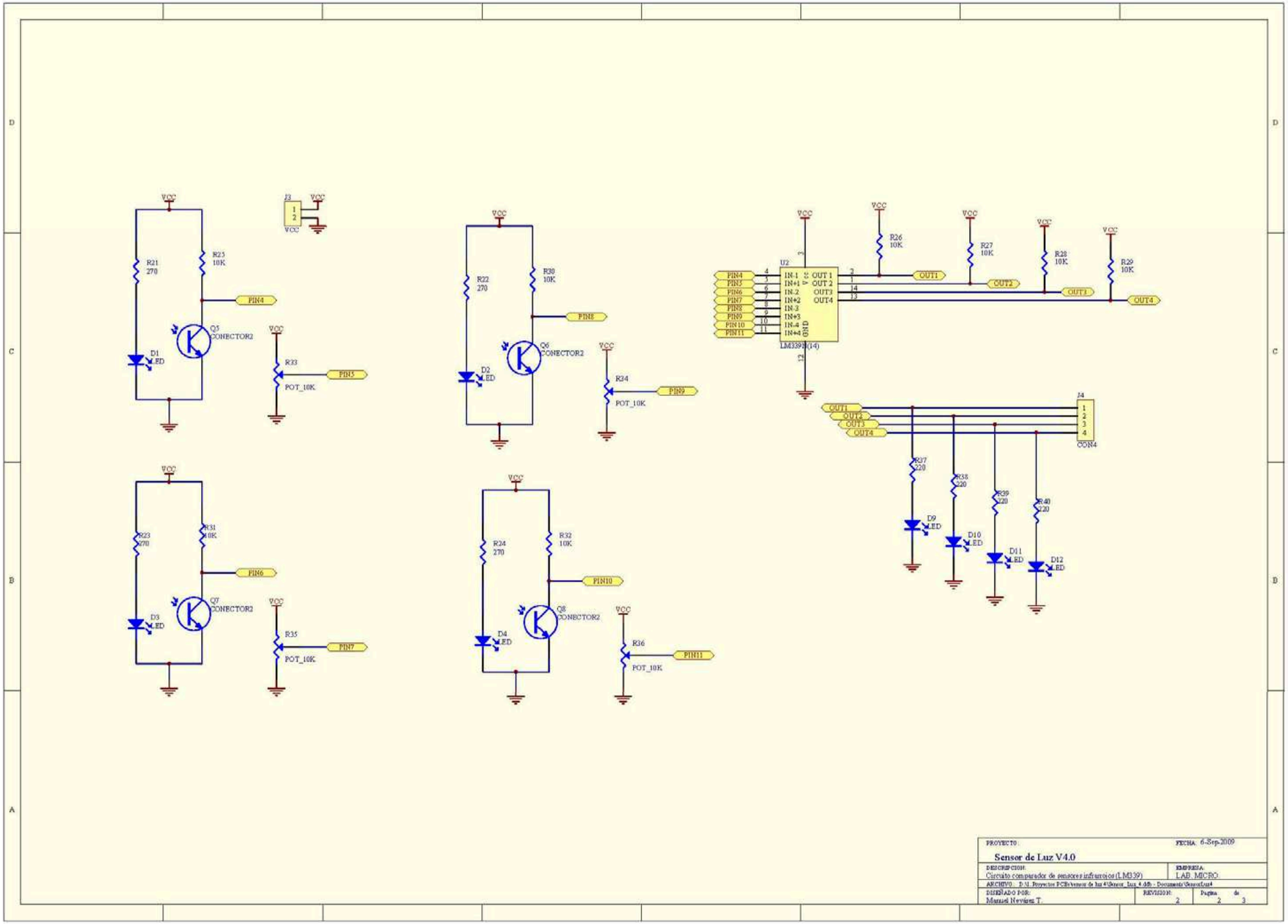
a) Descargue y descomprima EcRobot
b) Copie el archivo en el siguiente directorio: C:\MATLAB\ecrobotNXT

5. Instalar nxtOSEK

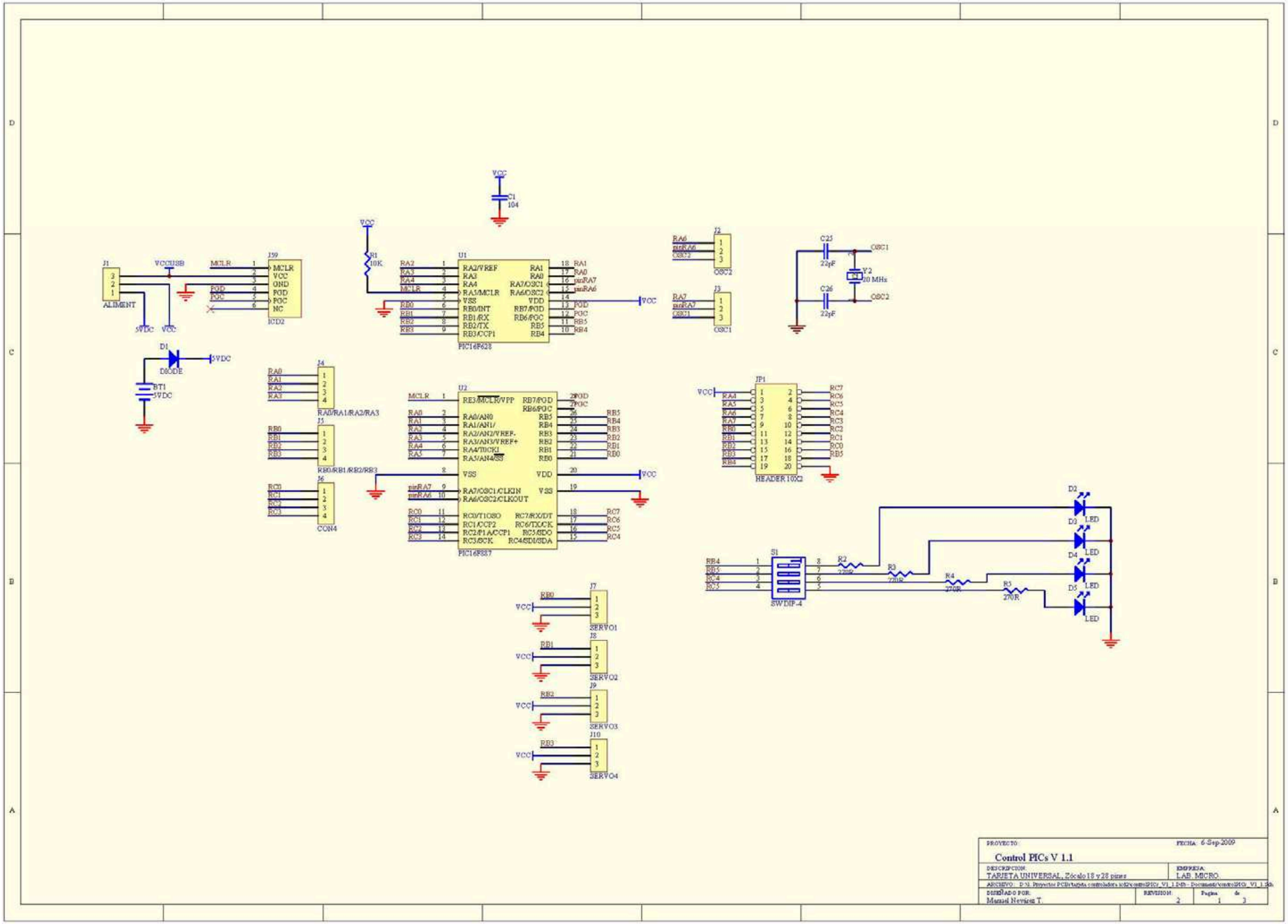
a) Descargue y descomprima nxtOSEK
b) Copie el archivo en el siguiente directorio: C:\MATLAB\ecrobotNXT\environment\nxtOSEK
c) Entre al siguiente directorio: F:\MATLAB\ecrobotNXT
d) Abra el siguiente archivo: ecrobotnxtsetup.m y lo ejecutamos.

ANEXO B:

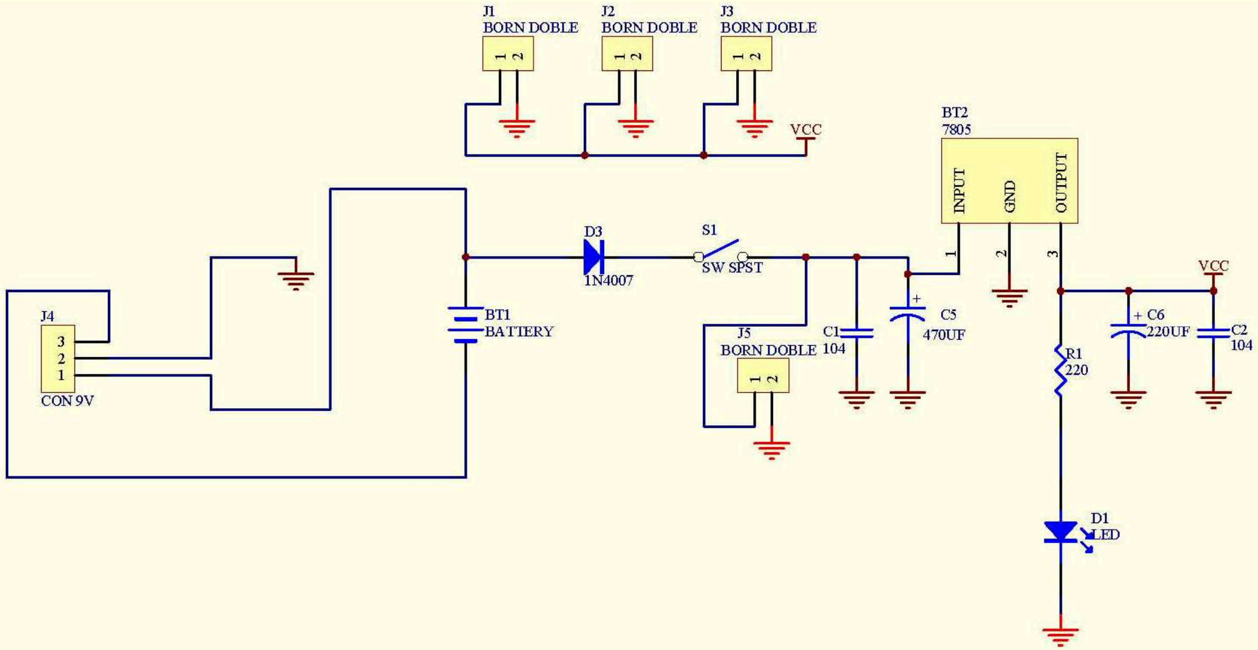
Circuitos esquemáticos de las tarjetas electrónicas utilizadas



VCC
R21
270
R25
10K
PIN4
Q5
CONECTOR2
D1
LED
R33
POT_10K
PIN5
J3
R30
10K
R22
270
PIN8
Q6
D2
R34
PIN9
R23
R31
PIN6
Q7
D3
R35
PIN7
R24
R32
PIN10
Q8
D4
R36
PIN11
U2
LM339N(14)
IN-1
IN+1
IN-2
IN+2
IN-3
IN+3
IN-4
IN+4
OUT 1
OUT 2
OUT3
OUT4
GND
R26
R27
R28
R29
OUT1
OUT2
OUT3
OUT4
J4
CON4
R37
R38
R39
R40
220
D9
D10
D11
D12
PROYECTO:
Sensor de Luz V4.0
FECHA: 6-Sep-2009
DESCRIPCION:
Circuito comparador de sensores infrarrojos (LM339)
EMPRESA:
LAB. MICRO.
DISEÑADO POR:
Manuel Nevárez T.
REVISION: 2
Pagina 2 de 3



J1
ALIMENT
VCCUSB
MCLR
PGD
PGC
J39
MCLR
VCC
GND
PGD
PGC
NC
ICD2
5VDC
VCC
D1
DIODE
BT1
5VDC
VCC
C1
104
R1
10K
U1
RA2/VREF
RA3
RA4
RA5/MCLR
VSS
RB0/INT
RB1/RX
RB2/TX
RB3/CCP1
RA1
RA0
RA7/OSC1
RA6/OSC2
VDD
RB7/PGD
RB6/PGC
RB5
RB4
PIC16F628
J2
OSC2
J3
OSC1
C25
22pF
C26
22pF
Y2
20 MHz
J4
RA0/RA1/RA2/RA3
J5
RB0/RB1/RB2/RB3
J6
CON4
U2
RE3/MCLR/VPP
RA0/AN0
RA1/AN1/
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS
VSS
RA7/OSC1/CLKIN
RA6/OSC2/CLKOUT
RC0/T1OSO
RC1/CCP2
RC2/P1A/CCP1
RC3/SCK
RB7/PGD
RB6/PGC
VDD
RC7/RX/DT
RC6/TX/CK
RC5/SDO
RC4/SDI/SDA
PIC16F887
JP1
HEADER 10X2
S1
SWDIP-4
R2
R3
R4
270R
R5
270R
D2
D3
D4
D5
LED
J7
SERVO1
J8
SERVO2
J9
SERVO3
J10
SERVO4
PROYECTO:
Control PICs V 1.1
FECHA: 6-Sep-2009
DESCRIPCION:
TARJETA UNIVERSAL, Zócalo 18 y 28 pines
EMPRESA:
LAB. MICRO.
DISEÑADO POR:
Manuel Neváres T.
REVISION:
2
Pagina 1 de 3



PROYECTO: Fuente V4	FECHA: 8-Sep-2009
DESCRIPCION: Fuente regulada 5Vdc	EMPRESA: LAB. MICRO.
ARCHIVO: D:\1. Proyectos PCBs\FUENTE REGULADA\fuente_3.ddb - Documents\fuente_3.Sch	
DISEÑADO POR: Manuel Nevárez T.	REVISION: 2 — Pagina 3 de 3

BIBLIOGRAFÍAS

LIBROS

1. McCOMB GORDON, The Robot Builder's Bonanza 2º Edición, McGrill-Hill, Estados Unidos, 2001.

2. MANUEL GIL RODRIGUEZ, Introducción rápida a Matlab y Simulink, Ediciones Díaz Santos, Madrid , 2003.

3. MYKE PREDKO, Programming Robot Controllers, McGrill-Hill, Estados Unidos, 2003.

4. ANIBAL OLLERO BATURONE, ROBÓTICA Manipuladores y Robots móviles, Marcacombo Boixareu Editores, 2001

SITIOS WEBS

5. Extreme NXT "Extending the LEGO MINDSTORMS NXT to the Next Level", página html

   http://www.apress.com

6. PIC Microcontrollers - Programming in C Mikroelektronika, página html

   http://www.mikroe.com/en/books/pic-books/mikroc/

7. Embedded Coder Robot NXT Instruction Manual, página html
   http://www.pages.drexel.edu/~dml46/Tutorials/BalancingBot/files/Embedded_Coder_Robot_NXT_Instruction_En.pdf

8. MICROCHIP, Hoja de Datos PIC16F628A, página html
http://www.datasheetcatalog.net/es/

9. How to upload a nxtOSEK program to the NXT , página html

http://lejos-osek.sourceforge.net/howtoupload.htm

10. Cygwin Information and Installation, página html
http://www.cygwin.com/

11.