ROBOT MÓVIL CON APLICACIONES METODOLOGICAS PARA EL ESTUDIO DE LA ROBÓTICA MEDIANTE LEGO NXT

ALFREDO SANCHEZ SILVERA

CARLOS ROLDAN ANGULO PÉREZ

MARJORY JULITZA GARCÍA FUENTES

CORPORACION UNIVERSITARIA DE LA COSTA.

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA ELECTRONICA

BARRANQUILLA

2010

ROBOT MÓVIL CON APLICACIONES METODOLOGICAS PARA EL ESTUDIO DE LA ROBÓTICA MEDIANTE LEGO NXT

ALFREDO SANCHEZ SILVERA

CARLOS ROLDAN ANGULO PÉREZ

MARJORY JULITZA GARCÍA FUENTES

Proyecto de grado presentado como requisito para optar al título de

Ingeniero Electrónico

DIRIGIDO POR:

ESP. ING. RUBÉN D. SÁNCHEZ DAMS

CORPORACION UNIVERSITARIA DE LA COSTA.

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA ELECTRONICA

BARRANQUILLA

2010

Nota de aceptación.

_____

Presidente del jurado

_____

Jurado

_____

Jurado

Barranquilla, 24 de Marzo de 2010

Agradecimientos.

Al Ing. Rubén Sánchez nuestro director, quien tuvo mucha paciencia y dedicación, por apoyarnos, orientarnos durante todo el proyecto, aguantar las llamadas a deshoras, trasnocharse con nosotros, y creer en el proyecto desde el principio.

A Juan Carlos Gutiérrez por su colaboración en la grabación, edición y captura del video de las aplicaciones.

A mi Primo Ulises por ayudarnos en la logística y envío de piezas desde Estados Unidos.

A Andrés Chamorro, Julio Chamorro, Juan Carlos Gutiérrez, por toda la colaboración, por dejar a nuestra disposición su PC.

Al Ing. Robinson Guzmán, que gracias a su proyecto de aula propuesto, es que nace la inquietud e iniciativa por ahondar y desarrollar el proyecto.

A la institución y al Ing. Jaime Vélez por la aceptación del proyecto.

Dedicatoria.

A mi madre quien con esfuerzo y dedicación me ha acompañado día a día en este proceso. Por haber querido para mí mejores horizontes y por haberme inculcado el deseo por superar los obstáculos que la vida me presenta.

A mi tío por ser apoyo incondicional. Por su ejemplo de buen profesional y consagración a su labor.

A mis hermanos y mi abuela por sus consejos, alegría, apoyo y buena energía.

A Enith por trasnocharse conmigo, por darme impulso para alcanzar mis metas y por su grandiosa compañía.

A Tatiana, amiga incondicional, por hacerme ver la vida de otra forma y mostrarme que vale la pena esforzarse por alcanzar los sueños en la vida.

A Carlos y Alfredo, quienes ante la adversidad, se esforzaron por sacar adelante este proyecto.

Dedico este logro a mi madre, quien ha consagrado toda su vida y esfuerzos por hacer de sus hijos personas de bien y por haberme acompañado a alcanzar cada meta que me propongo en la vida.

M.J.

"La religión explica los miedos del hombre, la ciencia descubre las verdades de la naturaleza, la ciencia es para valientes"

Anónimo

Luego de una infinita lista de infortunios y vientos en contra que en vez de debilitar el deseo de continuar, sirvieron de impulso en la culminación de esta etapa del comienzo del fin del desconocimiento consciente del conocimiento, me siento

inmensamente satisfecho de haber compartido con muchas personas que aportaron ideas y colaboraron tanto físico, como mentalmente. Ayudaron directa o indirectamente para que se llevara a la realidad esta idea.

A mi mama que siempre ha estado ahí, desde siempre y para siempre, y que gracias a sus esfuerzos he completado este primer escalón en los millares de esta escalera de sueños.

A mi tía luz marina por ser un apoyo mental y físico incondicional.

A mis tías Maribel Pérez, Consuelo Pérez por apoyarme en muchos aspectos desde el inicio de mi carrera.

A mis abuelos por siempre haberme brindado un hogar donde llevar a cabo este y muchos otros sueños.

A mis amigos cercanos que siempre brindaron su mano para colaborarme en múltiples aspectos:

Juan Carlos Gutiérrez, Andrés David chamorro, Oriana Onofri, Augusto Pardo, Alejandro Pardo.

A mis compañeros de investigación, marjory julitza García fuentes, Alfredo Sánchez, por haber creído y haber hecho parte de esta idea.

A Edwin garrido por habernos abierto espacios donde mostrar nuestro proyecto.

A la universidad por permitirnos desarrollar esta investigación, y por abrirnos campos donde presentarla.

Y a todos aquellos que olvidé mencionar en la prisa y el nervio de la entrega, ¡Muchas Gracias!

Carlos Angulo.

A DIOS porque todos nuestros planes en sus manos dan frutos.

A mis padres Alfredo y Flor María por sus sacrificios, y esfuerzos entregados para formarme como profesional.

A mi familia quienes me enseñaron que con dedicación y unidad todo es posible.

A Dios por la fortaleza, paciencia y prudencia recibida durante el desarrollo de este proyecto y la realización de mi carrera como Ingeniero Electrónico. Gracias Señor por tu guía y compañía.

A mis padres; Alfredo y Flor María por el apoyo y el ánimo, recibido en los momentos más adecuados.

A mis hermanos; Norley, Alex, Yelesis, Wilton y Carlos. Por los concejos, porque siempre estuvieron presentes para ayudarme.

A toda mi familia; porque me hicieron ver que nunca estaba solo.

A todos mis amigos y todas aquellas personas que hicieron posible alcanzar esta meta.

MUCHAS GRACIAS a todos sin ustedes no abría podido cumplir mi meta.

Alfredo Sánchez Silvera.

Contenido.

TABLAS.

LISTA DE FIGURAS.

LISTA DE ANEXOS.

Glosario.

Accionamientos eléctricos: La estructura es simple en comparación con la de los accionamientos hidráulicos y neumáticos, ya que sólo se requieren de energía eléctrica como fuente de poder. Como se utilizan cables eléctricos para transmitir electricidad y las señales, es altamente versátil y prácticamente no hay restricciones respecto a la distancia entre la fuente de poder y el accionamiento.

Accionamientos o efector finales: Son dispositivos que convierten una energía eléctrica (generalmente) en una acción sobre el entorno, sea de luz, mecánica, hidráulica, neumática, etc. En robótica son utilizados para dar movimiento al robot.

Accionamientos hidráulicos: Son los que tienes como fuente de energía los fluidos, como aceites minerales. Estos sirven para realizar grandes esfuerzos, como en el caso de las retro-excavadoras.

Accionamientos neumáticos: Son los que tienes como fuente de energía el aire a presión. Estos son utilizados para acciones que requieren precisión.

Acelerómetro: cualquier instrumento destinado a medir aceleraciones.

Androide: es un robot u organismo sintético diseñado para parecer y actuar como un humano.

API: Interfaz de Programación de Aplicaciones: Son una biblioteca con clases y métodos para realizar programación orientada a objeto.

Brazo antropomórfico: brazo mecánico con características humanoides.

Clase: Conceptualización lógica utilizada en el software conformada por una agrupación de datos (variables o campos) y de funciones (métodos) que operan sobre esos datos.

Compresibilidad: cualidad de compresible, es decir, que se puede comprimir.

Defusificación: Parte del control difuso donde se convierten los valores difusos a valores numéricos.

Encoders: Sensor para conocer la posición angular; los encoders normalmente se encuentra acoplados a motores.

Exoesqueleto robot: Es una estructura metálica que facilita la movilidad del robot

Firmware: Es un bloque de instrucciones de programa para propósitos específicos, grabado en una memoria de tipo no volátil (ROM, EEPROM, flash), que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.

Fuzzyficación: Es La primera etapa del control difuso, consiste en convertir los valores de entrada obtenidos por el sensor en conjuntos difusos.

Giroscopios: es un dispositivo mecánico formado esencialmente por un cuerpo con simetría de rotación que gira alrededor de su eje de simetría.

Hidráulica: es una rama de la física y la ingeniería que se encarga del estudio de las propiedades mecánicas de los fluidos.

IDE: Entorno de Desarrollo Integrado, es un programa informático compuesto por un conjunto de herramientas integradas que facilitan la programación.

Implicación: Es La segunda etapa del control difuso donde se evalúan las reglas de inferencia.

Interfaz: Es el medio por el cual un usuario puede comunicarse con un dispositivo electrónico.

Java: Lenguaje de programación de alto nivel orientado a objetos.

LEGO: Es una empresa de juguetes danesa reconocida principalmente por sus bloques de plástico interconectadles.

leJOS NXJ: Es un firmware para lego NXT, el cual permite programar el robot en java. (leJOS, Java for Lego Mindstorms, 2009).

Locomoción: Traslación de un lugar a otro.

Lógica difusa: La lógica difusa o lógica borrosa es un tipo de control que se basa en lo relativo de lo observado. Este tipo de lógica toma valores de procesos los cuales están contextualizados y referidos entre sí. Así, por ejemplo, una persona que mida 2 metros es claramente una persona alta, si previamente se ha tomado el valor de persona baja y se ha establecido en 1 metro. Ambos valores están contextualizados a personas y referidos a una medida métrica lineal.

Maniobrabilidad: capacidad de maniobrar.

Morfología: es la disciplina que estudia la generación y las propiedades de la forma.

Método: Son acciones, funciones o procedimientos que realiza el programa y operan sobre el estado interno del objeto, y definiendo la interacción del objeto con el mundo exterior.

Navegación inercial: Es un sistema de navegación que sirve para determinar la posición relativa de un robot, utilizando giroscopios y acelerómetros para medir la tasa de rotación y aceleración.

Neumática: es la tecnología que emplea el aire comprimido como modo de transmisión de la energía necesaria para mover y hacer funcionar mecanismos.

Objeto: Instancia de una clase

Odometría: es el estudio de la estimación de la posición de vehículos con ruedas durante la navegación.

P.O.O: Programación Orientada a Objetos (OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar

aplicaciones y programas de sistemas de computo. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento.

PID: Controlador Proporcional-Integral-Derivativo (controlador PID) es un mecanismo genérico de control de lazo realimentado ampliamente usado en los sistemas de control industrial. Un controlador PID corrige el error entre la variable de proceso medida y la consigna, realizando una acción correctiva sobre la salida, la cual será proporcional al error, con lo que se puede mantener las condiciones deseadas del proceso.

Plugins o complemento de software: Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

Puerto: Interfaz para enviar y recibir información.

PWM: Modulación por ancho de pulsos, consiste en modificar el ciclo de una señal periódica.

Rango: Amplitud de la variación de un fenómeno entre un límite menor y uno mayor claramente especificados.

Redes neuronales: Sistema de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas en una red que colabora para producir un estímulo de salida.

Redundancia: es un principio de diseño por el cual diversos sistemas pueden hacer la misma función simultáneamente, garantizando en caso accidental de uno de ellos los otros sistemas aún protejan el sistema.

Resolvers: Encoders ópticos.

Robot: Maquina capaz de realizar labores a través de una programación.

Robot Industrial: manipulador programable en tres o más ejes multipropósito, controlado automáticamente y reprogramable (ISO Standard 8373:1994, Manipulating Industrial Robots – Vocabulario).

Robot móvil: Robot con sistema locomotor.

Robótica: Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales.

Robótica pedagógica: Robótica Pedagógica: actividad de concepción, creación y puesta en práctica, con fines pedagógicos de objetos tecnológicos que son reducciones fieles y significativas de procedimientos y herramientas robóticas bastante usadas en la vida cotidiana, de forma especial en el medio industrial (*Martial Vivet, 1990, citado en Ruiz-Velasco, 2008).*

Sensores: Son dispositivos usados para convertir una variable física difícil de medir directamente (como velocidad, posición, aceleración, etc.) En otra variable medible (generalmente en variables eléctricas).

Servo-control: control realizado por servosistemas.

Servo-Sistema: sistema de bucle cerrado que permite asegurar el control de una magnitud de salida cualquiera como desplazamiento, temperatura, velocidad, etc., a partir de una magnitud de entrada llamada magnitud de referencia[1].

Set-point: Es el valor deseado que se busca obtener en la variable de interés de un proceso.

Sistema de control: Estructura que contiene la logica y programación para las acciones a realizar por robot.

---

[1] Milsant, 1972.p. 5.

Tele-manipuladores: Mecanismos controlados a distancia por un operador.

Tele-robótica: Área de la robótica donde las tareas de percepción del entorno, planificación y manipulación compleja son realizadas por humanos. Donde el operador se encarga de cerrar el bucle de control.

Vehículos omnidireccionales: vehículos que sin girar pueden moverse en cualquier dirección y que además de ello pueden girar sobre sí mismos

Formulación del problema.

El hombre, mediante el avance de la tecnología, ha dejado atrás el trabajo pesado y repetitivo a las maquinas, las cuales han comenzado a reemplazarlo en la realización de dichas tareas, ahora éste usa más el ingenio que la fuerza bruta, empleando su capacidad de análisis y a las maquinas, como herramientas para el desarrollo de estos trabajos. En esta búsqueda constante por mejorar la calidad de vida es cuando surgen las necesidades de crear cada vez mejores máquinas, situación que lleva a los ingenieros afines a la robótica a estar siempre en la búsqueda de mejores soluciones.

Con el auge de la revolución tecnológica y la globalización, ningún país puede hacer caso omiso a estos avances tecnológicos, Colombia un país en vía de desarrollo y con capacidad de exportación, no puede dar la espalda a esta realidad. Siendo la ciencia y la investigación uno de los principales motores en el avance de un país, las universidades tienen la responsabilidad social de hacer frente a esta necesidad creando tecnología, formando a profesionales aptos, y con capacidad de desarrollo e investigación.

La Corporación Universitaria de la Costa, en su programa de ingeniería electrónica tiene como asignatura la robótica, asignatura con un alto grado de complejidad tanto para la enseñanza como en el aprendizaje, debido a las múltiples áreas involucradas en esta. Teniendo en cuenta lo anterior y con la obligación de ser más competitivos y mejores profesionales, nace la necesidad de reforzar estos conocimientos de una forma práctica y didáctica, tanto para la enseñanza como en el estudio de la robótica, por medio de unas herramientas pedagógicas que se convierten en el medio por el cual se afianzan los conocimientos adquiridos en la teoría, contribuyen al desarrollo de habilidades investigativas y flexibilizan la enseñanza de los maestros mostrándola de una forma didáctica y aplicable.

¿Cómo reforzar el aprendizaje y enseñanza de la robótica por medio de una herramienta didáctica?

Justificación.

El robot pedagógico sirve no solo para modelar un proceso real, de manera experimental y a menor escala, sino, para ayudar a comprenderlo y enseñarlo; une la ruptura existente entre teoría y práctica, brindando significados a los conceptos aprendidos en los salones de clase, es una herramienta necesaria para los estudiantes, que les permite puntualizar sus ideas y resolver sus dudas mediante el uso del robot pedagógico. Por medio de esta herramienta didáctica, se pueden definir diferentes tareas, conociendo así el comportamiento de los tipos básicos de robot existentes, pudiendo modificar su programación, reforzando así las habilidades en esta área y en la del pensamiento lógico. Los profesores tendrán la posibilidad de resolver mayor cantidad de dudas de manera visual y experimental, podrán realizar ejemplos y demostraciones a los alumnos, incentivando la investigación, la búsqueda de conocimiento, y por ende el desarrollo tecnológico.

Con el desarrollo de este proyecto los principales beneficiados serán los profesores y estudiantes de Ingenierías Eléctrica, Electrónica, Sistemas y afines.

Sirviendo de herramienta a los profesores para complementar la metodológica en sus clases, incentivando a la investigación, se podrá interactuar con la realidad, haciendo más interactiva la asignatura. Por otro lado, los estudiantes tendrán la facilidad de realizar sus proyectos de manera real, por medio del cual podrán afianzar sus conocimientos, realizar investigaciones relacionadas con la robótica, tener acceso a una herramienta didáctica y de fácil manejo, lo que ayudará a que estos puedan observar de manera práctica todas las implicaciones que tienen los procesos robóticos.

Con la implementación del Robot pedagógico se busca reforzar el aprendizaje y la enseñanza de la robótica por medio de esta herramienta didáctica, robusta y versátil, con el fin de hacer lo más real, aplicada, y metodológicamente posible, la enseñanza y el aprendizaje de la robótica.

La Robótica Pedagógica se encarga de utilizar robots diseñados con el fin de proporcionar mayor entendimiento de procesos reales de manera experimental, y a menor escala. Para llevar a cabo este proceso la robótica pedagógica involucra muchas maneras del aprendizaje durante el mismo, brindando soluciones a problemas de muchas áreas del conocimiento, tales como matemática, robótica, física, y otras ciencias experimentales. Ofreciéndole por consecuencia la posibilidad al estudiante de inducir, descubrir de forma guiada conceptos nuevos o reforzar los aprendidos mediante la experimentación, permitiendo así al estudiante cultivar su propio conocimiento.

Los Robots Pedagógicos son usados para enseñar en diferentes áreas, por lo que un robot pedagógico resultará una herramienta poderosa para el estudio y la enseñanza de la robótica.

Objetivos.

1. Objetivo general.

- Diseñar y construir un Robot móvil orientado a la práctica y enseñanza de la robótica.

2. Objetivos específicos.

- Crear un conjunto de tareas con cierto de grado de flexibilidad (abierto a la creatividad) con el que el estudiante pueda realizar experimentos y pruebas usando el robot.
- Implementar estructuras adecuadas y sencillas de armado para los kits de lego robótica.
- Definir las tareas primordiales para la comprensión básica de la robótica móvil.
- Realizar 5 guías de laboratorio para utilizar lego NXT en la enseñanza y aprendizaje de la robótica móvil.
- Documentar las clases y métodos principales para el uso de leJOS programando Lego NXT.

Marco referencial.

3. Marco histórico

La robótica pedagógica es un área relativamente nueva, en la cual apenas las universidades están comenzando a interesarse para realizar sus investigaciones, puesto que no ha avanzado tanto como lo ha hecho el área de la robótica.

El trabajo más antiguo y directamente relacionado con los objetivos del proyecto que se ha encontrado, es de 1996; titulado "LA ROBÓTICA PEDAGÓGICA" (Cabrera, 1996); artículo publicado en la Universidad Tecnológica de Netzahualcóyotl, donde el autor describe la tarea que han desempeñado los robots a lo largo de la historia, y que a pesar de que no se les puede encomendar la tarea de enseñar, sí se les puede usar como mecanismos pedagógicos y lúdicos para esta enseñanza; incentivando la investigación por medio de la curiosidad hacia la ciencia, ingeniería y en general todas las áreas que abarcan la robótica.

En otra investigación sobre robótica pedagógica llamada "ROBÓTICA PEDAGÓGICA VIRTUAL PARA LA INTELIGENCIA COLECTIVA" (Ruiz-Velasco, s.f.), se muestra, cómo es posible concebir, desarrollar e implementar entornos de enseñanza-aprendizaje virtuales con robótica pedagógica. La robótica pedagógica es constructivista, colaborativa, cooperativa e interdisciplinaria. Estas cualidades le permiten al estudiante auto-afirmarse y aprender; aprovechando las condiciones de trabajo, tiempo y espacio para desarrollar situaciones didácticas de aprendizaje en ambientes virtuales.

En la ciudad de México los investigadores del trabajo "HERRAMIENTAS LÚDICO-TECNOLÓGICAS PARA LA ENSEÑANZA. LOS MECANISMOS ROBÓTICOS Y SUS APLICACIONES EN EL AULA" (García, 2002). Expuesta en el VIII congreso internacional de informática en la educación; evalúan directamente las ventajas que pueden traer las herramientas lúdicas-tecnológicas en la enseñanza, y como la educación ha ido evolucionando cada vez involucrando más a la tecnología como

apoyo para incentivar la investigación, en este mismo trabajo se muestra como por medio de un mecanismo robótico pedagógico se pueden ver resultados positivos en un aula.

En Colombia la Ingeniera Mónica Sánchez (Máster en Ingeniería Electrónica y Computadores de la Universidad de los Andes) en su trabajo "AMBIENTES DE APRENDIZAJE CON ROBÓTICA PEDAGÓGICA", muestra de qué manera se utiliza la robótica pedagógica como instrumento para que el estudiante aproveche su potencial y logre afianzar los conocimientos adquiridos; en este trabajo se expone también que: "Dado el carácter polivalente y multidisciplinario de la robótica pedagógica, ésta puede ayudar en el desarrollo e implantación de una nueva cultura tecnológica en todas las regiones del país, permitiendo el entendimiento, mejoramiento y desarrollo de sus propias tecnologías" (Sánchez, s.f.). Dicho trabajo está basado en las experiencias de los estudios realizados por el investigador de la UNAM (Universidad Autónoma de México), Enrique Ruiz-Velasco.

Esta misma autora plantea una serie de interrogantes en otra publicación titulada "IMPLEMENTACIÓN DE ESTRATEGIAS DE ROBÓTICA PEDAGÓGICA EN LAS INSTITUCIONES EDUCATIVAS" (Sánchez, 2003). Donde uno de estos interrogantes establece por qué promover el uso de la robótica pedagógica en las instituciones educativas; planteando que la presencia de tecnología en los salones de clase mezcladas con unos métodos adecuados de pedagogía terminan aportando experiencias que finalmente contribuyen directamente al desarrollo de la creatividad y el pensamiento de los estudiantes, asimismo los estudiantes adquieren habilidades para estructurar y llevar a cabo investigaciones para luego resolver problemas reales.

Algunos de los logros de los estudiantes que participan en este ambiente de aprendizaje son:

Construir estrategias para la resolución de problemas. Utilizar el método científico para probar y generar nuevas hipótesis sobre la solución, de manera experimental, natural y vivencial desde su perspectiva.

Utilizar vocabulario especializado y construir concepciones propias acerca del significado de cada objeto que manipulan. Además, tomar conciencia de su proceso de aprendizaje y valorar su importancia, al ocupar su tiempo libre en una actividad mental permanente y retadora.

El doctor Enrique Ruiz Velasco (uno de los principales autores que se han encontrado con respecto al tema) describe en la publicación "CIENCIA Y TECNOLOGÍA A TRAVÉS DE LA ROBÓTICA COGNOSCITIVA" un programa teórico para la construcción de un robot como recurso didáctico para aprender ciencia y tecnología. También demuestra como la robótica cognoscitiva puede ser un medio de integración que se convierte en base de conocimientos, a través de la manipulación y control de entornos robotizados, con los que a la vez se resuelven problemas concretos. En la tesis "DISEÑO Y CONSTRUCCIÓN DE UN ROBOT MÓVIL ORIENTADO A LA ENSEÑANZA E INVESTIGACIÓN" (Muñoz, 2006), los investigadores plantean la solución a la necesidad de una herramienta pedagógica, usando sus conocimientos y experiencias; y es así como llevaron a cabo este proyecto.

En este escrito se da a conocer la plataforma como un robot para desarrollar experimentación e investigación en ambientes reales. Estos robots móviles poseen características similares a los robots industriales.

Con la elaboración de este tipo de plataformas se logra sensibilizar y capacitar a los estudiantes en varias de las áreas de la Ingeniería con mayor desarrollo en la actualidad, como lo son la robótica y la inteligencia artificial; además, se deja abierta la posibilidad para que los estudiantes se vinculen formalmente a proyectos relacionados con estas áreas, y/o a la creación de cursos, dentro del plan de estudios, que cuenten con varios de estos prototipos como elementos de práctica en los laboratorios de las universidades. Todo esto también hace que las instituciones educativas se interesen por temas relacionados con la robótica y sus afines.

Se resalta la necesidad que, hacia el futuro, las Universidades y especialmente las facultades de Ingeniería, apoyen los grupos de investigación formados por

estudiantes, los cuales constituyen un promisorio semillero de ingenieros(as) con orientación investigativa y con un alto nivel de desarrollo en su creatividad e ingenio.

Además de las universidades, grupos de estudiantes y en general instituciones educativas, las industrias también han mostrado interés con lo referente al desarrollo de cierto número de kits para la construcción de robots, diseñados para estimular el aprendizaje de conceptos y métodos relativos a la educación de estudiantes en diversas áreas del conocimiento. Los kits incluyen sensores sencillos, pequeños motores, ruedas, engranajes, poleas, relés y en general todo lo que el estudiante necesite para construir robots. LEGO es una de estas empresas que se han ocupado de producir dichos kits aptos para los procesos de educación-aprendizaje. Esta información se encuentra consignada en el documento "LA ROBÓTICA COMO HERRAMIENTA PARA LA EDUCACIÓN" (Miglino, 1999).

4. Marco teórico.

4.1. Definición del término Robot.

El término robot fue utilizado por primera vez en el año 1921, cuando el escritor checo Karel Capek le da vida a su obra de ciencia ficción Rossum´s Universal Robot (R.U.R). Esta palabra proviene de la palabra eslava robota, que traduce trabajo forzado, o trabajo realizado de manera forzada. Los robots de la obra eran androides fabricados a partir de una fórmula que obtuvo un científico llamado Rossum. Estos androides realizaban las labores físicas que les ordenaban sus jefes humanos hasta que se enfrentaron a sus dueños, destruyendo toda la vida humana, a excepción de uno de sus creadores, pues tenían la esperanza de que este les pudiera enseñar a reproducirse (Ollero, 2007).

Este término a pesar de haber tenido una amplia aceptación y que pronto fue aplicado en autómatas construidos durante los años veinte y treinta, los cuales fueron exhibidos en ferias, promociones de productos, y otras aplicaciones más o menos festivas, posiblemente hubiera sido olvidado sino es porque varios escritores de ciencia ficción de la época retomaron el mensaje de la obra de Capek. Siendo

Isaac Asimov el más reconocido exponente de este tema, y fue cuando en octubre de 1945 fecha en la cual publicó en la revista Galaxy Sciencia Fiction su historia en donde expresa sus tres famosas leyes de la robótica.

1. Un robot no puede perjudicar a un ser humano, ni con su inacción permitir que un ser humano sufra daño.

2. Un robot ha de obedecer las órdenes recibidas de un ser humano, excepto si tales órdenes entran en conflicto con la primera ley.

3. Un robot debe proteger su propia existencia mientras tal protección no entre en conflicto con la primera o segunda ley.

A Isaac Asimov se le atribuye la creación del término robótica.

4.2. Generalidades de la robótica.[2]

El robot como máquina ha tenido un desarrollo independiente del término. A diferencia de los robots de los libros, películas y ferias, los cuales tenían rasgos similares a los humanos y existían solo como ideas concebidas literariamente. Los primeros robots reales surgieron en los ámbitos industriales, eran una mezcla entre control automático y tele-operación (maquinas operadas por hombres), pero estas no eran totalmente autónomas, y lo que se busca, es precisamente crear maquinas "independientes", que puedan gobernar un proceso sin la intervención exterior del hombre. Estos sistemas basan su funcionamiento en la retroalimentación de señales, fijándose al inicio del proceso una señal de referencia la cual va a ser constantemente comparada con la señal medida, y en caso de error o diferencia entre estas dos el sistema hará lo necesario para corregir esta desviación. En los sistemas de control automático este lazo cerrado de acción-medida-acción se realiza sin intervención del hombre.

[2] Ollero, 2007.

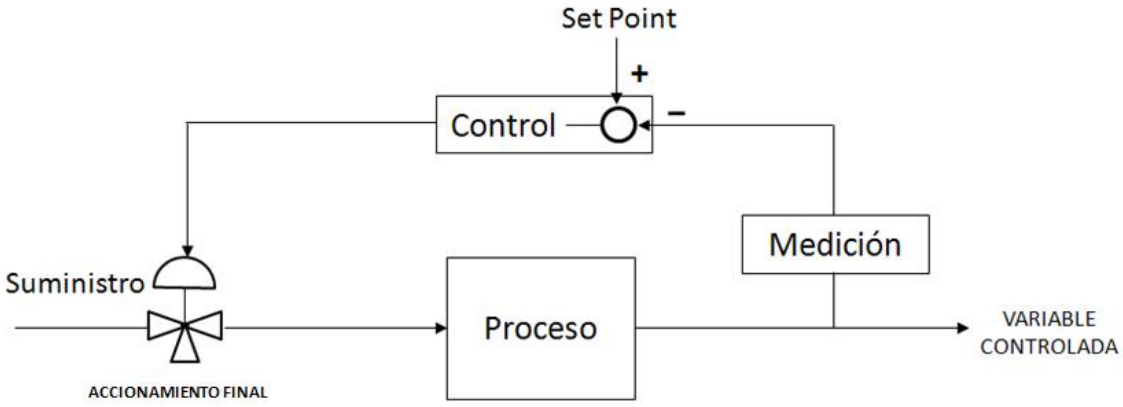


Figura 1. Sistema de control automático

La automatización industrial comenzó a usar sistemas de control automático alrededor del siglo XIX, pero en el siglo XX es cuando empieza a extenderse de forma importante en todos los sectores industriales. Poco a poco fueron naciendo los sistemas de control automático para procesos industriales, sistemas de control de posición, velocidad y control automático de trayectoria (piloto automático) el cual tuvo un gran auge en la aviación, navegación marítima y espacial. Así como también en el posicionamiento de radares.

La tecnología empleada en la realización de dichos sistemas de control tiene un alto nivel de complejidad y requieren de amplios conocimientos de diferentes áreas tales como la neumática, hidráulica, eléctrica y posteriormente a estas áreas se suma la electrónica.

Los primeros robots con los que se comenzó a trabajar fueron llamados tele-operados, y fue R.C Goertz quien desarrolló el primer prototipo en el año 1948, este prototipo tenía el objetivo de manipular elementos radioactivos sin riesgo para el operador. Funcionaba con un dispositivo mecánico maestro-esclavo. El manipulador maestro, que se encontraba en la zona segura, era movido directamente por el operador, mientras que el esclavo, que estaba unido mecánicamente al maestro, se encontraba en contacto con los elementos radioactivos, reproducía exactamente los movimientos del operador, el cual observaba a través de un grueso cristal el resultado de todas sus acciones,

mientras sentía a través del dispositivo maestro las fuerzas que el esclavo ejercía sobre el entorno; esto debido a un servo-sistema con retroalimentación de fuerza hacia la pinza (Milsant, 1972).

Luego de varios años, se reemplazó la tecnología electrónica y el servo-control para sustituir, a la transmisión mecánica, con lo que se desarrolló el primer tele-manipulador con servo-control bilateral, el cual combinaba la articulación de un tele-operador con el eje servo-controlado por medio de una maquina de control numérico.

Otro de los pioneros de la tele-manipulación fue Ralph Mosher, ingeniero de la General Electric que en 1958 desarrolló un dispositivo denominado Handy-Man. Dispositivo que consistía en dos brazos mecánicos tele-operados mediante un maestro del tipo exoesqueleto. Durante varios años la beneficiada principal de esta tecnología fue la industria nuclear, luego la industria submarina comenzó a interesarse por el uso de tele-manipuladores, a la cual posteriormente se sumó la industria espacial en la década del 70.

La primera patente de un aparato robótico fue solicitada en Reino Unido. Sin embargo, en Estados Unidos (EU) es donde se dieron los desarrollos y se establecieron las bases del robot industrial que se utiliza hoy en día. Los primeros desarrollos en EU fueron hechos por el ingeniero George C. Devol, quien creó un dispositivo de transferencia de artículos programado. Devol da a conocer su idea a Joseph F. Engelberger, director de ingeniería de la división aeroespacial de la empresa Manning Maxwell y Moore en Estados Unidos. Devol y Engelberger comienzan trabajando en el uso industrial de sus máquinas, conforman una sociedad e instalan su primera máquina Unimate (1960) el cual realizaba soldadura automática, obedeciendo a ordenes almacenados en un tambor magnético en la fábrica de General Motors de Trenton, Nueva Jersey. Al mismo tiempo grandes empresas, comenzaron la construcción de máquinas similares (Ollero, 2001),

Para la misma década de los 60’s se crea la primera asociación de robótica del mundo, la Asociación de Robótica Industrial de Japón, que le proporcionó un impulso significativo, dándole ventaja competitiva con respecto a Estados Unidos.

Poco tiempo después los norteamericanos fundaron el Instituto de Robótica de América. Mientras que Europa tardó más en comenzar la construcción de robots y crean la Federación de Robótica. A partir de estos inicios el crecimiento de la robótica industrial ha venido creciendo aceleradamente; en menos de 30 años los robots han tomado posiciones en casi todas las áreas productivas y tipos de industrias, debido a que tanto en pequeñas como en grandes empresas los robots son capaces de reemplazar al hombre en tareas repetitivas y hostiles, también con la incorporación de un programa de ordenador para controlar a los mismos permitiéndoles adaptarse con mayor facilidad a los cambios.

La evolución de los tele-manipuladores a lo largo de los últimos años no ha sido tanta como la de los robots autónomos (Los robots autónomos son aquellos robots con capacidad de percepción y procesamiento suficiente como para desenvolverse en un entorno sin intervención externa), posiblemente, debido a estar reducidos a un mercado selecto y limitado (industria nuclear, militar, espacial, etc.) son en general bastante desconocidos así como también poco atendidos por los investigadores y usuarios de robots. Un tele-manipulador requiere del mando continuo de un operador; salvo por las aportaciones incorporadas con el concepto del control supervisado y la mejora de la tele-presencia, debido a la realidad virtual, sus capacidades no han variado mucho desde sus orígenes.

La configuración de los primeros robots eran las llamadas configuraciones esférica y antropomórfica (Ollero, 2001), de uso especialmente válido para la manipulación. La configuración esférica es la que posee dos articulaciones rotacionales perpendiculares entre sí y una prismática ortogonal a las dos primeras y la configuración antropomórfica o también llamada angular, es la que tiene dos o tres articulaciones rotacionales, de las cuales dos de ellas tienen ejes rotacionales paralelos (las dos finales).

Los progresos de la robótica apuntan a aumentar la autonomía y movilidad. Sin embargo existen muchos robots estáticos encargados de tareas repetitivas como soldar, ensamblar, etc. No obstante existen otras aplicaciones donde se requiere que tanto la morfología como el propio concepto hayan evolucionado, en este grupo

se pueden encontrar robots móviles como robots espaciales, para aplicaciones submarinas, militares, para aplicaciones médicas, y muchos más; los cuales, se encuentran antes de los robots industriales en cuestión de desarrollos tecnológicos.

A causa de dichos desarrollos tecnológicos, el robot industrial es un dispositivo capaz de manipular y cuenta con un control más o menos complejo. Por otra parte, un sistema robotizado es algo diferente, porque contiene el robot junto con su enteorno, y encierra todos los mecanismos que sustituyen al ser humano realizando tareas automáticas en conjunto con otros dispositivos robóticos.

4.3. Esquema general del sistema robot[3].

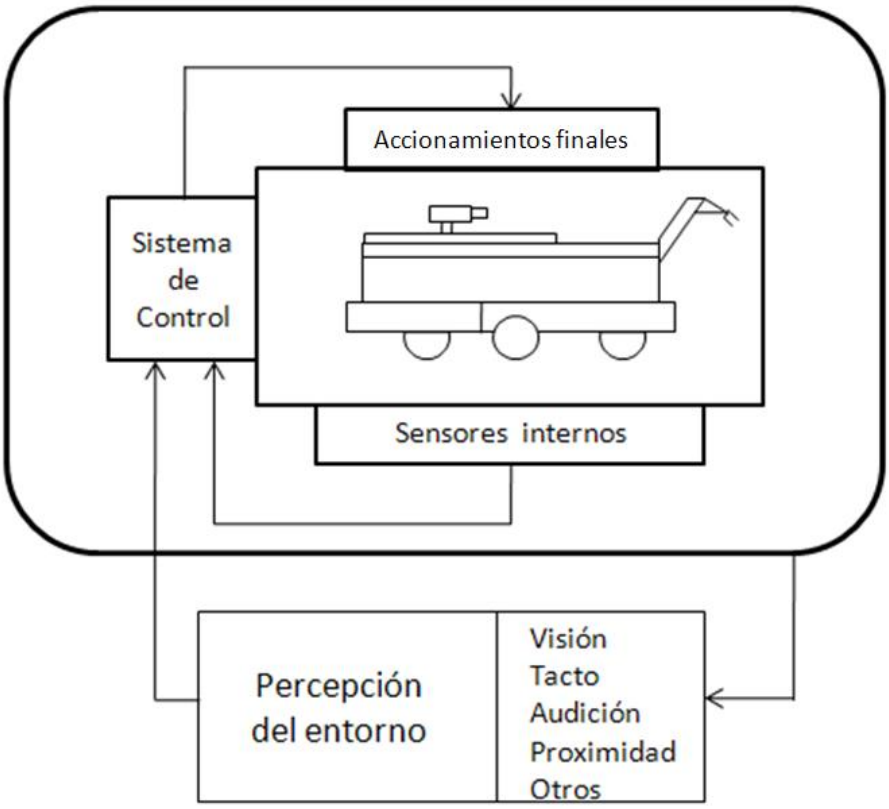


Figura 2. Esquema general del sistema robot.

[3] Ollero, 2001.

En robótica se abarcan funciones de control de movimientos, percepción y planificación, desde la perspectiva del procesamiento de la información.

Los robots están formados por una estructura mecánica, un sistema de accionamiento, sensores (sistema sensorial) los cuales pueden ser internos o externos, un sistema de control y elementos terminales.

El sistema de control de manera general involucra los bucles de realimentación de la información suministrada por los sensores internos y externos, los cuales son usados por el sistema de procesamiento para comparar la información previamente almacenada con la medida de los sensores para entregar posteriormente una respuesta al proceso, (todo esto dependiendo de la lógica manejada y el modelo a seguir en el procesamiento de información).

Los sensores son los dispositivos que dotan al robot de sentidos, son equivalentes a los ojos, tacto, oído, y otros de los que carecemos los humanos; tales como sonares, infrarrojos, giroscopios, entre otros; y sirven para que el robot pueda interactuar con su entorno. Estos se dividen en sensores externos, y sensores internos. Los sensores internos tienen como objetivo medir el estado de la estructura mecánica, en particular, giros o desplazamientos relativos entre articulaciones, velocidades, fuerzas y pares. Estos sensores permiten que se realice el control sobre la estructura del robot, haciendo posible aumentar la precisión y eficiencia del mismo.

Los sensores externos son los que dotan de sentidos al robot. La información que suministran es utilizada por el sistema de percepción para conocer el entorno. Los sistemas de percepción sensorial hacen posible que un robot pueda adaptar automáticamente su comportamiento en función de las variaciones que se producen en su entorno, teniendo posibles soluciones frente a situaciones imprevistas. Lo que no sólo involucra la captación de la información suministrada, sino también su tratamiento e interpretación; para ello el sistema de control del robot incorpora bucles de realimentación sensorial del entorno, generando automáticamente acciones luego de la comparación de dicha información sensorial con patrones de referencia. Para realizar este proceso se requiere un previo conocimiento de la

estructura del entorno, dependiendo de cuan estructurado sea éste, así mismo será la complejidad de la percepción de dicho entorno.

La planificación está relacionada con la percepción, tiene como objetivo encontrar una trayectoria desde una posición inicial a una posición objetivo, sin colisiones. En el caso más simple, el problema se plantea en un entorno que se supone es conocido y está estático, se supone además que el robot es omnidireccional, que se mueve suficientemente lento y que es capaz de seguir el camino de forma perfecta (Ollero, 2001).

4.4. Accionamientos finales.

Los accionamientos finales son dispositivos que convierten una energía eléctrica (generalmente) en un acción sobre el entorno, ya sea mecánica, hidráulica, neumática, etc. (Barrientos, 1997).

En este contexto el objetivo de los accionamientos finales es intervenir en el comportamiento del sistema robótico, siendo común en robótica móvil que estos elementos se encarguen de generar movimiento a partir de las órdenes recibidas desde el sistema de control, los accionamientos pueden ser de tipo neumático, hidráulico o eléctrico.

4.4.1. Accionamientos neumáticos.

La fuente de energía de los accionamientos neumáticos es aire a presión. Existen dos clases de estos Accionamientos; cilindros neumáticos y motores neumáticos, los primeros funcionan desplazando un émbolo ubicado dentro de un cilindro (Barrientos, 1997). Dicho movimiento es producido por diferencia de presiones en los extremos del cilindro. El propósito del cilindro neumático es ubicar el émbolo en los extremos de sí mismo.

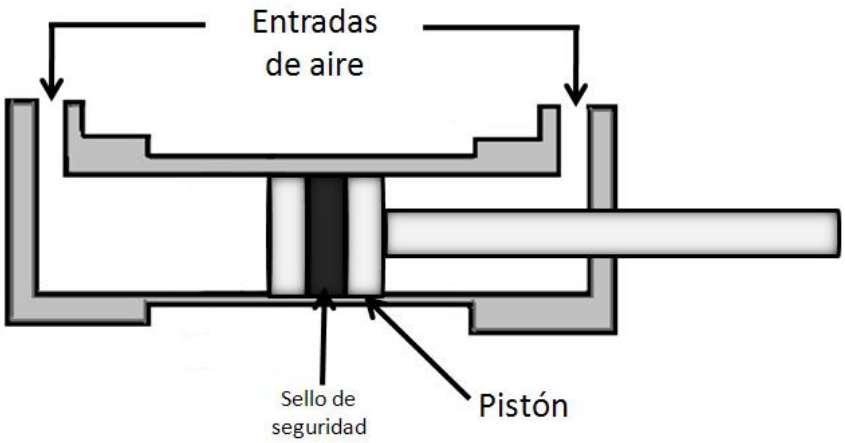


Figura 3. Cilindro neumático.

Son los motores de aletas rotativas y los motores de pistones axiales, los motores neumáticos los más utilizados, en los cuales se consigue el movimiento de rotación por medio de aire a presión.

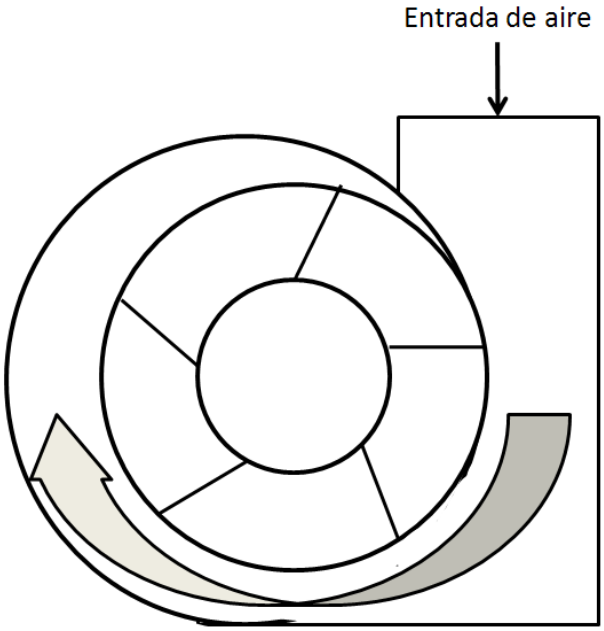


Figura 4. Principio de funcionamiento del motor neumático.

Los accionamientos neumáticos no consiguen alcanzar precisiones muy exactas debido a las características de compresibilidad del aire, impidiendo realizar un adecuado control sobre éstos mismos. No obstante debido a su sencillez este tipo de accionamiento puede ser utilizado en robots que no requieran grandes precisiones.

Se debe tener en cuenta que para utilizar un accionamiento neumático se debe contar con una instalación de aire comprimido, la cual incluye: compresor, tuberías, filtros, secadores, entre otros.

4.4.2. Accionamientos hidráulicos.

Funcionalmente son muy parecidos a los neumáticos, la diferencia radica en que en éstos utilizan fluidos, como aceites minerales, con características de compresibilidad inferior a la del aire, permitiendo alcanzar presiones más altas. Esta diferencia permite alcanzar precisiones mayores, razones por las cuales es más fácil realizar un control continuo sobre éstos; y se hace posible alcanzar mayores pares. Además poseen capacidad elevada de carga, alta lubricación y robustez (Barrientos, 1997).

Debido al uso de fluidos en el sistema hidráulico, se hace necesaria la utilización de filtrado de partículas, eliminación de aire, sistemas de refrigeración y unidades de control de distribución, haciendo la instalación más compleja que para los accionamientos eléctricos e inclusive que para los neumáticos. Por otra parte las altas presiones que se manejan propician fugas de aceite sobre toda la instalación.

4.4.3. Accionamientos eléctricos.

Debido al mejoramiento de las características de control, sencillez y precisión los accionamientos eléctricos han sido más utilizados en los robots móviles Barrientos, 1997), por lo que a continuación se habla en detalle de estos.

- Motores de corriente continua (DC): Son los más utilizados en la actualidad. Están constituidos por dos devanados internos, inductor e inducido, que se alimentan de corriente continua. El inductor, está situado en

el estator y crea un campo magnético de dirección fija. El inducido, situado en el rotor, hace girar al mismo debido a la fuerza que aparece como combinación de la corriente circulante por él y el campo magnético de excitación.

- Motores paso a paso: Son capaces de desarrollar pares suficientes en pequeños pasos. Existen tres tipos de motores de paso a paso; de imanes permanentes, de reluctancia variable e híbridos.

- Motores de corriente alterna: No son utilizados en robótica, al no poderlos controlar con precisión, la velocidad de estos motores se varía con respecto a la frecuencia de la tensión de alimentación; el control de velocidad se realiza por medio de un variador de frecuencia.

- Otros tipos de motores: Existen aplicaciones en las que se requieren motores con características específicas. A continuación se describen algunos (Palacios et al, 2004).

- Motores de corriente continua de pequeña potencia: Su número de revoluciones por minuto es elevado y su par es muy pequeño, los que no les permite ser utilizados en micro-robots a menos que se utilicen reductoras de velocidad o algún otro sistema de regulación.

- Motores de corriente continúa con reductoras: permiten aumentar el par, permitiendo mover el micro-robot con su estructura y fuente de energía que pesa mucho en proporción al tamaño del propio robot.

- Servomotores: los servomotores están construidos por un pequeño motor DC, unas ruedas dentadas que trabajan como reductoras, proporcionándole una potencia considerable, y un circuito con la electrónica necesaria para realizar giros controlados en posiciones angulares específicas por medio del envío de una señal eléctrica codificada.

  El motor que utiliza un servo, es un pequeño motor DC, al suministrarle voltaje en sus terminales, gira en una dirección a alta velocidad pero con

poca fuerza (torque); razón por la cual se hace necesario la utilización de una caja reductora (mecanismo de ruedas dentadas), la cual transforma gran parte de la velocidad de giro en torque.

Debido a las características de alimentación del motor, voltaje DC, se requiere un circuito capaz de controlar el giro del mismo, conformado principalmente por un convertidor de ancho de pulso/voltaje, un amplificador de set-point y un potenciómetro.

El funcionamiento de un servo consiste en convertir el tren de pulsos de la señal de control en voltaje DC, por medio del conversor de ancho de pulso – voltaje. A la salida de este dispositivo se encuentra un amplificador en el que se establece un set-point o punto de referencia, el cual, compara este valor con el de un potenciómetro que está conectado al motor y que gira con éste, para realizar la comparación del estado en el que se encuentra con el de la señal de control; así se determina cuanto gira el motor para ir a la posición requerida.

Estos dispositivos se utilizan en la práctica para mover palancas, pequeños ascensores y timones; como también en radiocontrol, manejo de títeres y en robots por supuesto. Este tipo de motor es de gran uso en robótica, debido a que posee las características necesarias para realizar tareas de alto torque en aplicaciones de tamaños reducidos.

4.5. Sensores.

Los sensores son dispositivos usados para convertir una variable física difícil de medir directamente (como velocidad, posición, aceleración, etc.) En otra variable medible (generalmente en variables eléctricas). Son ampliamente usados en robótica, tanto móvil como industrial ya que son los que permiten que el robot tenga conocimiento de sí mismo y del entorno para poder realizar sus tareas con adecuada precisión, velocidad, exactitud y eficiencia. Para conocer cómo se encuentra su propio estado el robot debe contar con sensores internos, como encoders, giroscopios, acelerómetros, etc. Mientras que para estar al tanto de su

entorno debe contar con sensores externos como sensores de luz, color, temperatura, sonido, infrarrojos, de contacto, de aceleración, ultrasónicos, entre otros.

Entre algunos ejemplos de sensores internos se encuentran los sensores de posición, como los encoders para el control de posición angular. Los encoders normalmente se encuentran acoplados al motor. También se encuentran en esta categoría los sensores lineales de posición, giroscopios y compases magnéticos.

Otros sensores de igual importancia que los de posición son los sensores de velocidad, los cuales permiten al robot saber cuál es su velocidad y regularla a través de un bucle cerrado de control. Para realizar control sobre su estado, el robot también debe contar con sensores de presencia, que le permiten tener un rango de acción determinado, sin obstruir su propio movimiento.

Para conocer el entorno el robot debe contar con sensores que le permitan este objetivo (Palacios, 2004). Se encuentran entre ellos:

- Sensor pasivo de luz: Se utiliza para detectar ausencia o presencia de luz, teniendo en cuenta la intensidad de ésta.

- Sensor infrarrojo: Es un tipo de sensor de luz que utiliza, el espectro infrarroja para la detección de obstáculo. Puede ser pasivo (contiene un fototransistor) y activo (consta de un emisor y receptor). Este sensor realiza las tareas de medir distancia, lo que permite al robot usar esta información para seguir y/o evadir obstáculos.

- Final de carrera mecánico o sensor de contacto: Este sensor funciona como un interruptor mecánico que presenta los estados de abierto o cerrado. Puede ser utilizado para la detección de presencia y proximidad. Los más sencillos son los interruptores mecánicos como los finales de carrera.

- Sensor ultrasónico: Este sensor funciona enviando una breve ráfaga de ultrasonido (habitualmente superior a los 20 KHz) a través del emisor, la cual, tras reflejarse en los obstáculos es captada por el receptor. Para

determinar la distancia el sensor mide el tiempo de espera desde el envío hasta la recepción de la onda ultrasónica. Este dispositivo es útil para medir distancias y para la detección de obstáculos.

4.5.1. Sensores internos.

Entre los sensores internos se encuentran los encoders, giroscopios, acelerómetros y compases.

- Encoders o codificadores ópticos: Los encoders son utilizados para determinar posiciones angulares. Su principio de funcionamiento consiste en enviar un haz de luz, desde un emisor, hasta un foto-detector, dicho haz es periódicamente interrumpido por un patrón de marcas localizadas en un disco, adjunto a un eje. El disco, al rotar genera una serie de pulsos que permiten determinar la cantidad de grados girados.

  Existen dos tipos básicos de encoders ópticos; incremental y absoluto. La versión incremental mide la velocidad rotacional y puede inferir la posición relativa, mientras que los modelos absolutos miden directamente la posición angular e infieren la velocidad. Si la información de posición no volátil no es considerada, los encoders incrementales generalmente proporcionan una resolución equivalente a los encoders ópticos absolutos pero a un mucho menor costo y de forma fácil.

- Compás magnético o brújula: Es usado para la orientación del robot. Su funcionamiento está basado en el de una brújula común. Es decir una aguja imantada colocada sobre una base vertical que le permite girar libremente y apuntar hacia el norte. Las brújulas magnéticas actuales usan anillos magnéticos o barras, sujetos a un disco graduando de mica; que flotan en una mezcla de agua, alcohol y glicerina, de forma que puede rotar alrededor de un pivote.

- Giroscopio: Este sensor es utilizado para la navegación de robots móviles. Con este dispositivo se puede medir el ángulo de giro de un robot, su

funcionamiento consiste, en el principio de conservación del momento angular.

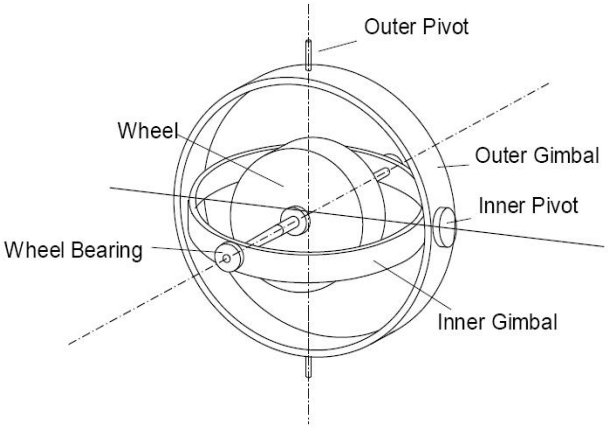


Figura 5. Giroscopio[4]

- Acelerómetro: con este sensor se mide la aceleración del objeto al cual se encuentra unido. Su funcionamiento se basa en el en la segunda ley de newton (F= m*a), en la cual si se conoce la masa y la fuerza del robot se obtiene la aceleración. Entre los acelerómetros más comunes se encuentra el piezoeléctrico. Este se basa en el principio de que cuando se comprime una malla cristalina piezoeléctrica, se produce una carga eléctrica proporcional a la fuerza aplicada.

4.6. Elementos terminales.

Los elementos terminales son los que le permiten al robot interactuar con el entorno. Pueden ser herramientas o también elementos de sujeción; generalmente son diseñados para una tarea específica (Barrientos, 1997).

[4] Feng, 1994. p. 26.

Los elementos terminales generalmente más conocidos son las pinzas, las cuales se utilizan para agarrar y sostener los objetos; entre los elementos se distinguen los que utilizan sistemas mecánicos y las que utilizan otro tipo de dispositivos como ventosas, pinzas magnéticas, adhesivas, ganchos, entre otras.

Entre los elementos terminales tipo herramienta se encuentran los utilizados en tareas específicas como lo son las pinzas de soldadura por puntos, soplete de soldadura por arco, fresa-lija, pistola de pintura, cañón láser y cañón de agua a presión. Existen otras herramientas como manos articuladas y pinzas dotadas de tacto capaces de manipular objetos difíciles y delicados.

4.7. Estructura y paradigmas del sistema de control de robot móviles.

Antes de mostrar los paradigmas más comunes o generales de la robótica, hay que preguntar ¿Qué es un paradigma en la robótica?

Murphy (2000) lo define como “Un paradigma es una filosofía o un conjunto de supuestos de Paradigmas y / o técnicas de que caracterizan un enfoque a una clase de problemas”. En otras palabras, la forma como se enfocan los posibles problemas a los cuales se enfrentará el robot. Existen varios enfoques, o varios paradigmas, y no hay ninguna regla general que diga cual es mejor o peor que otro. Sin embargo el objetivo es encontrar el paradigma más adecuado al problema que se busca resolver, puesto que a pesar de que es posible encontrar más de uno solo que pueda usarse para resolver el mismo problema, se pretende encontrar el más eficiente.

Por ejemplo: La trayectoria que se debe seguir para llegar de un punto a otro. Se pueden tomar múltiples caminos para cumplir esta meta, pero se sabe que de manera general la distancia más corta entre dos puntos, es una línea recta. Ahora, este mismo ejemplo pero con un obstáculo entre los dos puntos. Ya no se podría una simple línea entre ambos. Con este simple ejemplo se puede observar que para cada problema se debe buscar la solución más adecuada, usando el paradigma más

adecuado, por lo que conocer los principales paradigmas, puede ayudar a disminuir el tiempo de diseño y programación de un robot.

De forma general, en robótica existen tres tipos generales de paradigmas. Jerárquico (también conocido como deliberativo), reactivo, hibrido (Jerárquico /reactivo). Para analizar estos paradigmas se usan las principales primitivas en la robótica: planeación, acción, percepción:

Percepción: Tomar la información de los sensores del robot para producir una salida útil para otras funciones.

Planificación: Tomar la información de los sensores o de su base de conocimiento, para generar una o más tareas que el robot debe realizar.

Acción: Es una función que provoca una salida, u orden para los accionamientos finales.

En la figura 6 se pueden ver las primitivas en función de entradas y salidas.

PRIMITIVAS DEL ROBOT	ENTRADA	SALIDA
SENSA	Dato de sensor	Información sensada
PLANEA	Información (sensada y/o cognitiva)	Directivas
ACTÚA	Información sensada o directivas	Comandos de Accionamientos finales.

Figura 6. Primitivas en función de entradas y salidas

Los paradigmas pueden describirse de dos formas: Una, como se interrelacionan las primitivas (percepción, planificación, acción), y dos, la forma como se maneja la información dentro del sistema.

Paradigma jerárquico o deliberativo: Es uno de los primeros paradigmas de la robótica, data de 1967-1990. Es un paradigma que se basa principalmente en Top-Down. , hace un especial énfasis a la etapa de planificación, y fue creado basado en la estructura que se tenia del pensamiento de los seres humanos, en ese entonces. Por ejemplo: “el sujeto ve una pelota, decide si tomarla o no, y luego se dirige hacia ella a tomarla”. Pero varios años después se demostró que no todo se hacía de esa forma.

En el paradigma deliberativo, el robot sensa, planea que hacer con lo sensado, y actúa, (ver figura 7). Donde se ve el diagrama de flujo que explica este enfoque.

Las arquitecturas deliberativas, de forma general pueden proveer una solución óptima, siempre y cuando tal solución exista, el problema esté bien definido y el entorno no cambie. La construcción de modelos universales se hace muy complicado, debido, entre otras cosas, a la necesidad de suponer un *mundo cerrado*. Y siempre la información proveniente de los sensores posee algún grado de incertidumbre y puede estar alterada, o incluir errores que impiden un conocimiento total y real del mundo.

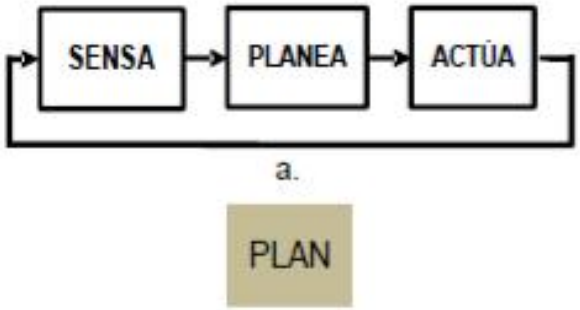


Figura 7. Diagrama del paradigma deliberativo

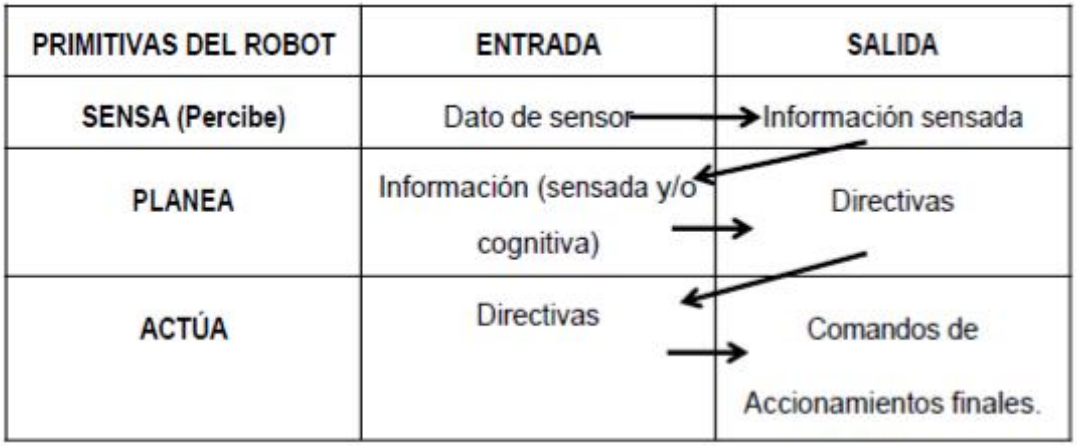


Figura 8. Diagrama deliberativo

Paradigma Reactivo: Este paradigma omite la parte de la planificación, hay una relación directa entre la percepción y la acción, como se puede ver en la grafica 9.

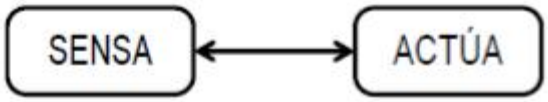


Figura 9. Diagrama del Paradigma Reactivo

Este paradigma nació como una contra-propuesta del paradigma deliberativo, por lo que lleva menos tiempo que este. Alrededor de 15 años. Este enfoque tiene dos circunstancias a su favor que hicieron posible su buena acogida y desarrollo.

Una de estas fue el comienzo de investigaciones acerca la Inteligencia artificial (A.I), con el fin de buscar en la biología y la psicología cognitiva, inteligencia animal que pudiera ser aplicado a las plataformas móviles. Y el otro fue el decremento de los costos de los materiales de microelectrónica con que se fabrican los robots,

estos cada vez se hacen más accesibles, a diferencia del costo del primer robot móvil Shakey, $USD 100,000.

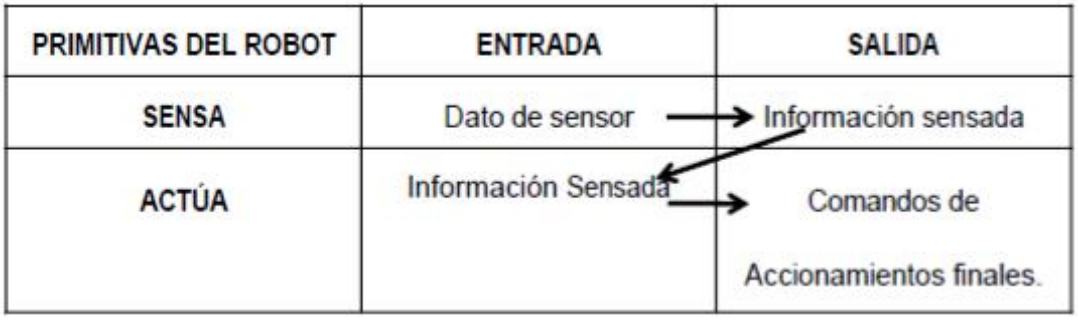


Figura 10. Paradigma reactivo

Teniendo en cuenta este planteamiento descrito y mostrado en la figura 10 se tiene que la salida está directamente relacionada con un valor sensado. Entonces siendo esto así, hay que preguntarse, ¿como hacen los robots para hacer más de una sola tarea? La solución es que existen diferentes procesos concurrentes, que son llamados comportamientos (Murphy, 2000). Entonces los robots programados en comportamientos, usando el control reactivo, descomponen las tareas en múltiples comportamientos, que generalmente son eventos de tipo Sensado-Acción.

Este tipo de comportamiento basado en el paradigma reactivo, ha tomado mucho auge, sobre todo debido a la incorporación de conceptos de comportamiento animal.

Paradigma Hibrido

Se puede inferir al observar el enfoque del paradigma reactivo, que este tiene un buen tiempo de respuesta, debido a que omite la etapa de la planificación. Pero esto termina siendo contraproducente cuando se presentan problemas un poco más complejos; entonces deja de ser un método óptimo para el robot.

Basándose en el paradigma reactivo, nace un nuevo paradigma, que combina el reactivo con el deliberativo, conocido también como, paradigma Hibrido.

En el paradigma hibrido el robot planea como descomponer una tareas en sub-tareas, y cuáles son los comportamientos más adecuados para cada una. Y estos comportamientos son ejecutados de forma reactiva. La interacción entre las primitivas puede verse en la figura 11 y la figura 12 como se relacionan las primitivas de acuerdo a entrada-salida.

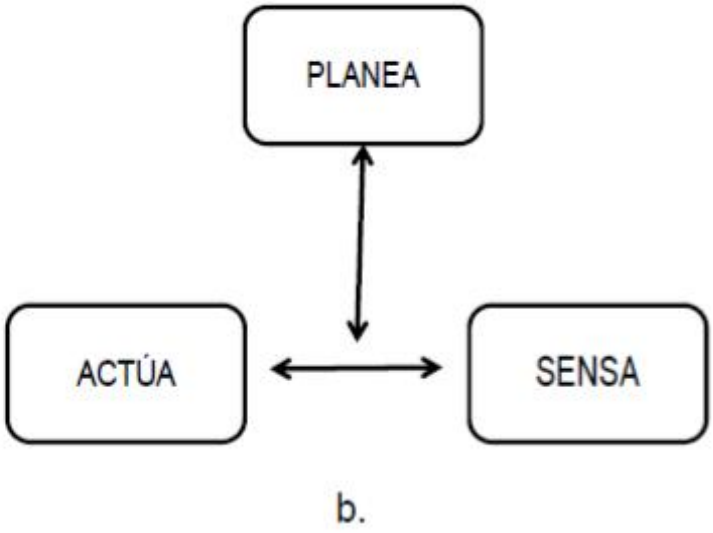


Figura 11. Diagrama del Paradigma Híbrido

PRIMITIVAS DEL ROBOT	ENTRADA	SALIDA
PLANEA	Información (sensada y/o cognitiva)	Directivas
SENSA-ACTÚA (comportamientos)	Datos del Sensor	Comandos de Accionamientos finales.

Figura 12. Paradigma Híbrido

4.8. Tipos de robots.

De los robots se podrían hacer muchas clasificaciones, si se tiene en cuenta sus parámetros o características. En esta sección se señala los robots con más auge:

- Humanoide: es un robot con apariencia humana que busca imitar el comportamiento de éste.
- Robot móvil: es un robot con un sistema locomotor. Este tipo de robot es objeto de estudio en este proyecto. En la sección siguiente se describe con mayor profundidad este tipo de robot.
- Robot industrial: es un robot manipulador diseñado y programado para realizar tareas de mover materiales, herramientas o dispositivos especializados.
- Robot de servicio: Estos robots son diseñados y programados para realizar tareas de educación, cuidado médico, domesticas, intervención en ambientes peligrosos, etc. Estos robots pueden ser móviles o estacionarios.

4.8.1. Robots móviles[5].

Los robots móviles nacen por la misma curiosidad y capacidad del hombre de imitar su entorno, por querer crear maquinas que sean capaces de comportarse semejante a él, al principio las maquinas industriales simplemente eran maquinas que seguían ordenes, pero la robótica móvil busca objetivos más ambiciosos, que son proporcionar al robot de un sistema locomotor acompañado de un sistema de control que lo haga capaz de desenvolverlo en un entorno sin intervención humana.

Desde el punto de vista de la autonomía, los robots móviles tienen como antecesores los dispositivos electromecánicos, creados desde los años treinta para

[5] Ollero, 2007.

desarrollar funciones inteligentes tales como descubrir caminos en laberintos, como lo fue la tortuga de Walter, presentada en 1948, la cual podía reaccionar ante la presencia de obstáculos, subir pendientes y cuando la fuente de energía comenzaba a ser insuficiente se dirigía hacia una posición de recarga.

Luego en la década del ochenta la industria comenzó a usar vehículos autónomos que se desenvolvieran en un entorno fuertemente estructurado, guiados por cables bajo el suelo o por medio de sensores ópticos para seguir líneas, haciendo así más fácil la automatización. Más adelante se vuelve a trabajar en el desarrollo de robots móviles dotados de una mayor autonomía. La mayor parte de las investigaciones se enfocan y basan en el empleo de plataformas que soportaran sistemas de visión. Sin embargo, el desarrollo tecnológico de esa época todavía no daba soporte para poder lograr la navegación autónoma de forma eficiente. El incremento de la capacidad computacional y el desarrollo de nuevos sensores, mecanismos y sistemas de control, logra aumentar esta autonomía. Se pretende que el robot tenga la suficiente "inteligencia" como para reaccionar y tomar decisiones basándose en observaciones de su entorno, sin necesidad que este sea totalmente conocido.

La autonomía de un robot móvil está prácticamente basada en el sistema de navegación automática. En estos sistemas se incluyen tareas de planificación, percepción y control. En la parte de percepción se encuentra toda la sensorica del robot, tanto externa como interna, la etapa de control es lo referente al tipo de control usado (PID, Lógica difusa, control adaptativo) y en la parte de planificación se utiliza la información captada por los sensores, combinada con los objetivos o metas del robot (ver paradigmas de la robótica ítem 4.7) para planear su trayectoria.

En un robot para interiores, la misión podría consistir en determinar a qué habitación hay que desplazarse, mientras que la ruta establecería el camino desde la posición inicial a una posición en la habitación, definiendo puntos intermedios de paso. El vehículo puede desviarse de la ruta debido a la acumulación de imprecisiones mecánicas y de control. Para la planificación de trayectoria existen

numerosos métodos los cuales están basados en hipótesis simplificadoras, donde se supone un entorno conocido y estático, robots omnidireccionales, con movimiento lento y ejecución perfecta de trayectoria. Una vez se haya planificado dicha trayectoria, se requiere planificar movimientos concretos y controlar dichos movimientos para mantener al vehículo en la trayectoria planificada, con lo que surge ahora el problema del seguimiento de caminos, que para vehículos con ruedas se concreta en determinar el ángulo de dirección teniendo en cuenta la posición, orientación actual del vehículo con respecto a la trayectoria que debe seguir, sin olvidar también que hay que resolver el problema del control y regulación de la velocidad del vehículo.

Es necesario disponer de medidas de posición y orientación del robot para realizar un control óptimo, también se requiere que estos datos sean captados e interpretados en intervalos de tiempo suficientemente cortos. La técnica más simple consiste en la utilización de la odometría a partir de las medidas suministradas por los sensores situados en los ejes de movimiento, generalmente codificadores ópticos. Sin embargo, la acumulación de error puede ser muy grande. Para disminuir este error, se emplean sistemas de navegación inercial que incluyen giroscopios y acelerómetros, aunque estos sistemas también acumulan error, especialmente en la determinación de la posición empleando los acelerómetros. Sin embargo la combinación de la navegación mediante odometría con la navegación inercial (medida de los ángulos de orientación) puede dar buenos resultados en intervalos de tiempo y distancia pequeños.

La inevitable acumulación de error hace necesario el empleo de otros sensores. Sobre todo en aplicaciones de exteriores, en las que las distancias que recorre el vehículo autónomo son considerables. Se emplean sistemas de posicionamiento global mediante satélites. En general el sistema de percepción de un robot móvil o vehículo autónomo tiene un triple objetivo: permitir una navegación segura, detectando, localizando obstáculos y situaciones peligrosas en general, modelar el entorno construyendo un mapa o representación de dicho entorno (generalmente geométrica), y estimar la posición del vehículo de forma precisa. Asimismo, el sistema de percepción de estos robots puede aplicarse no solo para navegar sino

también para aplicaciones tales como el control del manipulador situado en el robot.

Para el diseño de estos sistemas de percepción se deben tener en cuenta diferentes criterios, algunos de los cuales son conflictivos entre sí. Es necesario considerar la velocidad del robot, la precisión, el alcance, la posibilidad de interpretación errónea de datos y la propia estructura de la representación del entorno.

En muchas aplicaciones se requiere tener en cuenta múltiples condiciones de navegación con requerimientos de percepción diferentes; por lo que puede ser necesario estimar de forma muy precisa, aunque relativamente lenta, la posición del robot y a la vez, detectar obstáculos lo suficientemente rápido, aunque no sea necesario saber la posición de manera precisa.

Existen arquitecturas en las que el sistema de percepción se encuentra integrado en el controlador de forma que, en entornos estructurados, es posible estimar de forma muy rápida la posición para navegar a alta velocidad. También se han aplicado redes neuronales para generar el ángulo de dirección a partir del sistema de percepción.

Específicamente con los sensores, además de las características de precisión, rango, e inmunidad a la variación de condiciones del entorno, es necesario tener en cuenta su robustez ante vibraciones y otros efectos originados por el vehículo y el entorno, su tamaño, consumo, seguridad de funcionamiento y desgaste.

Las cámaras de video tienen la ventaja de su amplia difusión y precio, su carácter pasivo (no se emite energía sobre el entorno) y que no es necesario el empleo de dispositivos mecánicos para la captación de la imagen. Las desventajas son los requerimientos computacionales, la sensibilidad a las condiciones de iluminación, y los problemas de calibración y fiabilidad.

La percepción activa mediante laser es un método alternativo que ha cobrado una importante significación en robots móviles. Se utilizan dispositivos mecánicos y

ópticos de barrido en el espacio obteniéndose imágenes de distancia y reflectancia a las superficies intersecadas por el haz.

Los sensores de ultrasonido son económicos y simples para la navegación. Se basan en la determinación del denominado tiempo de vuelo de un pulso de sonido (entre 30 KHz y 1 MHz). Sin embargo, la influencia de las condiciones ambientales puede ser significativa, debiendo corregirse mediante una calibración adecuada. Por otra parte, la relación señal/ruido es normalmente muy inferior a la de los otros sensores, lo que puede hacer necesario el empleo de múltiples frecuencias y técnicas de filtrado y tratamiento de la incertidumbre de mayor complejidad computacional, asimismo, la resolución lateral es mala, existiendo para evitarlo técnicas de enfoque mediante lentes acústicas o transmisores curvos.

4.8.1.1. Sistemas de locomoción[6].

Entre los robots móviles se puede encontrar gran variedad de diseños. Son los vehículos con ruedas la solución más sencilla y fácil de implementar en superficies firmes y libres de obstáculos consiguiendo velocidades relativamente altas. Este tipo de diseño presenta problemas de deslizamiento debido a la impulsión, y sobre todo la locomoción en superficies blandas es bastante ineficiente.

Los robots móviles utilizan distintas clases de locomoción con ruedas que les permiten poseer cualidades y características diferentes en lo que se refiere a la eficiencia energética, dimensiones, cargas útiles y maniobrabilidad. Con los vehículos omnidireccionales es posible conseguir la mayor maniobrabilidad. Debido a que logra desplazarse en el plano simultáneamente e independiente en cada eje del sistema de coordenadas, y rotar según el eje perpendicular.

Tipo Ackerman:

---

[6] Ollero, 2007.

Éste es el sistema de locomoción que usan los vehículos con cuatro ruedas, es el resultado de la modificación de vehículos de transporte como automóviles o incluso vehículos más pesados como camiones, entre otros. En la siguiente figura se ilustra esta configuración.

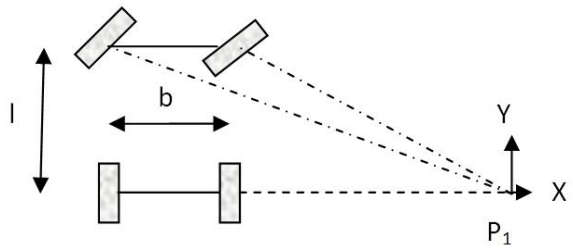


Figura 13. Estructura Ackerman

La rueda delantera interna (lado hacia donde se mueve el vehículo) gira en un ángulo sutilmente superior a la exterior con el propósito de eliminar el deslizamiento.

Triciclo clásico: Este sistema de locomoción se ilustrado en la siguiente figura.

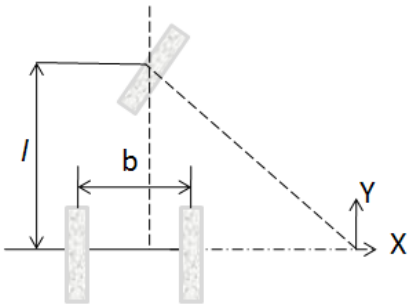


Figura 14. Estructura triciclo clásico

La rueda delantera sirve tanto para la tracción como para el direccionamiento. El eje trasero, con dos ruedas laterales, es pasivo y sus ruedas se mueven libremente. La maniobrabilidad es mayor que en la configuración Ackerman pero puede presentar problemas de estabilidad en terrenos difíciles. El centro de gravedad tiende a desplazarse cuando el vehículo se desplaza por una pendiente, causando la pérdida de tracción. Debido a su simplicidad es bastante frecuente en vehículos robóticos para interiores y exteriores planos pavimentados.

Locomoción diferencial:

El direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue también con estas mismas ruedas. Adicionalmente existen una o más ruedas para soporte. En la siguiente figura se ilustra este sistema.

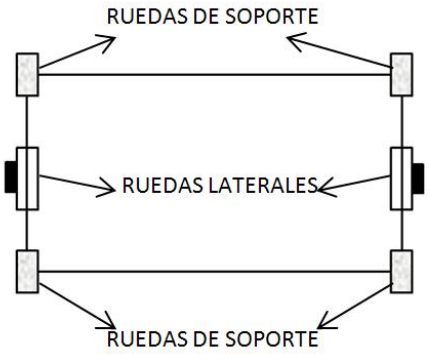


Figura 15. Estructura Direccionamiento diferencial

Esta configuración es la más frecuente en robots para interiores.

Skid Steer:

Se disponen varias ruedas en cada lado del vehículo que actúan en sincronía, combinando los movimientos de las ruedas de la derecha con los de la izquierda logrando de esta manera el movimiento requerido.

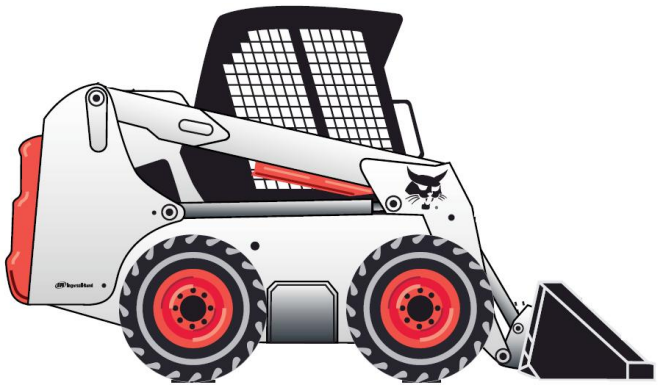


Figura 16. Cargadora Bobcat[7]

Pistas de deslizamiento:

Son vehículos que en vez de utilizar ruedas usan pistas de deslizamiento, por medio de las cuales se consigue la impulsión y el direccionamiento de los mismos. Esta configuración es funcionalmente análoga a la Skid Steer; las pistas actúan de manera análoga a ruedas de gran tamaño. La locomoción con pistas de deslizamiento es útil en terrenos irregulares.

Síncronas:

Consiste en tener tres o más ruedas mecánicamente acopladas de una forma en que todas giran en la misma dirección a la misma velocidad. La sincronización mecánica requerida puede ser alcanzada de algunas maneras, la más común es por medio de cadenas, correas o engranajes. En la siguiente figura se ilustra un ejemplo de la configuración síncrona con cadenas.

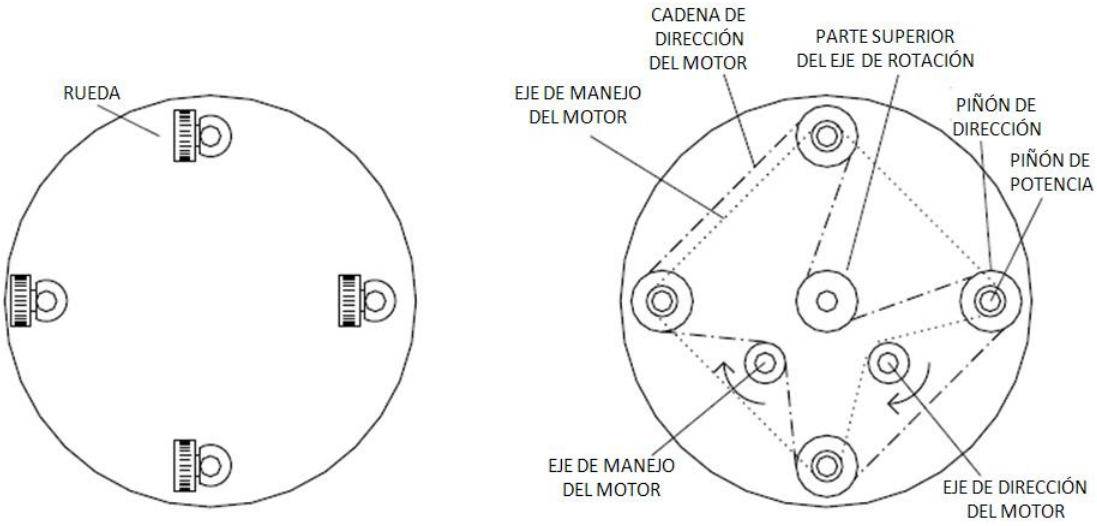


Figura 17. Estructura síncronas

[7] Bobcat, 2009.

Otras configuraciones:

Se puede señalarse la utilizada por el robot RAMI (Ollero et all, 1995). El sistema de locomoción de RAMI se ilustra en la siguiente figura.

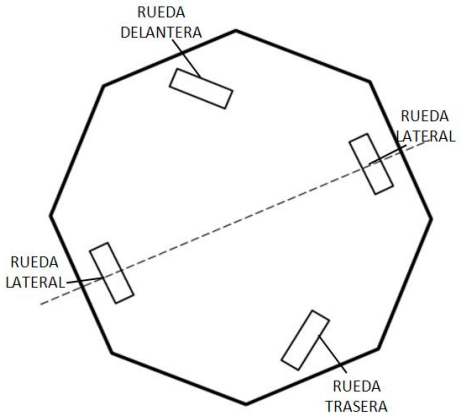


Figura 18. Sistema de locomoción de RAMI

Consiste en cuatro ruedas dispuestas en las diagonales de un rombo, con la diagonal principal según el eje longitudinal del vehículo y dos ruedas laterales actuadas de forma independiente.

Al actuarse en el eje longitudinal las ruedas delanteras y traseras giran en sentido contrario. El sistema combina este direccionamiento con el diferencial, que se consigue actuando de forma independiente las ruedas laterales que impulsan el vehículo. Con este sistema es posible tener un radio de giro nulo y se mantienen las buenas propiedades del guiado diferencial en el seguimiento de caminos.

Otras configuraciones consisten en el empleo de ruedas especiales tales como las denominadas “ruedas suecas” que permiten conseguir el movimiento omnidireccional de un vehículo.

Locomoción mediante patas[8].

[8] Ollero, 2007.

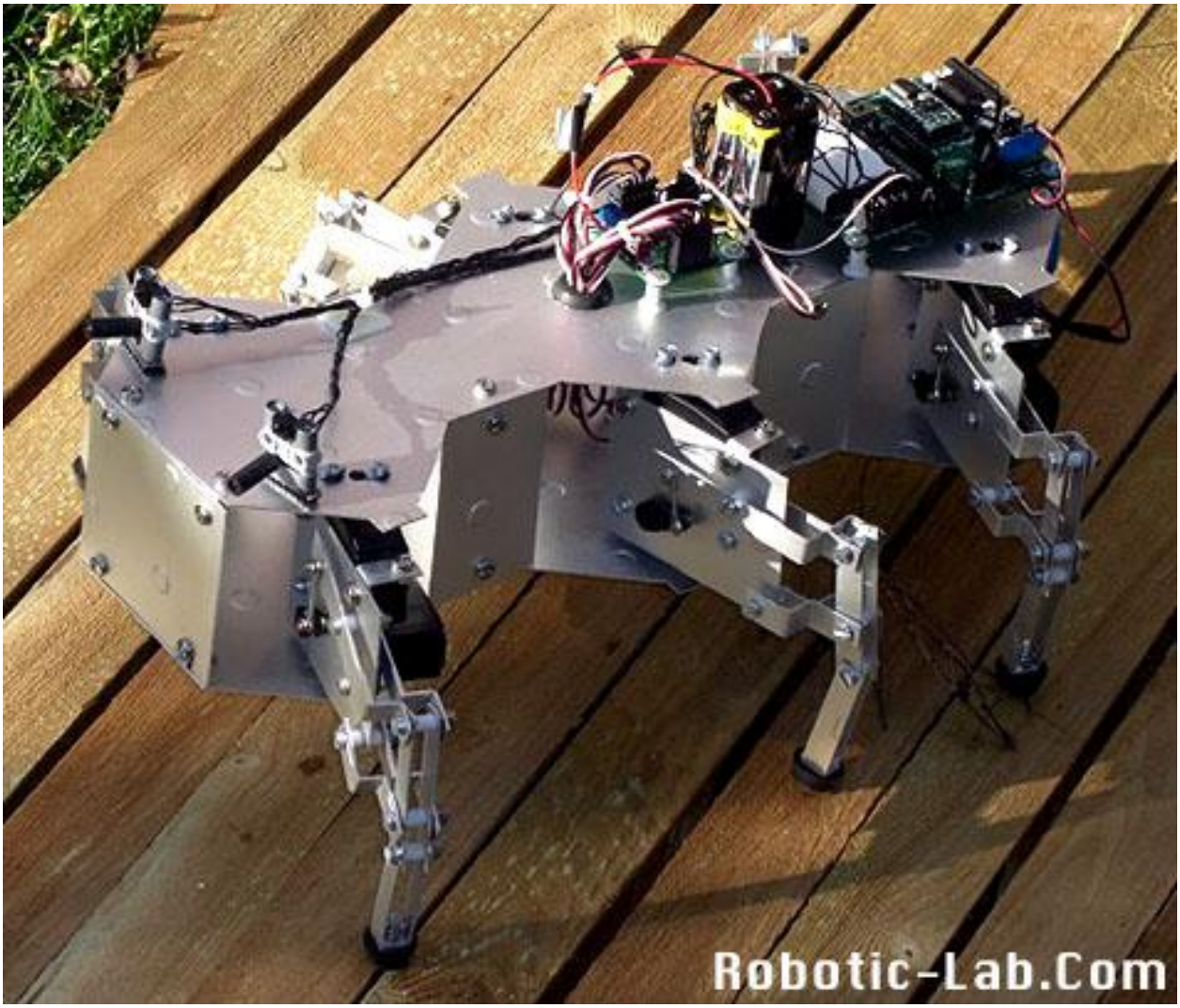


Figura 19. Robot Hexápodo (RH-1)

Permite aislar el cuerpo del terreno empleando únicamente puntos cruciales de soporte. En este tipo de robot se hace posible modificar la plataforma de soporte para permitirle desplazarse tanto por terrenos planos como por terrenos difíciles y con obstáculos, se puede incluso conseguir omnidireccionalidad y un mejor deslizamiento sobre la superficie de desplazamiento (Ollero, 2001).

Los mecanismos necesarios para la locomoción en los robots con patas son de mayor complejidad que en los vehículos con ruedas; así como los problemas de planeación, control, e incluso ocasiona un mayor consumo de energía.

Las configuraciones más utilizadas son las de seis u ocho patas. Pero también en esta categoría se pueden encontrar los robots trepadores, útiles en tareas como inspecciones y reparaciones en paredes verticales; su método de sujeción a la superficie es mediante pinzas generalmente, aunque también emplean dispositivos de succión o magnéticos en las patas y hasta sistemas mixtos de ruedas y patas.

Configuraciones articuladas.

En esta configuración se utilizan generalmente dos o más módulos con ruedas, articulados entre sí, lo que le permite adaptarse a terrenos blandos, inestables o de difícil acceso (Ollero, 2001). Las configuraciones articuladas con gran cantidad de eslabones son las ideales para caminos estrechos. Gracias a la redundancia de la estructura se garantiza mayor seguridad, lográndose incluso intercambiar segmentos y hasta su transporte se ve facilitado.

Robots submarinos y aéreos.

El desarrollo de robots submarinos y aéreos ha sido motivado por el hombre en aplicaciones tales como la inspección, recogida de datos o mantenimiento de instalaciones en entornos naturales a los que el acceso éste resulta muy difícil, o incluso imposible.

Estos vehículos han surgido de la evolución de vehículos total o parcialmente tele-operados u operados por el hombre.

4.8.1.2. Orientación y planificación para robot móviles[9].

Hay varios métodos de planificación de caminos para robots móviles que se basan en hipótesis de un entorno conocido y estático, robots omnidireccionales, con movimiento lento y ejecución perfecta de trayectoria, en particular, hay muchos métodos que buscan caminos libres de obstáculos que minimizan la distancia

[9] Ollero, 2007.

recorrida en un entorno modelado mediante polígonos. En otros casos, se modela el espacio libre tratando de encontrar caminos por el centro del mismo. Para facilitar la búsqueda existen técnicas de descomposición del espacio en celdas (Torpe, 1948), utilización de restricciones de varios niveles de resolución y búsqueda jerarquizada (Kambhampati y Davis, 1988; Stenz 1990, citado en Ollero, 2007) que permiten hacer más eficiente el proceso.

La planificación de la trayectoria puede realizarse también de forma dinámica, considerando la posición actual del vehículo y los puntos intermedios de paso definidos en la planificación de la ruta. La trayectoria se corrige debido a acontecimientos no considerados. La definición de la trayectoria debe tener en cuenta las características cinemáticas del vehículo. Por ejemplo en vehículos con ruedas y tracción convencional; interesa definir trayectorias de curvatura continua que puedan ejecutarse con el menor error posible (Kambhampati y Davis, 1988; Stenz 1990, citado en Ollero, 2007).

Además de las características geométricas y cinemáticas, puede ser necesario tener en cuenta modelos dinámicos de comportamiento del vehículo contemplando la interacción vehículo-terreno. Por otra parte, puede planearse también el problema de la planificación de la velocidad teniendo en cuenta las características del terreno y del camino que se pretende seguir.

Una vez realizada la planificación de la trayectoria, es necesario planificar movimientos concretos y controlar estos movimientos para mantener al vehículo en la trayectoria planificada. De esta forma, se plantea el problema del seguimiento de caminos, que para vehículos con ruedas se concreta en determinar el ángulo de dirección teniendo en cuenta la posición y orientación actual del vehículo con respecto a la trayectoria que seguir. Asimismo, es necesario resolver el problema del control y regulación de la velocidad del vehículo.

Conviene mencionar también métodos que permiten la integración de la planificación con el control del vehículo. Entre estos cabe mencionar el de los campos potenciales (Kanayama y Hartman, 1989; Nelson, 1989). La idea consiste

en determinar la resultante de fuerzas que atraen el robot hacia el objetivo y que lo repelen de los obstáculos.

4.8.2. Robots autónomos y tele-robótica[10].

De acuerdo con su grado de autonomía, los robots pueden ser clasificados en tele-operados, de funcionamiento repetitivo y autónomo. En los robots tele-operados las tareas de percepción del entorno, planificación y manipulación compleja son realizadas por humanos. Donde el operador se encarga de cerrar el bucle de control.

En los sistemas más evolucionados de robots tele-operados al operador le llega información sensorial del entorno (imágenes, fuerzas, distancias) para la manipulación se emplean brazos y manos antropomórficos con controladores automáticos que reproducen los movimientos del operador, y este mueve una réplica a escala del manipulador, el cual reproduce fielmente los movimientos de este (Hirzinger y otros, 1991).

Estos robots son muy aplicables para trabajos en una localización remota (acceso difícil, medios contaminados o peligrosos), en procesos difíciles de automatizar y en entornos poco estructurados.

Las mayores dificultades en estos sistemas radican en las limitaciones que tiene el hombre para el procesamiento numérico, precisión y en el acoplamiento, coordinación entre el hombre y el robot. En algunas aplicaciones, el retraso de transmisión de la información es algo para tener en cuenta en el diseño del sistema de control. El diseño de la interface persona-maquina generalmente resulta critico. Investigaciones actuales se dirigen a dejar a cargo del operador las tareas que requieren toma de decisiones en función de información sensorial, experiencia, y habilidad, sin embargo existen limitaciones por el ancho de banda de la transmisión

[10] Ollero, 2007.

y por la complejidad de la tarea del operador. Ejemplos de este tipo de robot son los de exploración en la superficie de Marte (Spirit y Opportunity).

Los robots de funcionamiento repetitivo son los más usados en la industria. Trabajan normalmente en tareas predecibles e invariantes, con una limitada percepción del entorno. Son precisos, de alta repetitividad y relativamente rápidos; incrementan la productividad ahorrando al hombre trabajos repetitivos y/o peligrosos.

Robots autónomos.

Los robots autónomos son los más evolucionados desde el punto de vista del procesamiento de información, son capaces de percibir, modelar el entorno, planificar y actuar para alcanzar objetivos sin la intervención, o con una mínima intervención del hombre. Pueden trabajar en entornos poco estructurados y dinámicos, realizando acciones en respuesta a variaciones del entorno.

Durante las últimas décadas se han realizado importantes esfuerzos en la aplicación de técnicas de inteligencia artificial, en donde son empleados métodos simbólicos de tratamiento de la información basados en modelos geométricos del entorno. Las dificultades debido a la elevada capacidad de procesamiento requerida para tratar en tiempo real problemas suficientemente significativos. Sin embargo estos modelos funcionan siempre y cuando este entorno corresponda exactamente a la realidad. La técnica utilizada para reducir esta incertidumbre consiste en incrementar la información que se dispone de dicho entorno mediante realimentación sensorial. Existen métodos que permiten intercalar la formulación y ejecución de planes con la captación de la información necesaria para asegurar que el modelo que se utiliza para la planificación sea lo suficientemente confiable. Las limitaciones son debidas al sistema de percepción y por la propia arquitectura del sistema de información y control del robot.

Desde el punto de vista de la planificación, existen diferentes arquitecturas diseñadas teniendo en cuenta especificaciones sobre el tiempo que tiene el sistema para responder y la disponibilidad de información potencialmente importante.

La solución se sitúa normalmente entre dos extremos, en uno de los cuáles esta la planificación puramente estratégica.

En este caso, se supone que la situación en la que va a ejecutarse el plan puede ser precedida de forma suficientemente precisa durante la planificación. En el otro extremo se sitúa la planificación puramente reactiva en la que se supone que el entorno es incierto, y el robot podrá responder ante variaciones y problemas que se le presenten en el entorno, por medio de discrepancias entre el modelo actual y lo observado o percibido por el sistema sensorial.

El problema puede plantearse también en términos de compromiso entre eficiencia y flexibilidad, es decir llegar a un equilibrio entre ambos aspectos y responder a las necesidades actuales aprovechando al máximo lo mejor de las herramientas técnicas. Un ejemplo de esto es el caso de las arquitecturas diseñadas para conseguir la mayor flexibilidad ante cualquier eventualidad del entorno resultan muchos menos eficientes que las que utilizan criterios de decisión basados en modelos del entorno suficientemente precisos. En este punto conviene mezclar ambas técnicas en una planificación estratégica basadas en técnicas de búsqueda, con la planificación puramente reactiva.

4.9. Robótica pedagógica.

"*Martial Vivet (citado en Ruiz-Velasco, 2008) define la robótica pedagógica como la actividad de concepción, creación y puesta en práctica, con fines pedagógicos de objetos tecnológicos que son reducciones fieles y significativas de procedimientos y herramientas robóticas bastante usadas en la vida cotidiana, de forma especial en el medio industrial*".

El principal objetivo de la robótica pedagógica es la creación de entornos de aprendizaje teniendo en cuenta la actividad del estudiante. Por medio de ésta, le es posible al estudiante comprender, crear y poner en práctica diferentes robots pedagógicos que le permiten encontrar la solución de uno o más problemas, y al mismo tiempo adquiere cierto aprendizaje. De esta manera el estudiante se apropia del conocimiento y hay una interacción con otros campos del conocimiento.

Teniendo en cuenta lo anterior se puede observar que la robótica pedagógica puede servir como herramienta para brindar soluciones a diferentes aéreas del conocimiento como las matemáticas, las ciencias naturales, la tecnología y la ciencias de la información y la comunicación.

La robótica pedagógica privilegia la experimentación-investigación y desarrollo para crear micro-mundos (robots). El objeto del micro-mundo es permitir al estudiante adquirir conocimientos de un determinado dispositivo o comprender un fenómeno.

4.9.1. Bondades cognoscitivas de la robótica pedagógica.

Algunas de las principales bondades cognoscitivas son:

Integración de distintas áreas del conocimiento.

En la construcción de un robot pedagógico se requiere tener conocimientos en mecánica para la construcción de la estructura, de electricidad para animar desde el punto de vista eléctrico al robot. Se debe tener cierto conocimiento en electrónica, para diseñar la interfaz de comunicación entre el computador (si este es el caso) y el robot pedagógico y por último es los conocimientos en informática son obligatorios para la creación del programa en cualquier lenguaje de programación, con el cual se puede controlar el robot. Es aquí donde se muestra la principal bondad que tiene la robótica pedagógica al integrar distintas áreas del conocimiento. Se piensa que a través del diseño y construcción de un robot pedagógico pueden interactuar estas áreas del conocimiento desde el punto de vista cognitivo y tecnológico.

Operación con objetos manipulables, favoreciendo el paso de lo concreto a lo abstracto.

En la medida que el sujeto manipule y adquiera experiencia en el manejo del robot pedagógico, éste se va favoreciendo en el desarrollo de estrategias para resolver problemas y al mismo tiempo su conocimiento se enriquece y es más significativo. Esto se logra mediante la acción del sujeto sobre los objetos. A partir de estas

acciones se llega a resultados u objetivos comunes y por abstracción reflexiva, construir conceptos también comunes.

Apropiación por parte de los estudiantes de distintos lenguajes (gráfico, icónico, matemático, natural, etc.), como si se tratara de lenguaje matemático.

Otras de las bondades que ofrece la robótica pedagógica es la coincidencia, entre un fenómeno de la vida real que se está produciendo y su simulación gráfica a través de la pantalla del computador. Con esto se obtiene la seguridad de que los estudiantes se apropien del lenguaje gráfico como si se tratara del lenguaje matemático. Desde el punto de vista cognitivo tiene la ventaja de la fácil lectura, memorización e interpretación de una gráfica normal, que el aprendizaje memorizado de números y ecuaciones que muchas veces nos cuesta interpretar o retener, debido a que la memoria de corto plazo de la mayoría de los humanos es limitada.

Operación y control de distintas variables de manera síncrona.

Aunque miniaturizada, la robótica pedagógica permite operar y manejar diversas variables al mismo tiempo. Una vez que el fenómeno de estudio sea puesto en marcha a través del robot pedagógico, se inicia el control de las variables que están interactuando en forma simultánea. Una primera visualización se realiza cuando el fenómeno está en marcha y una segunda visualización se realiza en los datos o gráficos arrojados por el computador, cuando el fenómeno está siendo estudiado. Es aquí cuando es posible visualizar física y mentalmente las diversas variables.

El desarrollo de un pensamiento sistémico y sistemático.

Mediante la robótica pedagógica el estudiante logra desarrollar sus estructuras cognitivas de manera más completa. En el desarrollo de una actividad en robótica educativa para resolver una problemática determinada, mediante un robot, exige al estudiante tener en cuenta un sistema formado por variables que interactúan entre sí. Es decir, que al modificar una las otras se alteran.

Construcción y prueba de sus propias estrategias de adquisición de conocimiento mediante una orientación pedagógica.

La robótica pedagógica brinda al estudiante la posibilidad de construir sus propias estrategias para la adquisición del conocimiento mediante una orientación educativa permitiéndole ser un receptor activo de conceptos. Esta posibilidad se inicia teniendo en cuenta la actividad a realizar por el estudiante cuando éste interactúa con los objetos de conocimiento.

Creación de entornos de aprendizaje.

La creación de entornos de aprendizaje por medio de la robótica pedagógica se ha convertido en una herramienta poderosa desde el ámbito cognitivo, al permitir mejores condiciones para la apropiación del conocimiento. A través de un entorno creado se puede observar, explorar y reproducir fenómenos precisos y reales; favorece la relación estudiante, computadora, robot y educador. Además, mediante el entorno creado, el estudiante puede confrontarse inmediatamente con el error en caso que exista. Con dichos entornos se puede simular el proceso cuantas veces crea necesario el estudiante para comprobar resultados.

Creación de un ambiente de aprendizaje lúdico y heurístico.

La robótica pedagógica permite crear un ambiente real de aprendizaje lúdico y heurístico, aprovechando y potencializando al máximo las tecnologías de la información, la comunicación y conocimiento durante el diseño y construcción del robot. Es así como la robótica pedagógica se convierte en un actuar tecnológico, en un saber comunicar y en un saber hacer. En un principio el estudiante no sabe cómo construir su robot; es en éste punto donde se crea el ambiente heurístico, obligándolo a investigar e indagar para construir su robot. De esta manera el estudiante poco a poco va afinando su capacidad de descubrimiento e inventando el camino para concebir, crear y poner en marcha su robot pedagógico.

4.9.2. Fases para el diseño y construcción de un robot pedagógico.

Una hipótesis de la robótica educativa es probar si los estudiantes pueden crear sus propios conceptos de ciencia y tecnología de base, mediante la interacción con un entorno robotizado, al tiempo que resuelven problemas. Los estudiantes aprenden a armar robots pedagógicos utilizando: LEGO, Robotix, Fishertechnic, Hero, Mecanos, etc.; o materiales reciclables como madera cartón, acrílico, etc.

Para la construcción de un robot pedagógico se requiere de cuatro fases o etapas pedagógicas:

- Fase mecánica. En esta fase los estudiantes construirán la estructura del robot educativo y aprenderán los conceptos de engranajes, poleas, ejes articulaciones, grados de libertad, motor, corriente, voltaje, movilidad, etc.

- Fase eléctrica. Esta fase consiste en animar el robot desde el punto de vista eléctrico. En esta etapa el estudiante aprenderá concepto tales como; los accionamientos finales que son los encargados del movimiento del robot. Para esto tendrá que saber que existen diferentes tipos de motores (de corriente continua, de corriente alterna, de paso, servomotores, hidráulicos, etc.).

- Fase electrónica. En esta fase los estudiantes aprenderán sobre los sensores y su clasificación. Se percatarán que por medio de los sensores es como el robot pedagógico puede interactuar con el entorno de trabajo. Los sensores pueden ser análogos o digitales, dependiendo del prototipo.

- Fase informática. En esta última fase los estudiantes deberán aprender que existe una interfaz de comunicación entre el robot pedagógico y el computador, que es la encargada de transferir los programas al controlador.

4.10.Herramientas para la construcción de robots pedagógicos[11].

En las últimas décadas investigadores e industrias han desarrollado kits para la construcción de robots, diseñados para participar en los procesos de enseñanza-aprendizaje. Los kits incluyen, pequeños motores, sencillos sensores, ruedas, engranajes, poleas y relés – todo lo que el estudiante necesite para construir robots. Productos como; LEGO Dacta y LEGO CyberMaster incluyen el equipamiento que le posibilita conectar el robot con un computador personal; esto permite al usuario, ya sea estudiante o educador, controlar el dispositivo robótico.

Hace unos pocos años LEGO lanzó al mercado LEGO Mindstorms, él provee la posibilidad al usuario de construir robots autónomos, con el control localizado al interior de la máquina.

Estos kits se han desarrollado teniendo en cuenta los principios educativos de las teorías del desarrollo cognitivo de Piaget (1966). Estas teorías hablan de una participación activa por parte del sujeto para adquirir el conocimiento a través de la manipulación y construcción de objetos.

4.11.Robots LEGO Mindstorms NXT.

El paquete Lego Mindstorms Education NXT para construir y programar robots ofrece una gran versatilidad, desde ser un juguete para niños, a todo un sistema de desarrollo de robots que es utilizado por millones de aficionados a la robótica y la ingeniería. Su bajo costo y facilidad para construir diferentes modelos de robots y la variedad de lenguajes de programación. Disponibles los hacen muy apropiados para gente que se inicia en robótica y hasta para investigaciones serias en robótica e inteligencia artificial.

[11] Miglino, 1999. p. 2.

Las principales partes para el diseño y construcción de un robot LEGO Mindstorms NXT son:

El Ladrillo Inteligente NXT[12].

El ladrillo NXT es el "cerebro" de cualquier robot realizado con lego, puesto que se encarga de controlar mediante la ejecución de un programa, todas las tareas para las que haya sido diseñado.

Una característica de este ladrillo es que una vez cargado el programa, no se requiere que el ladrillo se encuentre conectado al PC, puede ser ejecutado directamente desde el ladrillo.

A continuación una lista de especificaciones electrónicas para el ladrillo NXT:

Procesador principal	Atmel® 32-bit ARM7® processor, AT91SAM7S256
	256 KB FLASH
	64 KB de RAM
	48 MHz
Co-procesador	Atmel® 8-bit AVR processor, ATmega48
	4 KB FLASH
	512 Byte RAM
	8 MHz
Comunicación inalámbrica Bluetooth	CSR BlueCoreTM 4 v2.0 +EDR System
	Soporte de Serial Port Profile (SPP)
	47 KByte RAM Interna
	8 MBit FLASH Externa, 26 MHz
Comunicación por USB	2.0 velocidad (12 Mbit/s)
4 puertos de entrada	Interfaz de 6 hilos que soporta tanto A/D como D/A
	1 puerto de alta velocidad, IEC 61158 Tipo 4/EN 50170 compatible
3 puertos de salida	Interfaz de 6 hilos que soporta tanto A/D como D/A
Display grafico LCD	100x64 píxel, blanco y negro
	área de visión: 26x40.6mm
Altavoz	Canal de salida de 8-bits de resolución.
	Soporta una frecuencia de muestreo de 2-16 KHz
4 botones de interfaz usuario	Botones de goma
Fuente de energía	6 baterías 1.5 v, AA
	Se recomienda usar baterías alcalinas
	También se encuentra una batería recargable de Ion de Litio de 1400 mAH
Conectores	6 cables conectores de estándar industrial
	RJ12 con ajuste derecho

Tabla 1. Lista de especificaciones técnicas para el ladrillo NXT

Visión General Técnica del NXT.

[12] LEGO® MINDSTORMS®. (2009).

Esta sección proporciona una visión general gráfica de cómo las distintas funciones están conectadas y controladas en el ladrillo inteligente. La figura sólo incluye las unidades de más alto nivel en el NXT. Para obtener información detallada sobre cómo se conectan los elementos individuales, vea el esquema de hardware en el (ANEXO E).

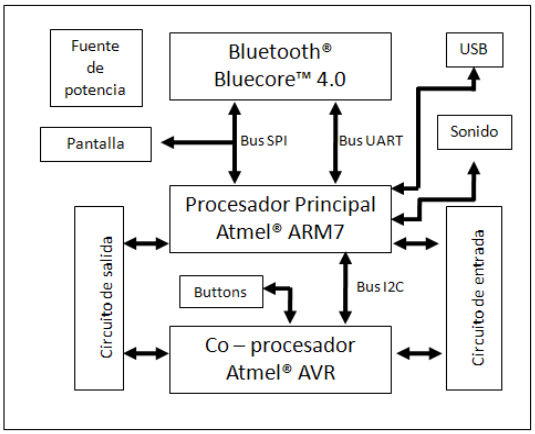


Figura 20. Diagrama de bloques del NXT

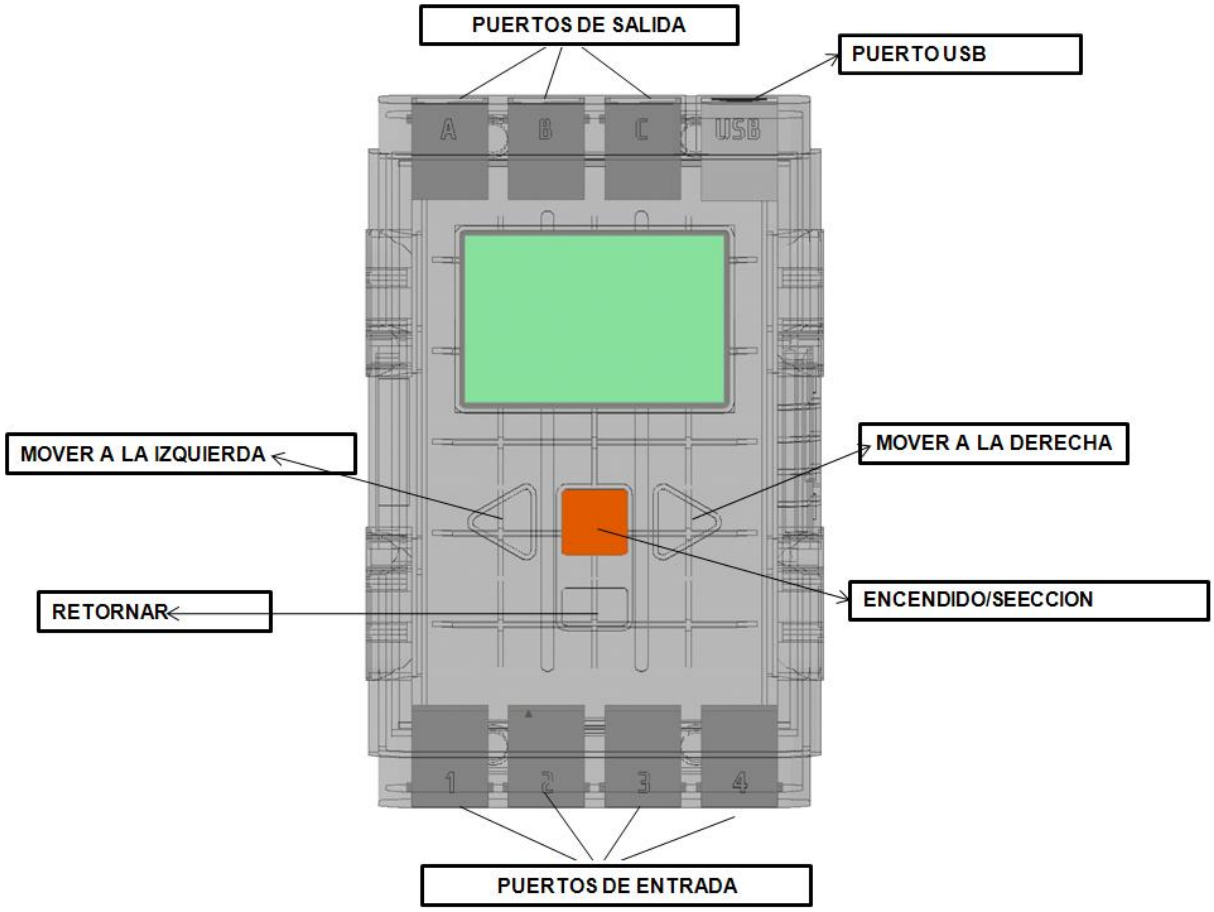


Figura 21. Esquema físico del ladrillo NXT

Puertos de salida del NXT.

LEGO MINDSTORMS NXT cuenta con tres puertos de salida utilizados para controlar los accionamientos finales conectados al ladrillo NXT.

Una interfaz digital de usuario de 6 cables en los puertos de salida se implementó con el fin de que los dispositivos de salida pudieran enviar información de vuelta al ladrillo NXT sin tener que utilizar un puerto de entrada nuevamente.

La figura siguiente muestra un esquema de detalles del puerto A del ladrillo. Los esquemas de los puertos B y C son idénticos.

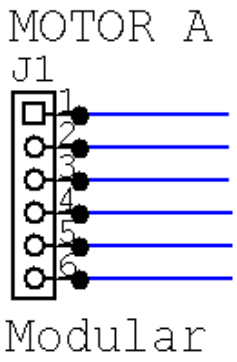


Figura 22. Esquema eléctrico del puerto A

Pin 1, MA0	Señal de salida PWM para los accionamientos finales
Pin 2, MA1	Señal de salida PWM para los accionamientos finales
Pin 3, GND	Señal de tierra relacionada con el suministro de salida
Pin 4, POWERMA	Suministra 4.3voltios a la salida
Pin 5, TACHOA0	Valor de la entrada que incluye la funcionalidad del Schmitt trigger
Pin 6, TACHOA1	Valor de la entrada que incluye la funcionalidad del Schmitt trigger.

Tabla 2. Configuración de pines de los puertos de salida del NXT

MA0 y MA1 son señales de salida para el control de accionamientos finales. Estas señales son controladas por un driver interno del motor que puede suministrar 700 mA, a cada puerto de salida y una corriente máxima de aproximadamente 1 A. La señal de salida es una señal PWM. El driver del motor tiene protección térmica incorporada, lo que significa que si se extrae demasiada potencia continuamente del ladrillo, el driver del motor se ajustará automáticamente a la corriente de salida.

La potencia de salida (POWERMA) está conectada internamente a todas las salidas de potencia en los puertos de salida y entrada. La corriente de salida máxima que se puede extraer de este suministro es de aproximadamente 180 mA. Esto significa que cada puerto tiene aproximadamente 20 mA. Si se extrae más potencia, el total de corriente se disminuirá automáticamente sin advertencia. Si la señal de la potencia hace cortocircuito a tierra, el ladrillo NXT se reiniciará.

El TACHOA0 y TACHOA1 son puertos de entrada que tienen un circuito Schmitt trigger (circuito integrado utilizado para tratar señales con ruido), montado entre los puertos de entrada y pines en el procesador ARM7. Estas dos señales posibilitan tener un detector de cuadratura en el sistema. En el nivel de firmware estas dos señales se utilizan para contar el número de pulsos del tacómetro de los motores y detectar si el motor está funcionando en el sentido de las agujas del reloj o en sentido contrario.

Puertos de entrada del NXT.

LEGO MINDSTORMS NXT cuenta con cuatro puertos de entrada que permiten que el NXT mida diversos parámetros en el mundo físico (en función del sensor conectado).

La interfaz digital de usuario de 6 cables en los puertos de entrada permite tener tanto una interfaz A/D y D/A en cada conector. Esto permite la posibilidad del desarrollo de los sensores analógicos y digitales para el ladrillo NXT.

La figura siguiente muestra algunos detalles del esquema detrás del puerto 1 en el ladrillo. Los puertos 2, 3, y 4 tienen los mismos esquemas. Detrás del puerto 4, los pines digitales (DIGIxI0 y DIGIxI1) están conectados a un controlador RS485 que maneja comunicación de alta velocidad.

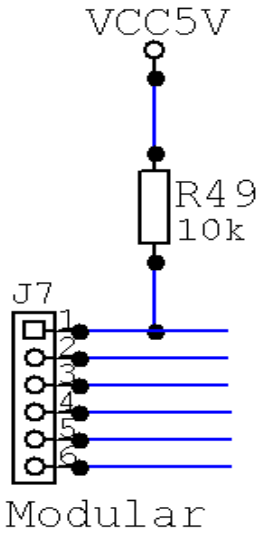


Figura 23. Esquema de los puertos de entrada

Pin 1, ANA	Entrada analógica y posible señal de salida de corriente
Pin 2, GND	Tierra
Pin 3, GND	Tierra
Pin 4, IPOWERA	Suministra 4.3voltios a la salida
Pin 5, DIGIAI0	E/S digital pin conectado con el procesador ARM7
Pin 6, DIGIAI1	E/S digital pin conectado con el procesador ARM7

Tabla 3. Configuración de pines de los puertos de entrada del NXT

El pin de entrada (ANA), es un pin de entrada analógica que está conectado a un convertidor A / D de 10 bits en el procesador AVR. Este está también conectado al generador de corriente que se utiliza para la generación de energía para los sensores activos del sistema de invención robótico de LEGO MINDSTORMS. (Vea la sección acerca de los sensores activos en este documento). Las señales de entrada A / D están incluidos en la muestra con la misma frecuencia de muestreo para todos los sensores analógicos. Como se describe en la documentación del sensor activo, los sensores analógicos necesitan de 3 ms de suministro de potencia de salida antes de que se de cualquier medición. La tasa de muestreo utilizado para todos los sensores analógicos es 333 Hz.

La Potencia de salida (IPOWERA) está conectada internamente a todas las salidas de potencia, en los puertos de salida y entrada. La corriente de salida máxima que se puede extraer de este suministro es de aproximadamente 180 mA. Esto significa que cada puerto tiene aproximadamente 20 mA disponible. Si se extrae más energía, la corriente total de salida se disminuirá automáticamente sin más advertencia. Si la señal de la potencia hace cortocircuito a tierra, el ladrillo NXT se reinicia.

Los pines digitales I / O (DIGIAI0 y DIGIAI1) se utilizan para la comunicación digital aplicada como comunicación I2C compatible, funcionando a 9600 bits / s. El NXT sólo puede funcionar como un maestro en relación con la comunicación I2C y exige que los dispositivos externos tengan alta resistencias incluida en su pines de comunicación. Véase en el capítulo de Comunicación I2C para más detalles.

Además, el pin I / O en el ARM7 que está conectado a DIGIXI1 se puede configurar para que funcione como un pin de entrada analógica (aunque esto no está soportado directamente en el nivel de firmware). Esto permite la posibilidad de implementar un pin de entrada analógica con una mayor frecuencia de muestreo.

Sensores de tipo activos en el NXT.

Para garantizar la compatibilidad mutua con LEGO MINDSTORMS Robotic Invention System sensors (para el ladrillo RCX inteligente), un generador de corriente se ha añadido al ladrillo NXT para obtener el poder y la medición correcta de intervalos en estos sensores más antiguos. Junto con el estándar actual de firmware de LEGO el generador de corriente da la misma funcionalidad que está disponible en el ladrillo RCX inteligente. El generador de corriente proporciona aproximadamente 18 mA de salida de corriente.

El generador controla la entrega de energía a los sensores activos. Suministra al sensor con potencia de 3 ms y entonces mide el valor analógico durante los siguientes 0,1 ms.

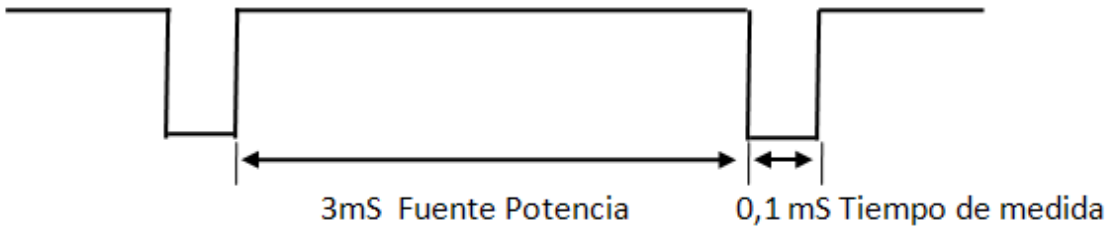


Figura 24. Diagrama de tiempo para el pin entrada de la señal de entrada A/D cuando se usan sensores activos

Los siguientes sensores del sistema de invención robótico de LEGO MINDSTORMS. Son sensores activos:

- Sensor de luz.
- Sensor de rotación.

Sensores de tipo pasivos en el NXT.

Todos los sensores que no necesitan la sincronización especial de potencia / medición antes mencionadas son denominados sensores pasivos. Estos sensores están también incluidos en la muestra cada 3 ms, porque el muestreo usando el convertidor A / D se realiza de forma simultánea y, por tanto, debe respetar los plazos requeridos por los sensores activos.

Los siguientes son sensores pasivos:

- Sensor de contacto (tanto las versiones RCX y NXT)
- Sensor de luz ofrecido en el set de LEGO MINDSTORMS NXT
- Sensor de sonido
- Sensor de temperatura.

Sensores de tipo digital en NXT.

Todos los sensores que utilizan la comunicación I2C se denominan sensores digitales, debido a que incluyen un micro-controlador externo que se encarga de la toma de muestras del medio físico.

Los siguientes son sensores digitales:

- Sensor ultrasónico

Puerto de comunicación de alta velocidad del NXT.

El puerto 4 en el ladrillo inteligente NXT puede funcionar como un puerto de comunicación de alta velocidad. Un chip de comunicación RS485 se implementa detrás del circuito normal de entrada. Esto permite la implementación de comunicación de datos a alta velocidad bidireccional sobre una línea de datos multipunto en distancias más amplias. Actualmente LEGO no ha desarrollado ningún dispositivo que necesite esta funcionalidad en la comunicación. Sin embargo, si en el futuro se desarrollan dispositivos que requieren una mayor

velocidad de comunicación, el ladrillo NXT está preparado. Para este tipo de dispositivos futuros, LEGO puede utilizar el protocolo de comunicación P-Net, que permite la comunicación de datos multipunto.

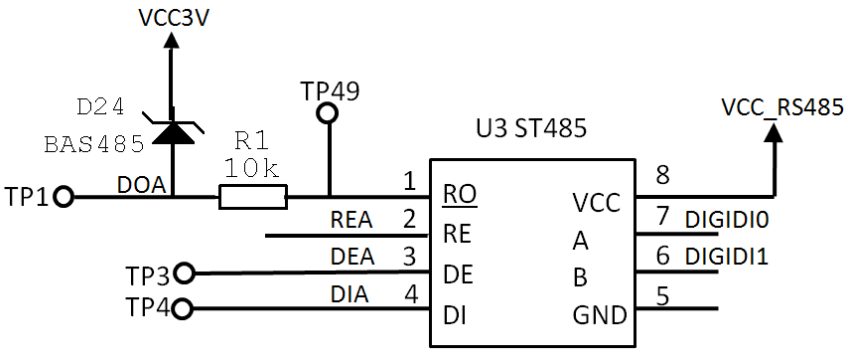


Figura 25. Esquema del Hardware para el chip RS485; puerto 4 del ladrillo NXT

Nota: El chip RS485 utiliza 5 voltios como voltaje de alimentación y el procesador ARM7 utiliza 3,3 voltios. Por esta razón, un nivel de cambio se ha aplicado entre el RS485 y el chip procesador ARM7. Véase el esquema para más detalles.

Los siguientes parámetros de comunicación se establecen para la comunicación de alta velocidad a nivel de firmware:

Velocidad de comunicación	921,6 Kbit / s
Bits de datos	8 bits
Bit de parada	1 bit
Paridad	0 bits

Tabla 4. Parámetros para la comunicación de alta velocidad a nivel de firmware

Comunicación I2C en el NXT.

En el ladrillo NXT, una interfaz digital se ha implementado mediante el protocolo I2C, El cual es un estándar de comunicación industrial que fue desarrollado por

Philips Semiconductors a principios de 1980. Desde entonces ha sido utilizado en diferentes componentes industriales donde la comunicación digital simple es obligatoria.

La Comunicación I2C funciona como la interfaz digital para dispositivos externos que necesita para comunicarse con el NXT. Tener una interfaz digital permite a los dispositivos externos demostrar su funcionalidad por separado y sólo enviar el resultado al NXT o recibir nueva información del NXT.

El ladrillo NXT I2C tiene cuatro canales de comunicación, uno para cada uno de los cuatro puertos de entrada. La comunicación digital I2C se aplica como funcionalidad "únicamente maestro", lo que significa que el NXT está controlando el flujo de datos en cada uno de los canales de comunicación.

Un aspecto importante que permite la comunicación I2C, entre dos dispositivos es la configuración del hardware, dentro de cada uno de los dispositivos. La figura siguiente muestra el esquema interno de hardware para el puerto de entrada 1 en el NXT. El esquema para los puertos de entrada 2, 3 y 4 son los mismos con respecto a la comunicación I2C.

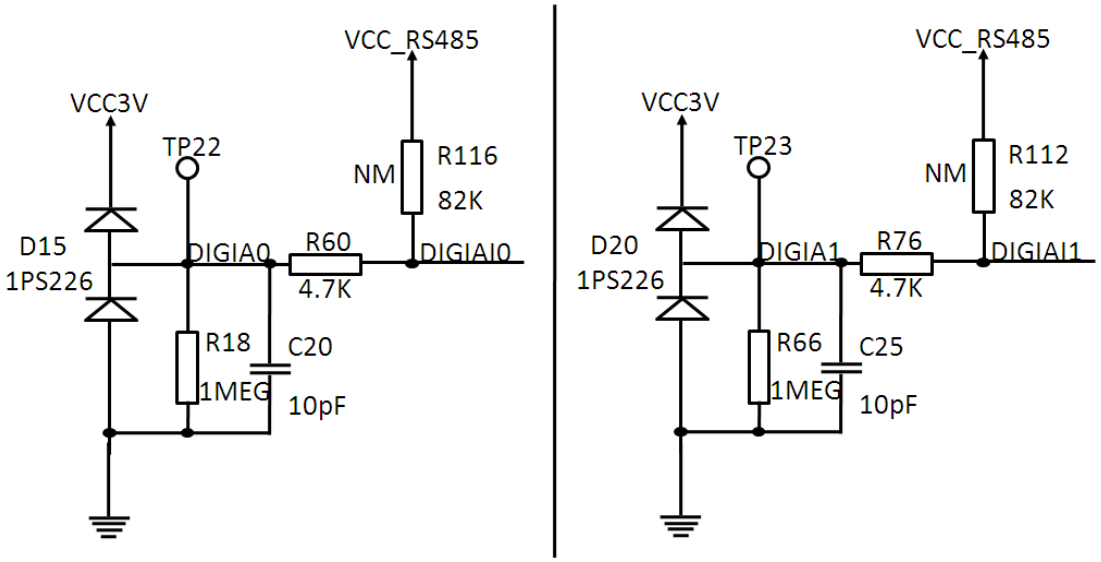


Figura 26. Esquema de comunicación I2C

Parámetros importantes a destacar:

- Hay una resistencia de 4,7 k en serie con la línea de la señal.
- No hay alta resistencia internamente en el NXT. Se debe usar en el dispositivo externo. Recomendamos el uso de resistencias de 82 K para altas resistencias tanto en las líneas de datos como en las líneas del reloj.
- DIGIx0 (Pin 5 en el conector) es la señal de CLK y DIGIx1 (6 pines en el conector) es la señal de DATOS para la comunicación I2C.
- Los pines digitales E/S en el NXT no se puede configurar para abrirse directamente. Por lo tanto, el NXT controlara los pines digitales I / O altos o bajos en función de la situación. Cuando el NXT no debe controlar la líneas de I / O, se definirán como entrada (por ejemplo, al leer los datos desde un dispositivo de lectura o de reconocimiento de lectura).
- La comunicación I2C está funcionando a 9600 bits / s.
- Cada canal tiene un buffer de 16 bytes de entrada y salida. Por lo tanto, un máximo de 16 bytes puede ser enviado y recibido durante cada ciclo de comunicación de datos.
- Si se conectan varios sensores en secuencia con el mismo puerto del sensor, la siguiente resistencia debe ser de 82 k. Por esta razón, es necesario considerar cuando se conectan varios sensores en secuencia para el mismo puerto.

Los dispositivos digitales tienen algunas ventajas en comparación con los dispositivos analógicos. Los dispositivos digitales pueden incluir los nombres del dispositivo y pueden referenciar distintos parámetros individuales que son específicos del dispositivo (como valores de calibración, los tiempos de inicio, y así sucesivamente). Para distinguir los diferentes dispositivos digitales entre sí, LEGO ha comenzado un plan para sus sensores que se ampliará en la medida en que la compañía desarrolle nuevos dispositivos digitales o apruebe los dispositivos de

terceros. Actualmente, la lista de candidatos incluye sólo el sensor Ultrasónico, que ha obtenido dirección 1 (dentro de un contexto de 7 bits). Esta dirección se envía junto con el bit de dirección de comunicación, y como con todos los demás datos de bytes, la dirección se transfiere en primer lugar con el bit más significativo.

Display en el NXT.

Una pantalla de matriz de puntos se ha añadido al ladrillo NXT para mejorar la interfaz de usuario. Esta es una pantalla en blanco y negro, con pantalla LCD gráfica de una resolución de 100 x 64 píxeles. La pantalla tiene una superficie de visión de 26 x 40,6 mm. El controlador LCD utilizado para controlar la pantalla es un UltraChip 1601.

Hay una interfaz SPI desde el micro-controlador ARM7 al UltraChip 1601 controlador LCD de la pantalla. La interfaz SPI se está ejecutando a 2 MHz en el firmware de LEGO estándar y dos mapas de memoria se retiran del firmware para actualizar la pantalla. La pantalla se actualiza de forma continua en una secuencia de línea que requiere 17 ms para una actualización total de la pantalla.

Especificaciones técnicas de la pantalla:

Formato	100 x 64 puntos
Modo LCD	STN /modo positivos reflectantes / Gris
Visualización de la dirección	6 horas
Sistema de conducción	1/65 dutv cycle, 1/9 bias
Tensión de alimentación (Vdd)	3.0V
Tensión de conducción LCD (VLCD)	9.0V (ajustable para mejor contraste)

Tabla 5. Especificaciones técnicas de la pantalla

Bluetooth en el NXT.

El ladrillo NXT soporta la comunicación inalámbrica Bluetooth, al incluir un chip de la CSR BlueCoreTM 4 versión 2. El ladrillo NXT puede conectarse sin cables a otros tres dispositivos al mismo tiempo, pero sólo puede comunicarse con un dispositivo a la vez. Esta funcionalidad se ha implementado utilizando el perfil de puerto serie (SPP), que puede ser considerado como un puerto de serie inalámbrico. El ladrillo

NXT puede comunicarse con los dispositivos Bluetooth que puede ser programado para comunicarse utilizando el Protocolo de Comunicación LEGO MINDSTORMS NXT y que resista el perfil de puerto serie (SPP). Es posible enviar programas y archivos de sonido entre ladrillos NXT y usar comunicación inalámbrica para enviar y recibir información entre ladrillos durante la ejecución del programa. Para reducir el consumo de energía utilizada por la tecnología Bluetooth, la tecnología que se ha aplicado es la de dispositivo Bluetooth Clase II, lo que significa que se puede comunicar hasta una distancia de aproximadamente 10 metros.

La Funcionalidad del Bluetooth dentro del Ladrillo NXT.

La funcionalidad del Bluetooth en el ladrillo NXT se configura como un canal de comunicación maestro/ esclavo. Esto significa que un NXT dentro de la red tiene que funcionar como unidad maestra para que los otros ladrillos NXT se comuniquen a través de él en caso de que lo necesiten. La figura siguiente muestra cuales dispositivos de NXT pueden comunicarse directamente dentro una red.

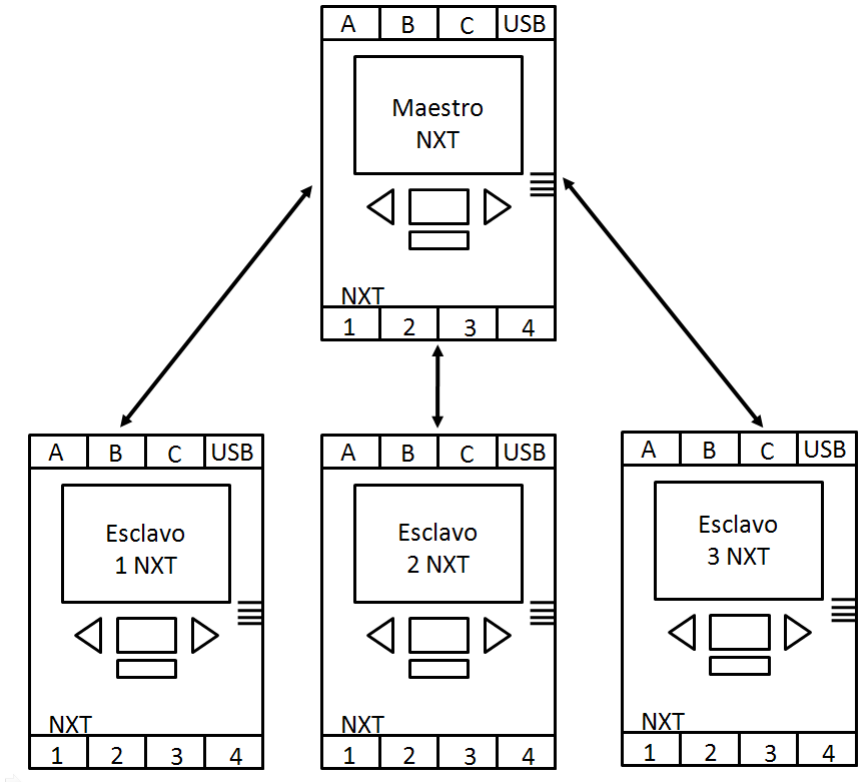


Figura 27. Comunicación de los NXT's usando Bluetooth

Como se muestra en la figura 27, el NXT maestro se puede conectar a otros tres dispositivos de Bluetooth al mismo tiempo. El NXT maestro puede comunicarse con una de las unidades esclavos durante un determinado momento, lo que significa que si el NXT maestro se comunica con NXT esclavo 1 y NXT esclavo 3 inicia el envío de datos al NXT maestro, el NXT maestro no evaluará los datos recibidos hasta que no se cambie a NXT esclavo 3.

Un NXT no puede funcionar como un dispositivo maestro y esclavo al mismo tiempo porque esto podría causar la pérdida de datos entre dispositivos de NXT.

Las conexiones a otros dispositivos Bluetooth se producen a través de canales. El NXT tiene cuatro canales de conexión utilizados para la comunicación Bluetooth. El canal 0 es siempre utilizado por los dispositivos NXT esclavos cuando se comunican con el NXT maestro (es decir, hacia el NXT maestro), mientras que los canales 1, 2 y 3 se utilizan para la comunicación saliente desde el dispositivo maestro a los dispositivos esclavos.

Sonido en el NXT.

El ladrillo NXT incluye un chip de sonido amplificador para mejorar la calidad y el nivel de salida del sonido. La salida de sonido es una señal de PWM que es controlada por el micro-controlador ARM7. Los filtros introducidos antes del amplificador reducirán el exceso de ruido en la señal.

El controlador de sonido (SPY0030A) es un chip amplificador de sonido diferencial de SUNPLUS que puede tener una ganancia máxima de 20.

El altavoz en el NXT es uno 16 ohmios con un diámetro de 21 mm. La tabla de abajo muestra el consumo de energía y de corriente cuando los sonidos se reproducen en dos frecuencias diferentes.

Frecuencia	corriente mA	potencia mW
440 Hz	102	169
4 kHz	78	97

Tabla 6. Consumo de corriente del sonido

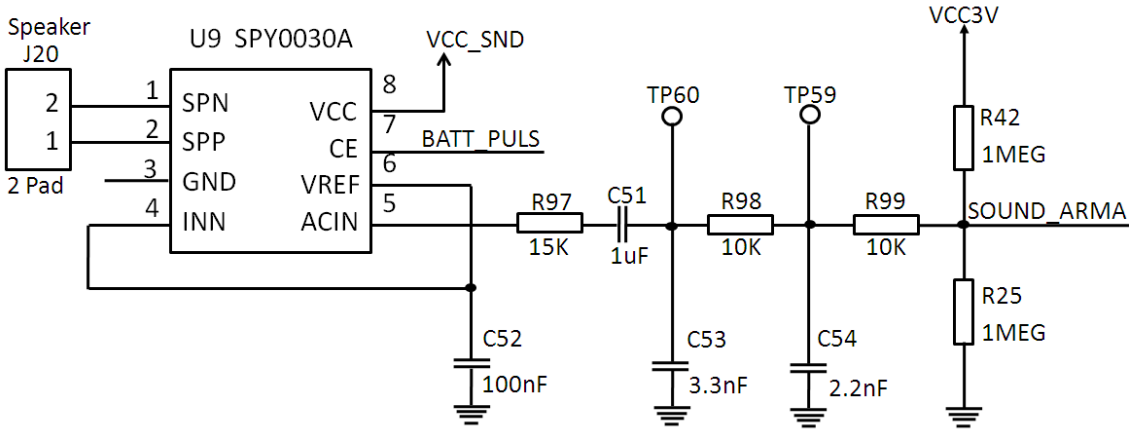


Figura 28. Esquema para la salida del sonido en el NXT

Administración de energía en el NXT.

La Potencia en el ladrillo NXT viene de 6 baterías AA o una batería recargable de iones de litio (que también incluye un conector de alimentación de AC al Transformador LEGO). La administración de energía en el ladrillo NXT consiste en una fuente de alimentación conmutada, lo que genera 5-voltios de las baterías y de los 5-voltios, otros 3,3-voltios se generan para el procesador ARM7 y el chip BlueCore.

Para proteger la fuente de alimentación en el NXT, un interruptor de plástico está conectado al inicio del circuito de alimentación. El interruptor de plástico tiene una corriente de espera 1,85A y se desencadena aproximadamente a 3,3 A.

El rendimiento del ladrillo NXT depende de las baterías utilizadas y de la carga aplicada al ladrillo.

Medición de la corriente de consumo:

Los efectos se basan en un voltaje de la batería de 9 voltios.

Voltaje de alimentación (Voltios)	Corriente (mA)		Efecto (Batería = 9V)	
	Máxima	Normal	Máximo	Normal
Motores sin carga				
9.0	339	114	5184	1422
5.0	271	112	1744	448
3.3	72	38	410	216
Motores con carga				
9.0	2901	848	26109	7632
5.0	271	112	1142	307
3.3	72	38	410	137
En espera	46 uA se asume el modo de espera debido a la falta de detección de carga.			

Tabla 7. Medición de la corriente en el ladrillo MINDSTORMS NXT

4.11.1. Sensores.

Con los sensores, los robots construidos pueden percibir el ambiente, de modo que puedan reaccionar a la luz, temperatura, el sonido, el movimiento o contacto con un objeto. Cuando se construyen modelos basados en una guía, es importante conectar los sensores a los puertos mostrados en las instrucciones de construcción.

Los sensores con los que cuenta el Robots LEGO MINDSTORMS NXT son:

Sensor de contacto NXT.

Este sensor le da al robot móvil el sentido del tacto. Internamente en este sensor encontramos; un pulsador, un conector, una resistencia y la placa de circuito impreso. Ver conexiones del circuito (ANEXO E).

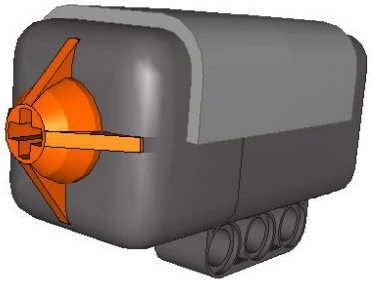


Figura 29. Sensor de contacto NXT

Sensor de sonido NXT.

Este sensor se utiliza para detectar el nivel de sonido y no ningún tipo de tono o modulación.

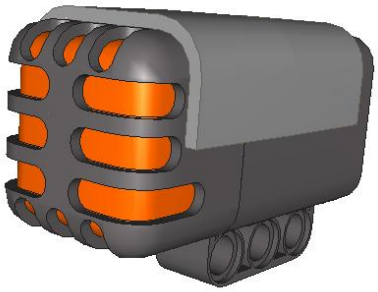


Figura 30. Sensor de sonido del NXT

Las unidades de medición de intensidad del sonido son en (dB), y se presenta en una escala logarítmica. El sensor mide la intensidad del sonido hasta 90dB Figura 24.

Además se puede configurar para que lea decibeles (dB) o decibeles ajustados (dBA). Los decibeles ajustados incluye sonidos que el oído humano puede percibir, mientras que los decibeles (dB) pueden incluir sonidos que no podemos percibir pero pueden ser captadas por el sensor de sonido.

Debido a que los decibeles se miden en una escala logarítmica, el sensor de sonido proporciona la facilidad de obtener valores porcentuales, de más fácil interpretación.

- 4-5% Una sala en silencio.
- 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 un alto volumen.

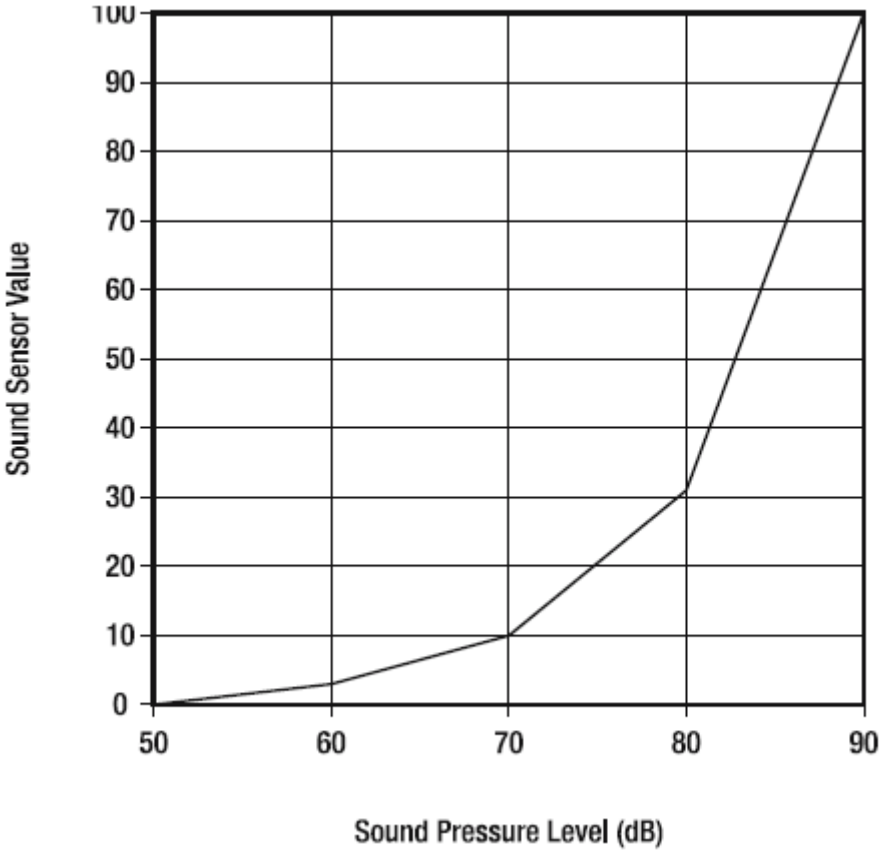


Figura 31. Valor del nivel sonoro Vs nivel de presión sonora[13]

En el sensor de sonido NXT, encontramos internamente un micrófono, montado en una pequeña esponja, un condensador y un conector. Todos montados sobre un circuito impreso. Ver conexiones del circuito (ANEXO E).

Sensor de luz en NXT.

El sensor de luz es uno de los dos sensores que dotan de visión al robot (El sensor ultrasónico es el otro). El sensor de luz permite al robot diferenciar entre la luz y la

[13] Gasperi, 2007. p. 13.

oscuridad. Puede leer la intensidad de la luz en una habitación y medir la intensidad de la luz de las superficies de color.

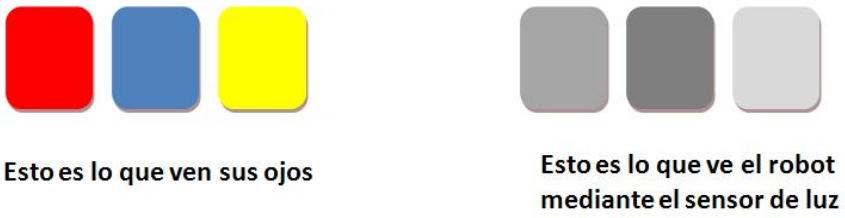


Figura 32. Comparación ojo humano vs. sensor de luz

Con esta característica el sensor puede usarse para que el robot siga una línea o para activarse al detectar la luz, dependiendo de la aplicación. En el (ANEXO E) podemos ver el circuito interno del sensor y sus componentes.

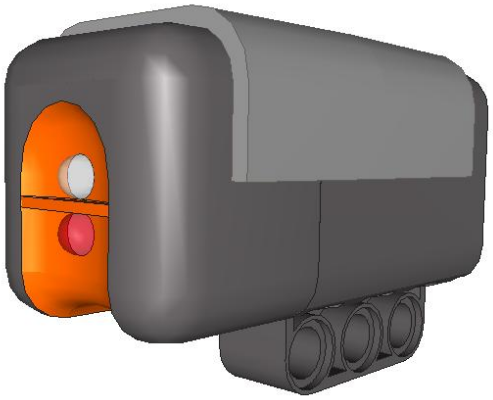


Figura 33. Sensor de luz del NXT

La figura 26, muestra la sensibilidad del sensor NXT en una amplia gama de intensidades de luz, que es normalmente medido en unidades de lux (lx).

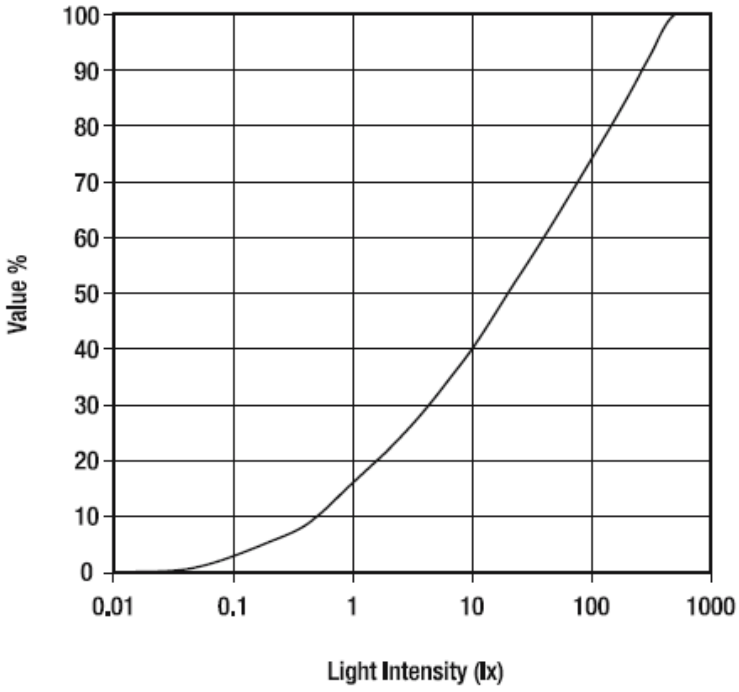


Figura 34. Sensibilidad del sensor de luz[14]

Sensor de ultrasonido.

Este sensor le permite al robot ver, detectar obstáculos y al mismo tiempo medir distancias.

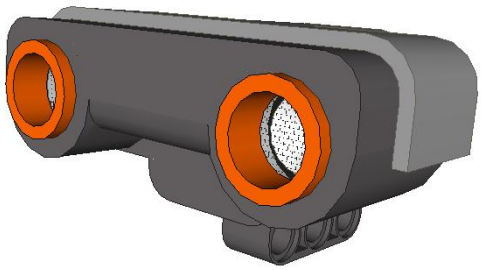


Figura 35. Sensor de ultrasonido del NXT

[14] Gasperi, 2007. p. 11.

Las distancias medidas por este sensor están entre 0 y 255 cm con una precisión de +/- 3 cm. Esto significa que si hay un obstáculo a 40 cm el sensor lo detecta desde 37 hasta 43 cm, este es su margen de error.

El sensor funciona como un sonar. Envía una breve ráfaga de sonido ultrasónico a 40 KHz A continuación el sensor mide el tiempo que tarda el sonido al detectar el objeto, reflexionar y viajar de vuelta al sensor.

Una característica importante de este sensor, es el ver y detectar objetos grandes, como una pared plana, en este caso la medida del sensor es muy buena. Sin embargo si el escenario se complica, con muchos objetos pequeños, su medida no es tan confiable.
En el (ANEXO E) se puede ver el circuito interno del sensor y sus componentes.

Sensor de color.

El sensor de color utiliza una luz diferente a la luz de ambiente, lo que le permite medir los diferentes valores de reflexión de cada color.

El sensor de color funciona iluminando con tres fuentes de luz (Diodos emisores de luz o led), rojo, verde y azul. El NXT recibe tres valores: el nivel de ROJO, el nivel de VERDE y el nivel de AZUL.

El valor correspondiente a cada color está comprendido entre 0 y 255. Por ejemplo, si el valor devuelto es Rojo =255, Verde =0 y Azul =0 el color leído es el Rojo. En la guía de laboratorio número tres encontraremos más ejemplos sobre el uso y funcionamiento de este sensor.

Figura 36. Sensor de color de Hitechnic para LEGO NXT

Giroscopio.

El Gyro NXT contiene un sensor giroscópico de eje único, que detecta la rotación y devuelve un valor que representa el número de grados por segundo de rotación. El Gyro sensor puede medir hasta + / - 360 ° por segundo de rotación. Se conecta a un puerto del NXT mediante un cable estándar NXT y utiliza la interfaz analógica del sensor. La tasa de rotación se puede leer hasta aproximadamente 300 veces por segundo.

El Gyro sensor se encuentra dentro del encapsulado estándar de Mindstorms NXT de lego para que coincida con el de otros elementos Mindstorms.

El eje de medición está en el plano vertical con la posición del sensor giroscopio que se muestra en la figura 29

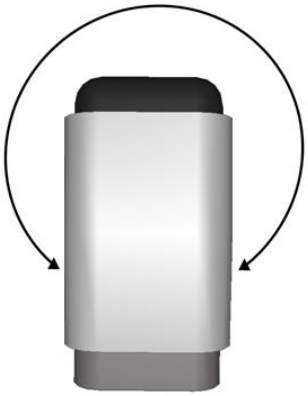


Figura 37. Sensor giroscópico de Hitechnic para LEGO NXT

Compás.

El NXT contiene un sensor de brújula digital magnética que mide el campo magnético terrestre y calcula un ángulo de inicio. Se conecta a un puerto del NXT mediante un cable estándar NXT y utiliza el protocolo de comunicaciones digital I2C. El ángulo de inicio se calcula con una aproximación de 1° y actualiza 100 veces por segundo.

La brújula se encuentra dentro del encapsulado estándar de Mindstorms NXT de lego para que coincida con el de otros elementos Mindstorms.

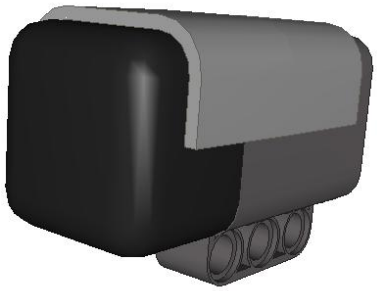


Figura 38. Compás de Hitechnic para LEGO NXT

Acelerómetro.

El sensor de aceleración de tres ejes es capaz de medir la aceleración en los ejes," x", "y", "z". La aceleración es medida en el rango de -2g a 2g con un periodo de 200 veces por segundo.

El sensor de aceleración también puede ser utilizado para medir la inclinación en los tres ejes. Se conecta a un puerto del NXT mediante un cable estándar NXT y utiliza el protocolo de comunicaciones digital I2C. La medida de aceleración para cada eje se actualiza aproximadamente 100 veces por segundo.

El sensor de aceleración se encuentra dentro del encapsulado estándar de Mindstorms NXT de LEGO para que coincida con el de otros elementos Mindstorms.

Los tres ejes de medida están etiquetados "x"," y", "z", como se muestra.

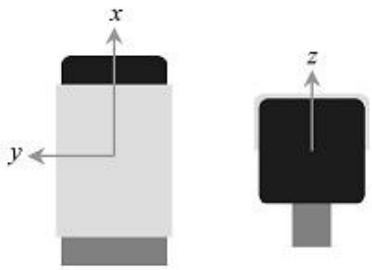


Figura 39. Compás de Hitechnic para LEGO NXT[15]

4.11.2. Motor LEGO NXT.

Este motor es específico para el set NXT (2006). Incluye encoder de rotación, retornando al NXT la posición del eje con una resolución de un grado (1º). Debido conector especial de este motor (no del tipo enchufe de teléfono), un adaptador de cable es requerido para manejar este motor con fuentes regulares de 9 voltios. No se recomienda usar con el RCX, el cual, no envía la corriente alta que este motor puede consumir. La velocidad baja de rotación minimiza la necesidad de engranes externos.

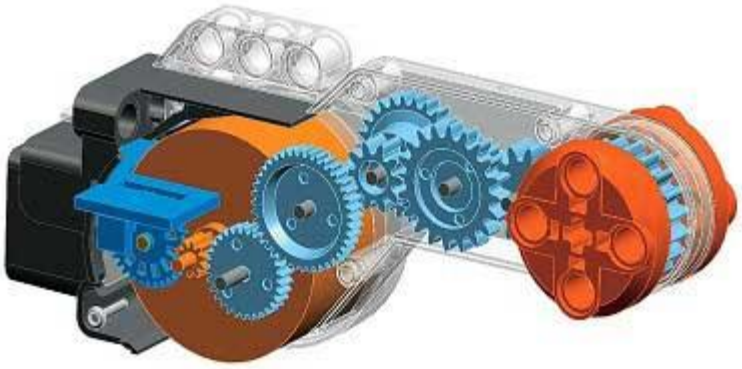


Figura 40. Vista interna del Servo-motor de LEGO NXT[16]

[15] Hitechnic. (2009).

[16] LEGO® MINDSTORMS®, 2009.

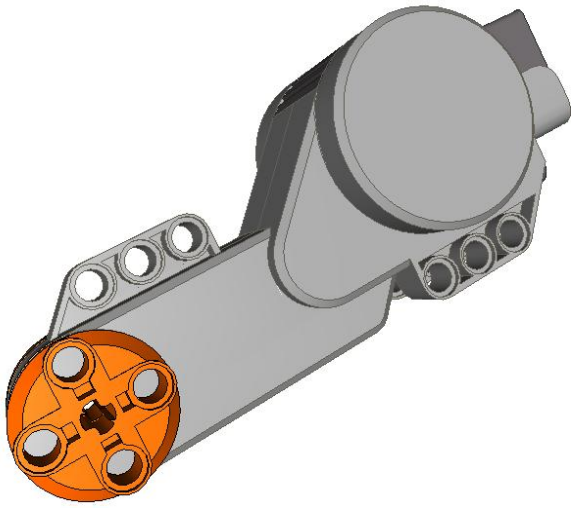


Figura 41. Servo-motor de LEGO NXT

Este motor tiene un peso de 80 gramos.

Características sin carga	
Fuente de potencia	9 voltios
Velocidad de rotación	170 rpm
Corriente sin carga	60 mA

Tabla 8. Características sin carga del servo-motor NXT

Como es usual para los motores DC, la velocidad de rotación es proporcional al voltaje que se les aplica, como se observa en la siguiente grafica. La corriente sin carga depende en poco del voltaje.

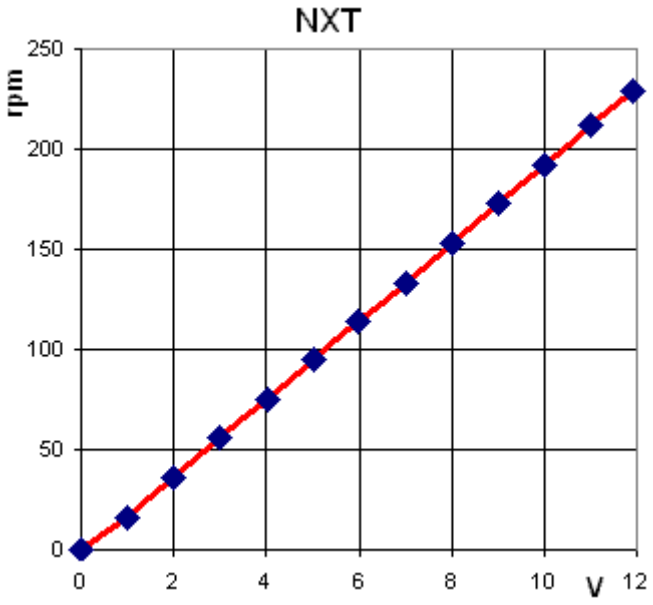


Figura 42. Gráfica velocidad de rotación Vs voltaje aplicado[17]

Características en motor bloqueado.

El consumo de corriente de motor bloqueado es medido simplemente con el eje del motor bloqueado con la mano.

Características motor bloqueado	
Fuente de potencia	9 voltios
Torque de motor bloqueado	50 N*cm
Corriente sin carga	2A

Tabla 9. Características motor bloqueado

El motor NXT está protegido también por un termistor (Raychem RXE065 or Bourns MF-R065). Lo que quiere decir que la corriente alta de 2A (y asociada al torque) puede ser prolongada por sólo pocos segundos.

[17] LEGO® MINDSTORMS®, 2009.

Características con carga.

Voltaje	Torque	Velocidad de Rotación	Corriente	Potencia Mecánica	Potencia Eléctrica	Eficiencia
4.5 V	16.7 N.cm	33 rpm	0.6 A	0.58 W	2.7 W	21.4 %
7 V	16.7 N.cm	82 rpm	0.55 A	1.44 W	3.85 W	37.3 %
9 V	16.7 N.cm	117 rpm	0.55 A	2.03 W	4.95 W	41%

Tabla 10. Características con carga

Velocidad y corriente vs. Torque

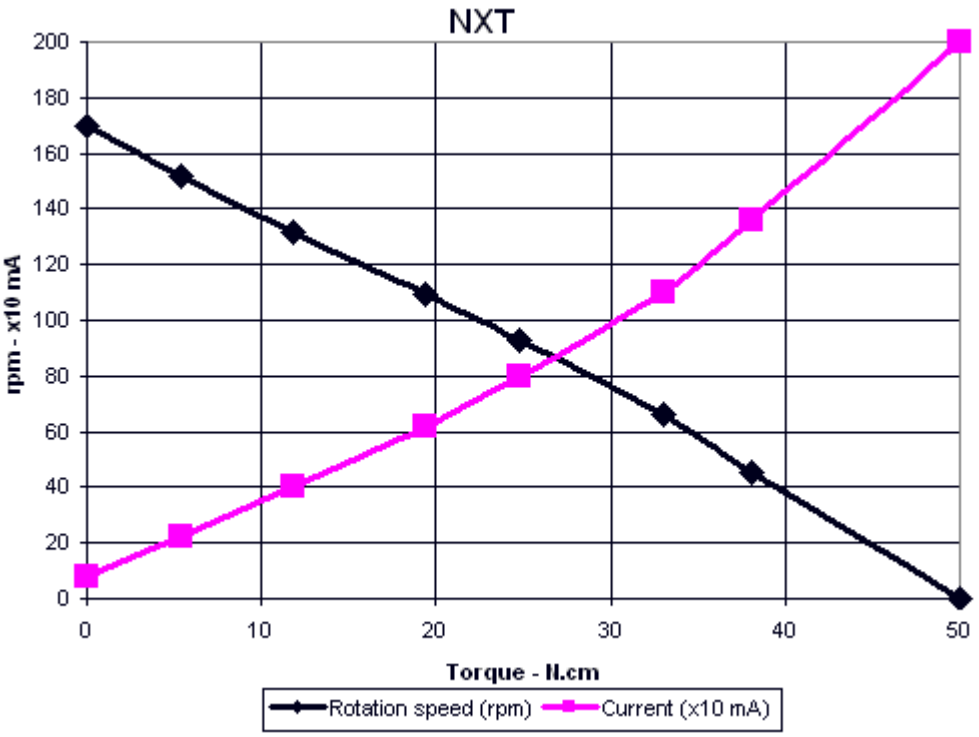


Figura 43. Velocidad y corriente vs. Torque[18]

El motor NXT envía un alto torque gracias a su engrane de reducción de velocidad. Debido a que, también da vueltas lentamente y la eficiencia es reducida en una

[18] LEGO® MINDSTORMS®, 2009.

pequeña cantidad. Este motor puede ser conectado al RCX gracias a la compatibilidad del cable, pero no es recomendado su uso en el RCX porque la alta corriente consume demasiado y es muy alta para la limitación de corriente de éste que es de 500 mA.

Diseño metodológico.

El desarrollo del proyecto de grado se enmarca en dos tipos de investigación: Descriptivo-Aplicada. La parte aplicada se hace necesaria al momento de confrontar los conocimientos teóricos adquiridos de la Robótica y la electrónica con la práctica (Tamayo, 1999).

Se realizó una encuesta para estudiar la problemática del grupo de estudio (estudiantes de octavo, noveno semestre de ingeniería electrónica).

Encuesta realizada:

1- ¿Cuál es tu expectativa para con la materia de robótica y programación?

Marque con una x, respuesta única

A)	Construir un robot		
B)	Por lo menos construir un robot		
C)	Aprender las teorías de la robótica		
D)	Porque necitas los fundamentos para tu desarrollo profesional		
E)	Quieres ver de qué se trata la materia		
F)	Otros	¿Cuál?	

2- ¿Cuánto te sientes preparado para afrontar la materia de Robótica y Programación?

A)	Mucho	
B)	Poco	
C)	Nada	

3- ¿Conoces de algún dispositivo robótico “pertenecientes” a la institución?

A)	Si	¿Cuál?
B)	No	

4- ¿Es la Robótica el área de la electrónica en la que harás énfasis en tu carrera profesional?

A)	Si	
B)	No	

5- ¿Crees que el área de la robótica ofrece oportunidades laborales en la región?

A)	Mucho	
B)	Poco	
C)	Nada	

6- ¿Crees que necesitarás los conocimientos del área de robótica para desempeñarte como profesional?

A)	Mucho	
B)	Poco	

C)	Nada	

7- ¿Cuánto crees que la Institución te motiva a adentrarte en los conocimientos de la robótica?

A)	Mucho	
B)	Poco	
C)	Nada	

8- ¿Conoces de algún desarrollo o aplicación robótica que se haya hecho en la institución?

A)	Si	¿Cuál?
B)	No	

9- ¿Cuánto de los avances de la tecnología robótica has podido percibir ya sea por revistas, TV, internet u otros?

A)	Mucho	
B)	Poco	
C)	Nada	

10- En caso de haber respondido mucho, ¿Cuánto crees que se encuentra la institución actualizada en el campo de la robótica a nivel mundial?

A)	Actualizada	
B)	Relativamente atrasada	
C)	Atrasada	

11- ¿Cuánto conocimiento tienes acerca del área de la Robótica?

A)	Mucho	
B)	Poco	
C)	Nada	

12- ¿Conoces algún lenguaje de programación?, Mencione cuales.

A)	
B)	
C)	
D)	
E)	

13- ¿Sabes que es LEGO?

A)	Si	
B)	No	

14- ¿Sabes que es Lego Mindstorm o Mindstorm NXT?

A)	Si	
B)	No	

15- Enumere una lista de los 5 conocimientos básicos que cree que son necesarios para la robótica:

A)	
B)	

C)	
D)	
E)	

16- ¿Cuánto crees que tienes fundamentados los conocimientos necesarios para realizar una aplicación robótica?

A)	Mucho	
B)	Poco	
C)	Nada	

17- Conoces alguna técnica para el control de Robots. ¿Cuáles?

A)	
B)	
C)	

D)	
E)	

18- Enumere las tres leyes de Asimov:

A)	
B)	
C)	

19- Asocias Robótica con:

A)	Humanoides estilo robocops	
B)	Otros	¿Cuáles?

20- En conclusión , crees que puedas crear una aplicación de robótica, y para ello consideras que :

A)	Solo Y solo si tengo guía total de: Manuales, profesores, ejemplos.
B)	Con alguna ayuda, sea de manuales o de profesores, y las herramientas de hardware y software.
C)	Con ayuda de los manuales me es suficiente y las herramientas me es suficiente...
D)	Sin ninguna ayuda, solo necesito las herramientas (hardware y software)
E)	Sin ninguna ayuda, y puedo construir y diseñar totalmente todo, no necesito ninguna herramienta aparte de mi conocimiento y creatividad.

Con la encuesta se logra ahondarse en el problema del grupo estudiado y conocer el nivel de conocimiento referente al tema del grupo encuestado. También se hace. Tener una parte de metodología de tipo aplicado, debido a que el proyecto no se

queda en la identificación y descripción del problema y sus variables, sino que se ofrece una solución, donde se busca confrontar los conocimientos teóricos adquiridos de la Robótica y la electrónica. Con la práctica, a través de una herramienta pedagógica construida (K.R.O.P.E.L); De acuerdo al libro (Vergel, 1997).

El proyecto parte con la iniciativa de construir un robot móvil desde cero; para esto se realizó una búsqueda en libros de robótica, electrónica, páginas de internet y proyectos de grado ya existentes. Una vez finalizada la búsqueda y analizada la información surgieron dos problemas, uno de tipo pedagógico y el otro de tipo mecánico. El problema mecánico consistía en encontrar un diseño de robot móvil fácil de ensamblar y óptimo para la enseñanza; se observaron diferentes prototipos como el Khepera, Nomad 200, Giraa_01, entre otros, pero estos brindaban una plataforma física muy robusta y poco atractiva para el estudio de la robótica móvil.

Se realizó una nueva búsqueda orientada a la robótica pedagógica, donde se determinó que la herramienta que cumplía con los estándares deseados tanto #a nivel" físico como "a nivel" pedagógico era *Lego Mindstorms NXT,* debido a que el kit ofrece una gran variedad de piezas y diferentes componentes didácticos para realizar un robot educacional completo. También ofrece una amplia versatilidad para realizar diferentes diseños; es decir, el Robot móvil puede ser armado y desarmado para realizar nuevas aplicaciones o tareas, con la ventaja de ofrecer continuamente un diseño robusto y estéticamente atractivo. Para lograr esto se realizó una nueva búsqueda con el objetivo de encontrar una estructura de fácil armado, que permitiera realizar múltiples aplicaciones con pequeñas modificaciones a la estructura base. El proceso llevado a cabo para desarrollar dicho modelo se definió en las siguientes etapas:

Etapa de diseño: Luego de varios intentos fallidos por diseñar la estructura mediante la herramienta virtual MLCad se optó por escoger como guía un prototipo común y más sencillo que viene con el Kit, el cual combinara la facilidad de armado con la flexibilidad que se requería para las distintas aplicaciones.

Existen aplicaciones (comportamientos) en robótica móviles que requieren un grado medio o alto de control, como lo son el control por PID y el control difuso (Fuzzy). El control PID se basa en un algoritmo matemático compuesto de tres parámetros distintos que en conjunto realizan la labor de control. El control difuso se basa en la lógica y en la observación, esto hace que sea un poco mas demorado el proceso de modelado. Para facilitar la parte de diseño del controlador fuzzy, y ahorrar tiempo en implementación, se realizaron previamente simulaciones con la herramienta de diseño de controlador fuzzy del software MATLAB, la cual permitió analizar la respuesta del controlador antes de realizar su codificación, y así notar errores de diseño antes de pasar a la codificación.

Luego de esta etapa se procedió con el desarrollo y construcción de la estructura física.

Etapa mecánica: en esta etapa se tenía el diseño de la estructura (tipo triciclo clásico), el cual, es el más adecuado para desarrollo de las distintas aplicaciones; se procedió al armado de dicho modelo, se realizaron pruebas sobre distintas superficies y se observó que el prototipo presentaba problemas de diseño en la "rueda loca", por lo cual, se optó por modificarla con el fin de lograr una mayor estabilidad en la estructura. Posteriormente se procedió a realizar la etapa de programación (etapa de software).

Etapa de Software: debido a que la mayor parte del trabajo realizado gira en torno a los programas desarrollados para el kit y de la comprensión de los mismos, esta etapa tuvo su inicio desde el momento en que se escogió el tema; consistió en hacer búsquedas exhaustivas de información relacionadas con el entorno y el lenguaje de programación, así como la configuración de cada uno de los software utilizados para el desarrollo de los programas aplicativos.

Definido el lenguaje de programación (Java) y configurado el entorno de programación (eclipse) se procedió al estudio de las clases incluidas en el software de desarrollo (leJOS), con el propósito de facilitar la comprensión de las clases y los métodos de leJOS. Se identificaron las clases y los métodos más importantes, es

decir, los que permitiesen controlar los sensores, accionamientos finales y el ladrillo para luego utilizarlos en el desarrollo de las aplicaciones.

Por medio de pruebas se observó el funcionamiento de los métodos y clases relevantes para el desarrollo de los aplicativos y se documentó de una forma sencilla de entender. Luego de este paso se dio inicio al desarrollo de los programas que permiten la enseñanza de la herramienta pedagógica (Kit Robótico Pedagógico Lego K.R.O.P.E.L)

Etapa pedagógica:

Con los conocimientos obtenidos en la etapa de software se dio comienzo al desarrollo de los programas aplicativos. Se diseñó el esquema de la estructura de un controlador difuso, con el que se busca

- Diseño del primer programa de control difuso para el robot: Para el diseño del primer programa Fuzzy se llevaron a cabo los siguientes pasos.
    - Se estableció la variable a controlar,
    - Se analizó el proceso a controlar.
    - Escogencia de los conjuntos difusos
    - Se estableció la tabla de reglas del controlador
    - Simulación en fuzzylogic toolbox de matlab
    - Codificación del programa en java
    - Pruebas con el controlador fuzzy y el robot

  Aparte de esta estructura llevada a cabo, se realizó una guía donde se ilustra lo necesario para crear un controlador fuzzy, donde se definió la lógica de los conjuntos difusos a estudiar. Se realizó una simulación en el software matlab, con el objetivo de comparar los resultados obtenidos en el ladrillo,

con los obtenidos en el software matlab, esta serie de pasos se siguió durante los otros controladores fuzzy elaborados.

- Diseño de programas experimentales para obtención de valores para lógica difusa: se realizó un programa para tomar datos experimentales del robot y poder construir la base de conocimiento del controlador difuso para seguidor de pared, con el fin de ilustrar los experimentos que se deben de realizar a la hora de estudiar el proceso a modelar.

- Diseño de diferentes ejemplos para cada uno de los laboratorios: Con el fin de hacer más interactivas las guías y reforzar la teoría, se agregaron ejemplos comentados para complementar la enseñanza. En estos ejemplos sencillos se demuestra lo explicado en la guía correspondiente, permitiendo afianzar lo aprendido. Estas guías se dividieron de tal forma que se comprenda de manera general el uso de java en leJOS para programar el NXT.

Registro teórico – práctico: Esta etapa se desarrolló durante todo el proyecto, su objetivo principal fue mantener al día al grupo con los avances desarrollados por cada uno de los integrantes del proyecto. Mediante la generación de documentos se organizó la información contenida en el marco teórico y el aporte más importante de esta etapa fue la creación de cinco documentos que recogen las experiencias con el Robot móvil pedagógico. Estos documentos son guías que permiten al estudiante construir su propio conocimiento Teórico-Práctico de la robótica móvil de manera guiada.

Los cinco documentos mencionados son los siguientes:

- Laboratorio uno: Levantamiento de la plataforma de desarrollo en lejos.

- Laboratorio dos: Manejo de botones y pantalla LCD.

- Laboratorio tres: Uso de sensores y accionamientos finales.

- Laboratorio cuatro: Diseño e implementación de controladores Difuso y PID en robótica móvil.

- Laboratorio cinco: Comportamientos más comunes en robótica aplicaciones y tareas principales.

Se recopiló información y documentación concerniente a las investigaciones en robótica, robótica pedagógica, control, planeación de tareas, sensores, accionamientos finales, y programación, con lo que se creó una base teórica suficiente para cumplir los objetivos del proyecto.

Se realizó un diseño del montaje principal, teniendo en cuenta que tuviera un equilibrio entre sencillez de armado y versatilidad, frente a los diferentes diseños de cada aplicación. Esto se hizo observando los diferentes diseños de lego NXT encontrados en libros y en la web. Luego de haber visto y analizado un número considerable de opciones de armado, se optó por realizar una combinación de las características de cada modelo, de tal forma que permitiera cierto grado de flexibilidad para poder, con solo una base, armar diferentes modelos, realizando la menor cantidad de modificaciones a la base de armado.

Se realizó una guía de instrucciones de instalación y configuración para eclipse con leJOS NXJ, donde se indica paso a paso como descargar, instalar y configurar leJOS, el driver y el entorno de desarrollo eclipse, el cual, es el entorno con el que se programan y descargan las aplicaciones al ladrillo, y leJOS NXJ es el firmware del ladrillo.

Se realizaron diferentes pruebas y experimentos con el Robot (K.R.O.P.E.L) para comprobar cada uno de los métodos y clases de leJOS, con estas pruebas y experimentos se comprendió el uso de dichas clases, sirviendo de base para la documentación de las principales clases y métodos de leJOS. Las cuales se describen de manera sencilla en las diferentes guías de laboratorio. Esta descripción permite comprender y usar los diferentes sensores, motores y el ladrillo de lego NXT.

Se elaboró un controlador PD y un controlador difuso, con el fin de comparar el control convencional (PD), con el control adaptativo (Fuzzy), estos controladores fueron implementados en el robot, el lenguaje usado es java, y ambos programas fueron documentados y explicados., tanto procedimentalmente como cada parte del código; con el fin de mostrar las diferencias en el proceso llevado a cabo en el desarrollo de cada modelo, sea fuzzy o PD, y así poder identificar cual es el modelo más adecuado para cada aplicación. El objetivo del programa elaborado es controlar la distancia del robot hacia un objeto frente a este. El control se hizo por PD, y por control difuso con el fin de comparar, la respuesta, el proceso de diseño, la implementación en código, y de manera general todas las diferencias tanto de diseño como procedimentales de ambos. El esquema de este programa fue utilizado por otras aplicaciones que lo requirieran, como es el seguidor de pared, o el prototipo de la pinza. Y puede ser implementado por cualquier aplicación que requiera un control de distancia.

Se escogieron los principales comportamientos en robótica, para mostrar de manera práctica y general los alcances del kit de lego combinado con la programación en java. Para determinar los comportamientos escogidos se tuvo en cuenta la complejidad de estos, y que se tuvieran las herramientas necesarias (tanto teóricas como físicas) para llevarlos a cabo. Estos comportamientos fueron implementados, documentados y explicados en cada guía correspondiente.

Se realizó un programa para la recolección de datos experimentales, que fueron usados como bases para el diseño del programa de seguidor de pared. Éste programa experimental tiene la flexibilidad de permitir cambiar diferentes parámetros, tales como: distancia, control, sentido de giro; con el fin de, por medio de pruebas, obtener las bases del conocimiento para el controlador difuso. La importancia de este programa es que permite continuar experimentando con el seguidor de pared, permitiendo crear mejoras al programa, para que este sea más eficiente.

Se realizó un robot seguidor de pared, junto con un documento que permite comprender cada una de las fases llevadas a cabo para la construcción e

implementación de un seguidor de pared. Éste se hizo luego de realizar varias pruebas experimentales, diseños físicos, y enfoques en programación, hasta llegar al mejor enfoque encontrado.

Se elaboró una guía de armado de cada diseño, donde se muestra paso a paso como armar cada uno de estos modelos. Esto se realizó usando las herramientas Ldraw y MlCad, para realizar las capturas de pantalla de cada una de las piezas y su armado.

Se realizaron cinco guías de laboratorio divididas en los principales temas necesarios para comenzar en la robótica, y la programación de robot lego NXT en Java. Los temas de estas guías son (ver ANEXO A):

- Laboratorio uno: Levantamiento de la plataforma de desarrollo en lejos.
- Laboratorio dos: Manejo de botones y pantalla LCD.
- Laboratorio tres: Uso de sensores y accionamientos finales.
- Laboratorio cuatro: Diseño e implementación de controladores Difuso y PID en robótica móvil.
- Laboratorio cinco: Comportamientos más comunes en robótica aplicaciones y tareas principales

Estas guías permiten comenzar a realizar proyectos con el kit de robótica lego NXT, permite a los investigadores practicar, leer ejemplos, realizar ejercicios y sirven como bases para futuras investigaciones.

Presupuesto del proyecto.

La inversión del proyecto son todos los rubros de gastos que se efectúan para desarrollar el mismo, la cual, se encuentra en la siguiente tabla.

Concepto	Dólar	Cantidad	Precio Unitario	Valor (Pesos)
Kit de Lego NXT	$279,95	1	$ 699.875	$ 699.875
Sensor de color	$ 54,95	2	$ 137.375	$ 274.750
Sensor Acelerómetro	$ 54,95	1	$ 137.375	$ 137.375
Compass	$ 54,95	1	$ 137.375	$ 137.375
Giroscopio	$ 54,95	1	$ 137.375	$ 137.375
IR Seeker	$ 44,95	2	$ 112.375	$ 224.750
Brick separator	$ 1,69	1	$ 4.225	$ 4.225
Envió EEUU	$ 33,06	1	$ 82.650	$ 82.650
Envió EEUU - Colombia - EEUU	$ 38,00	4	$ 95.000	$ 380.000
Fotocopias		50	$ 70	$ 3.500
Impresión y Encuadernación		3	$ 128.000	$ 384.000
Refrigerios		300	$ 8.000	$ 2.400.000
Transportes		693	$ 1.300	$ 900.900
Horas de trabajo Dir. Del proyecto		260	$ 28.000	$ 7.280.000
Total				**$ 13.046.775**

Tabla 11. Presupuesto del proyecto

Análisis de Resultados, Conclusiones y Trabajo Futuro

5. Análisis de resultados

5.1. Introducción

Los resultados están compuestos por los dos enfoques de la investigación (descriptiva-aplicada). Por una parte se encuentran resultados de experimentos realizados con el robot y los sensores, los cuales fueron llevados a cabo para conocer el funcionamiento de los mismos, y para analizar el comportamiento de las aplicaciones descritas en las guías.

Se realizaron las siguientes pruebas: Prueba sensor de color, prueba sensor de luz, prueba sensor ultrasónico, seguidor de línea, seguidor de luz, control de distancia usando lógica difusa, control de distancia por PID, seguidor de pared usando lógica difusa.

Por otro lado se encuentran los resultados y los análisis de la encuesta realizada a los estudiantes de octavo y noveno semestre (antes de dar la electiva de robótica), para poder conocer la realidad de la problemática.

5.2. Resultados y Análisis de la encuesta:

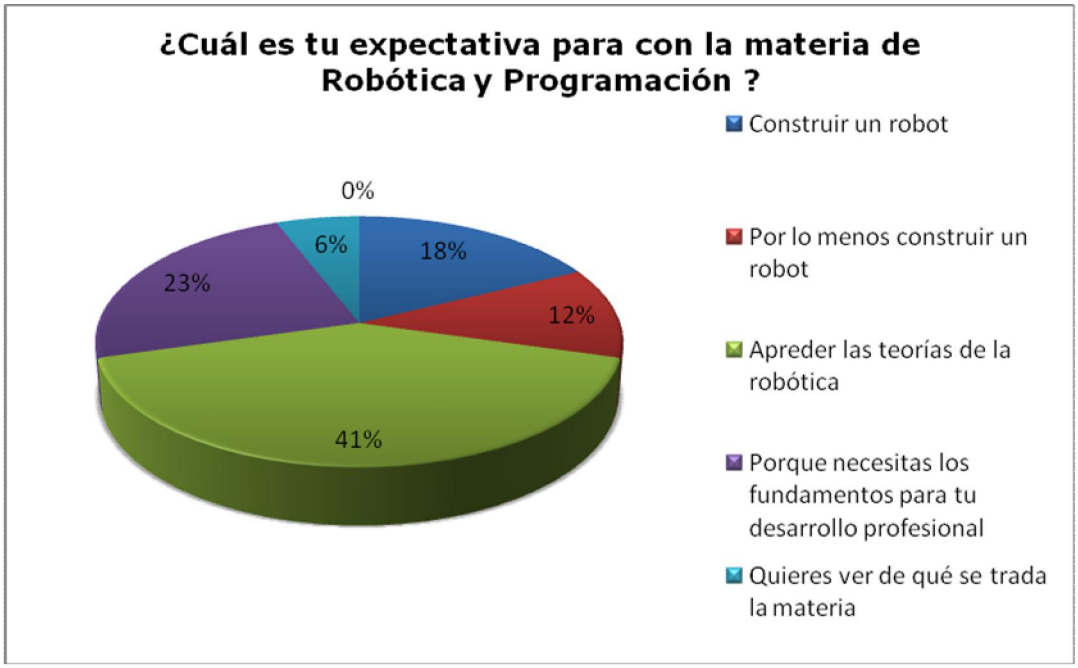
La presente encuesta se hizo con el fin de conocer interés, conocimientos y aspectos más solicitados por los en estudiantes de la CUC en el estudio de la robótica y programación.

Para esto fueron distribuidos 17 formularios, entre los estudiantes de octavo y noveno de ingeniería Electrónica, estos formularios fueron llenados en su totalidad a excepción de algunas preguntas no contestadas.

En las siguientes tablas y gráficos observamos la información recopilada:

1. ¿Cuál es tu expectativa para con la materia de robótica y programación?

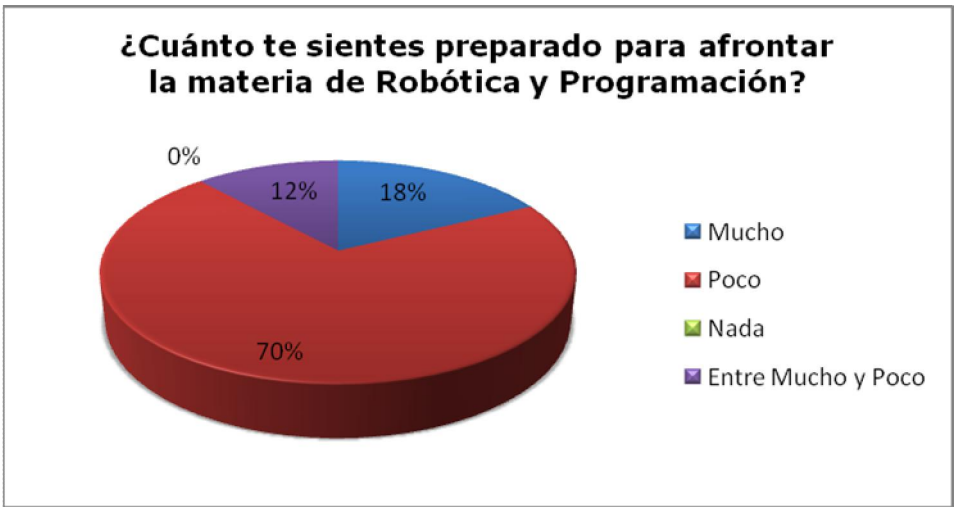
	Cantidad	%
Construir un robot	3	18
Por lo menos construir un robot	2	12
Aprender las teorías de la robótica	7	41
Porque necesitas los fundamentos para tu desarrollo profesional	4	23
Quieres ver de qué se trata la materia	1	6
Otros	0	0
Total	17	100



2. ¿Cuánto te sientes preparado para afrontar la materia de Robótica y Programación?

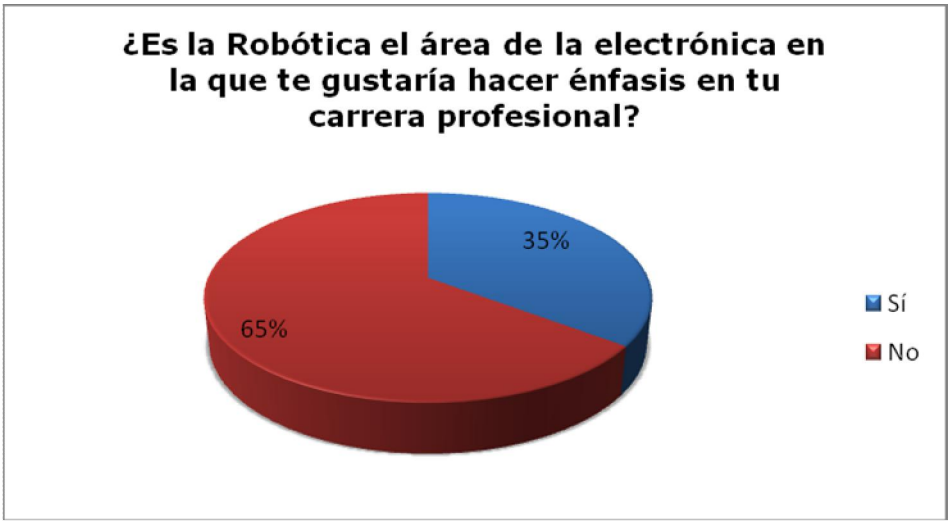
	Cantidad	%
Mucho	3	18
Poco	12	70
Nada	0	0
Entre Mucho y Poco	2	12

Total	17	100



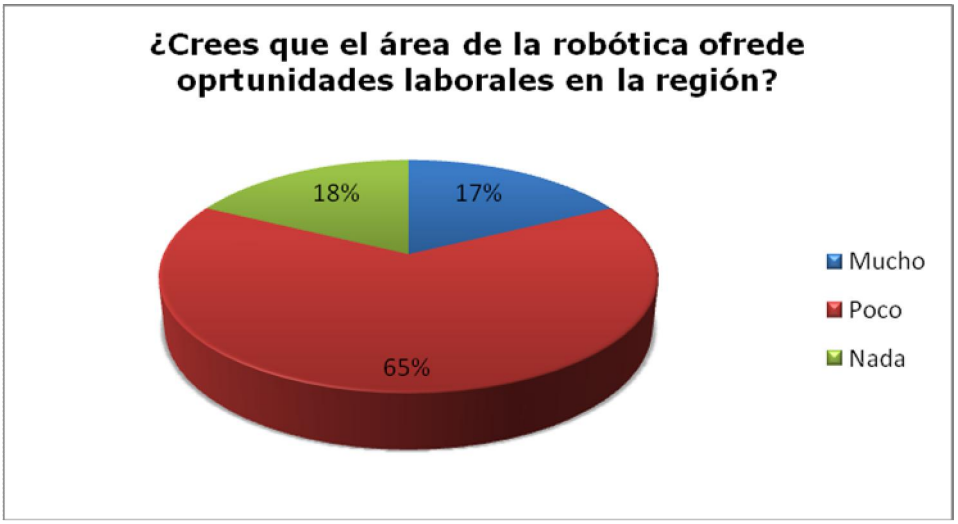
3. ¿Es la Robótica el área de la electrónica en la que te gustaría hacer énfasis en tu carrera profesional?

	Cantidad	%
Sí	6	35
No	11	65
Total	17	100



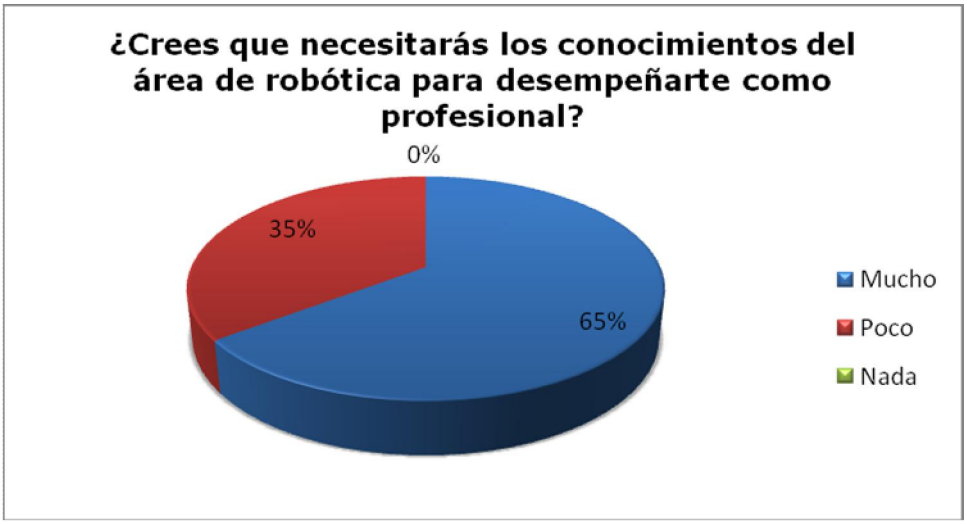
4. ¿Crees que el área de la robótica ofrece oportunidades laborales en la región?

	Cantidad	%
Mucho	3	17
Poco	11	65
Nada	3	18
Total	17	100



5. ¿Crees que necesitarás los conocimientos del área de robótica para desempeñarte como profesional?

	Cantidad	%
Mucho	11	65
Poco	6	35
Nada	0	0
Total	17	100



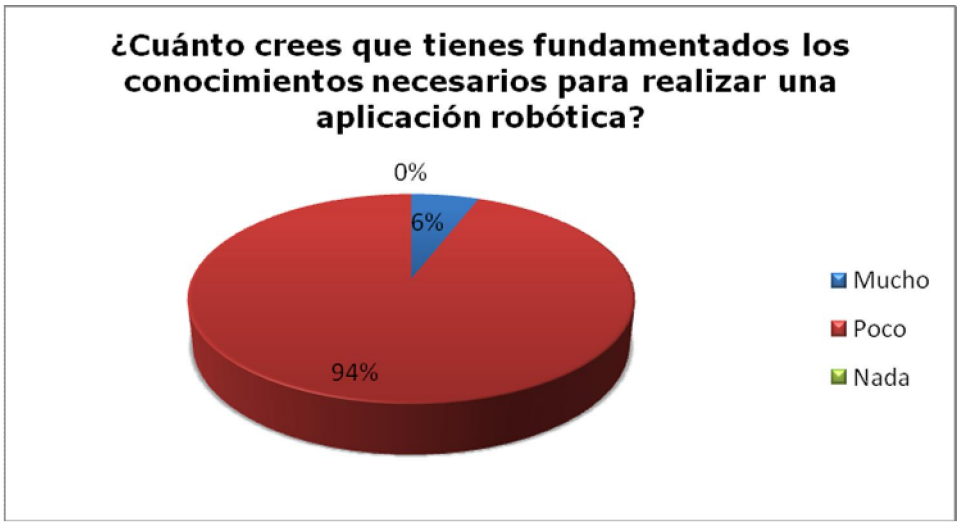
6. ¿Cuánto crees que la Institución te motiva a adentrarte en los conocimientos de la robótica?

	Cantidad	%
Mucho	2	12
Poco	12	70
Nada	3	18

Total	17	100

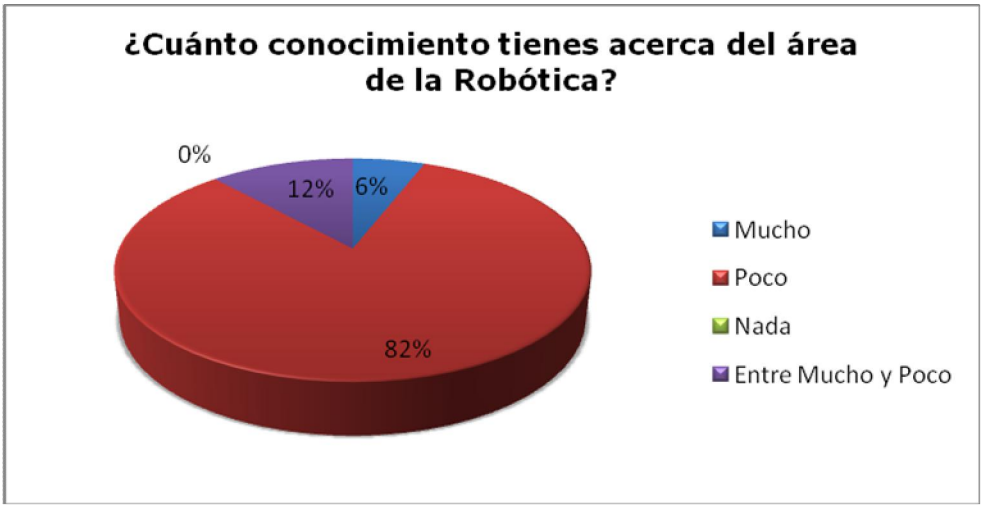
7. ¿Cuánto crees que tienes fundamentados los conocimientos necesarios para realizar una aplicación robótica?

	Cantidad	%
Mucho	1	6
Poco	16	94
Nada	0	0
Total	17	100



8. ¿Cuánto conocimiento tienes acerca del área de la Robótica?

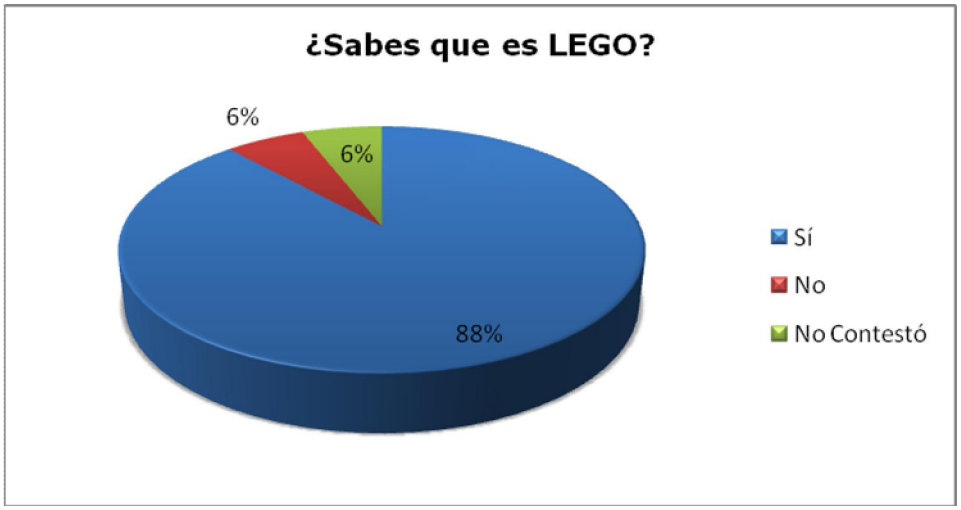
	Cantidad	%
Mucho	1	6
Poco	14	82
Nada	0	0
Entre Mucho y Poco	2	12
Total	17	100



9. ¿Sabes que es LEGO?

	Cantidad	%
Sí	15	88
No	1	6

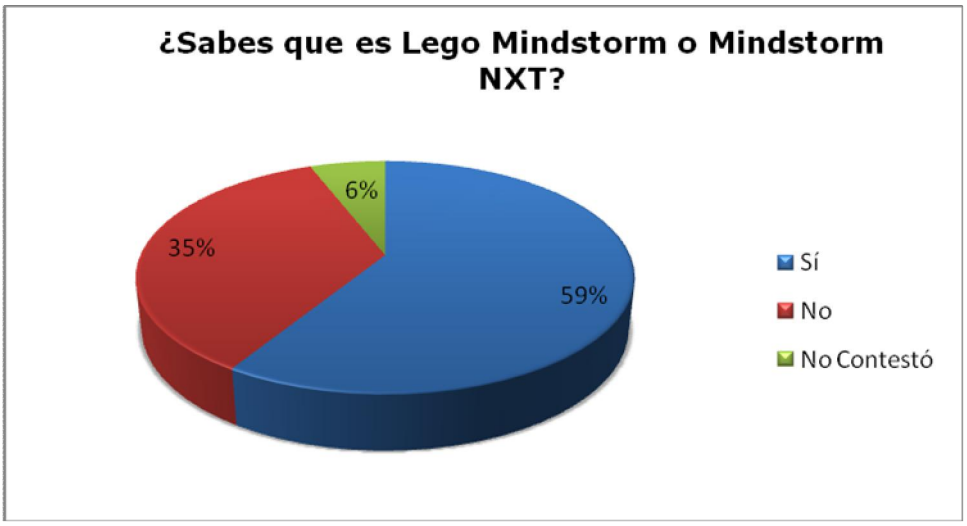
No Contestó	1	6
Total	17	100



10. ¿Sabes que es

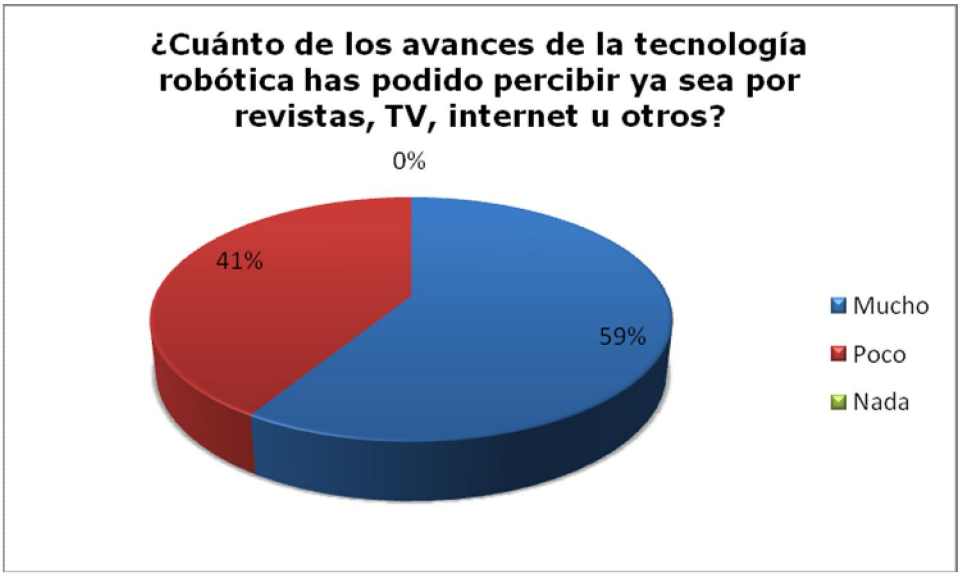
	Cantidad	%
Sí	10	59
No	6	35
No Contestó	1	6

Total	17	100



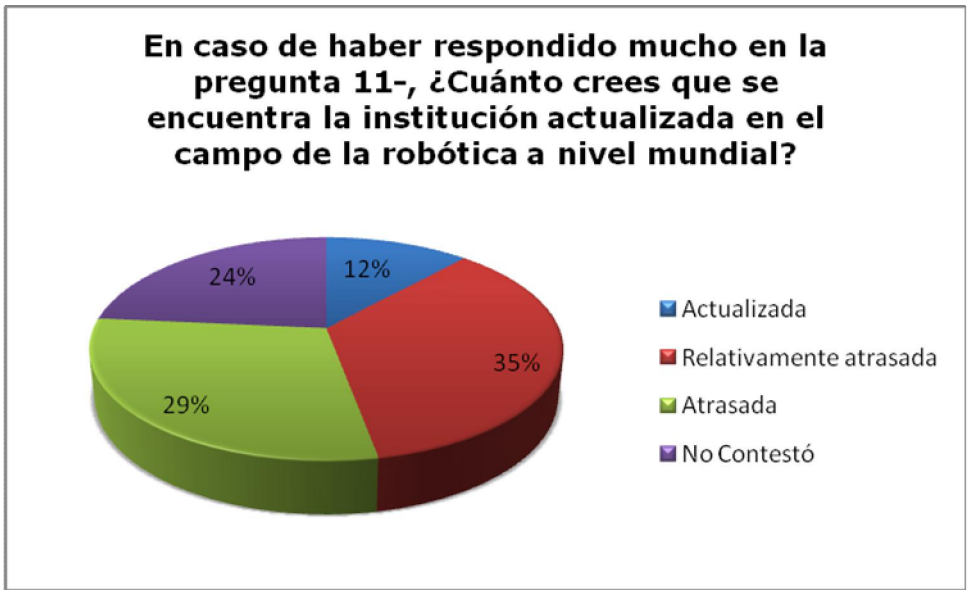
11. ¿Cuánto de los avances de la tecnología robótica has podido percibir ya sea por revistas, TV, internet u otros?

	Cantidad	%
Mucho	10	59
Poco	7	41
Nada	0	0
Total	17	100



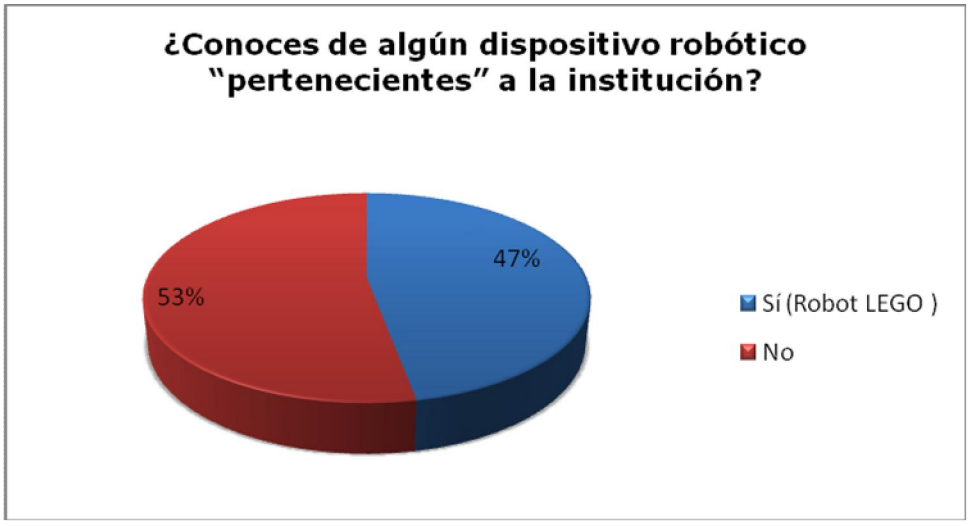
12. En caso de haber respondido mucho en la pregunta 11-, ¿Cuánto crees que se encuentra la institución actualizada en el campo de la robótica a nivel mundial?

	Cantidad	%
Actualizada	2	12
Relativamente atrasada	6	35
Atrasada	5	29
No Contestó	4	24
Total	17	100



13. ¿Conoces de algún dispositivo robótico "pertenecientes" a la institución?

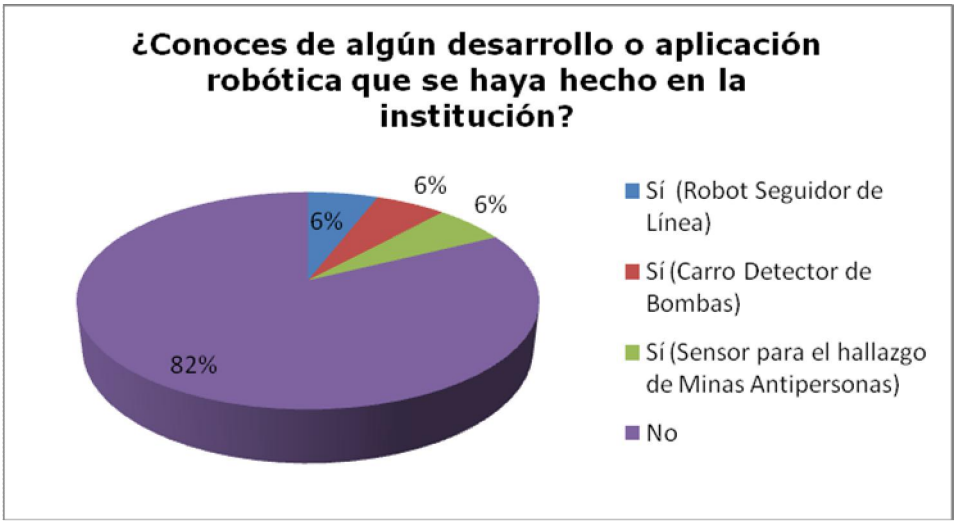
	Cantidad	%
Sí (Robot LEGO)	8	47
No	9	53
Total	17	100



14. ¿Conoces de algún desarrollo o aplicación robótica que se haya hecho en la institución?

	Cantidad	%

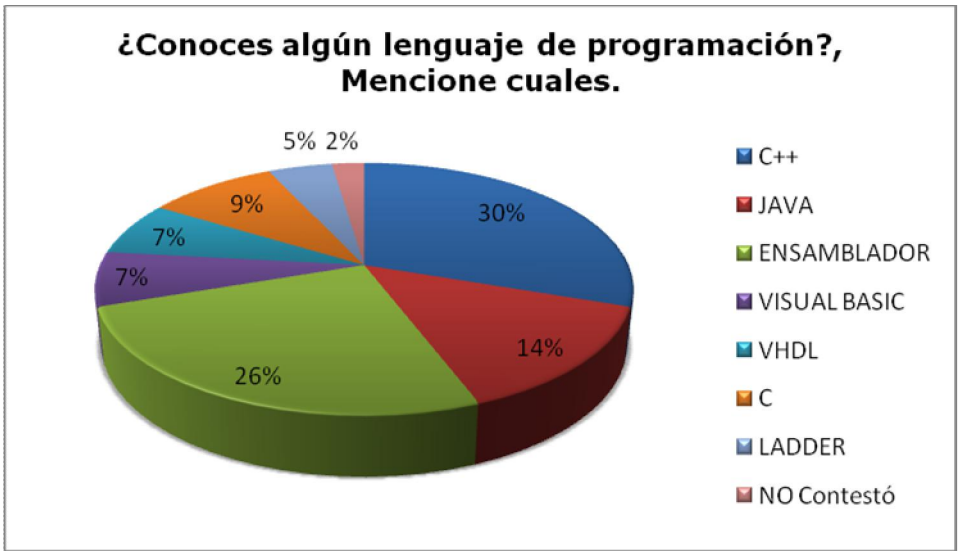
Sí (Robot Seguidor de Línea)	1	6
Sí (Carro Detector de Bombas)	1	6
Sí (Sensor para el hallazgo de Minas Anti personas)	1	6
No	14	82
Total	17	100



15. ¿Conoces algún lenguaje de programación?, Mencione cuales.

	Cantidad	%

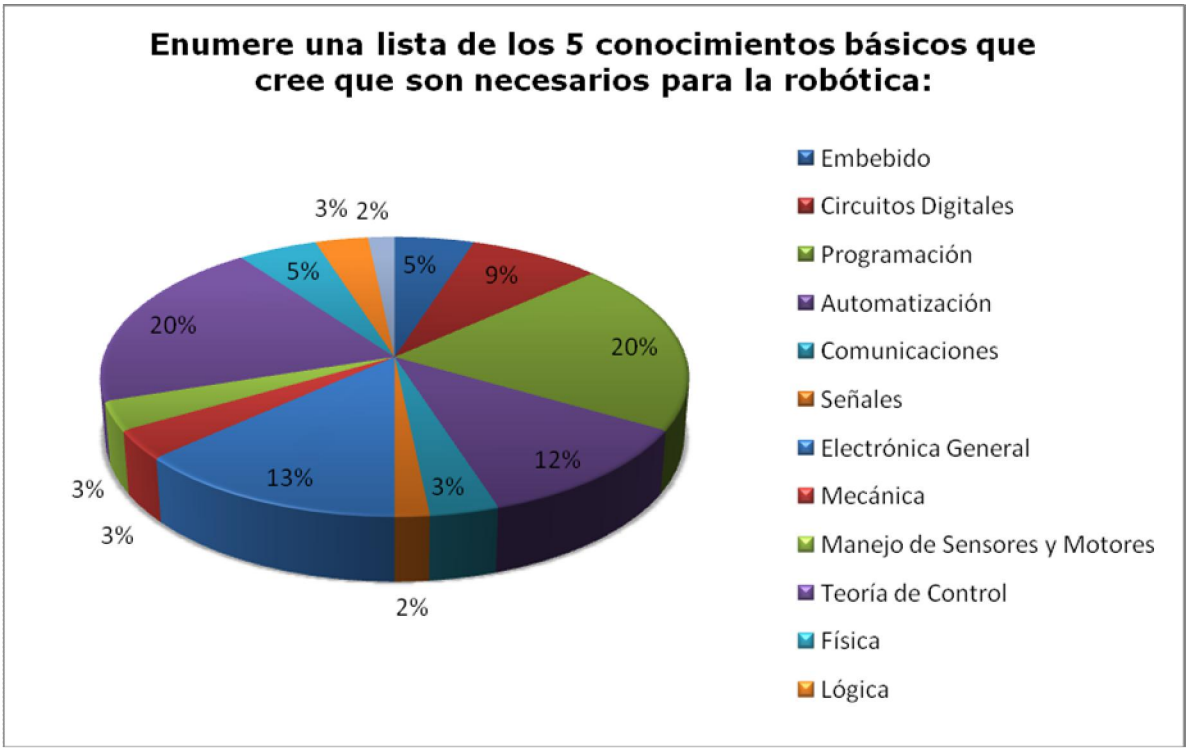
C++	13	30
JAVA	6	14
ENSAMBLADOR	11	26
VISUAL BASIC	3	7
VHDL	3	7
C	4	9
LADDER	2	5
NO Contestó	1	2
Total		100



16. Enumere una lista de los 5 conocimientos básicos que cree que son necesarios para la robótica:

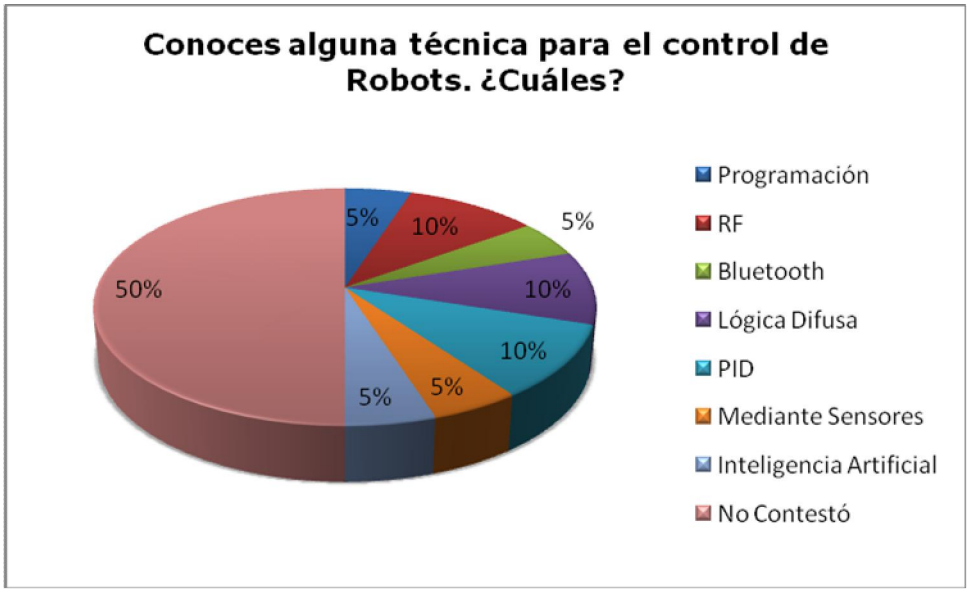
	Cantidad	%
Embebido	3	5
Circuitos Digitales	5	9
Programación	12	20
Automatización	7	12
Comunicaciones	2	3

Señales	1	2
Electrónica General	8	13
Mecánica	2	3
Manejo de Sensores y Motores	2	3
Teoría de Control	12	20
Física	3	5
Lógica	2	3
No Contestó	1	2
Total		100



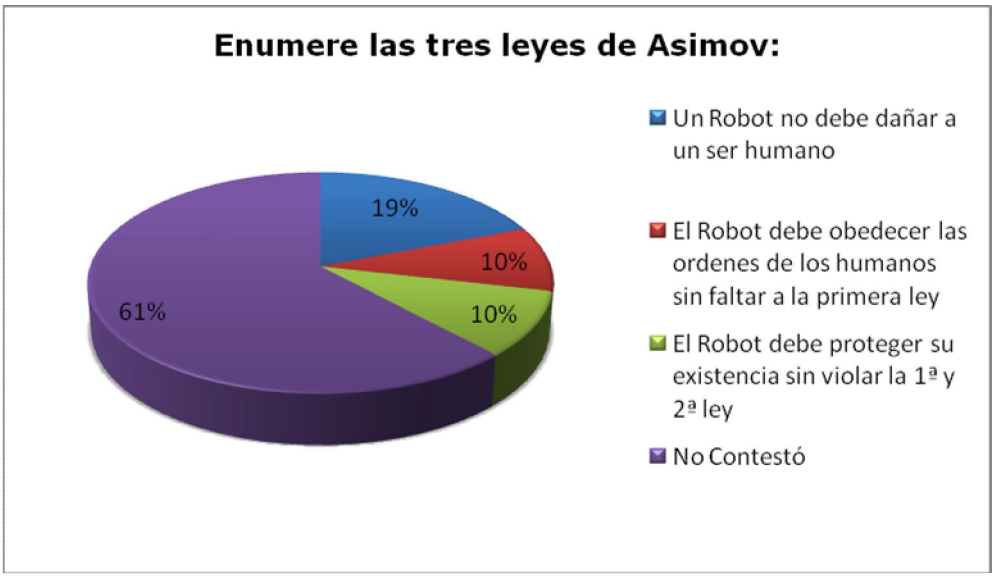
17. Conoces alguna técnica para el control de Robots. ¿Cuáles?

	Cantidad	%
Programación	1	5
RF	2	10
Bluetooth	1	5
Lógica Difusa	2	10
PID	2	10
Mediante Sensores	1	5
Inteligencia Artificial	1	5
No Contestó	10	50
Total		100



18. Enumere las tres leyes de Asimov:

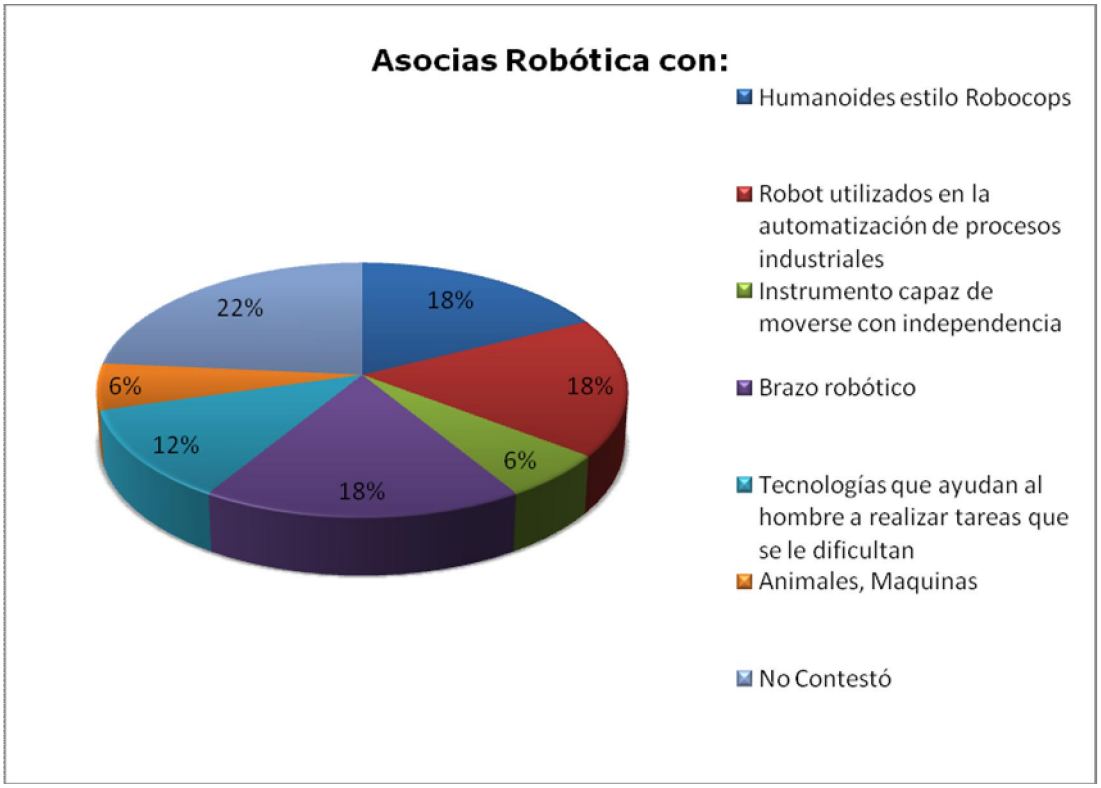
	Cantidad	%
Un Robot no debe dañar a un ser humano.	4	19
El Robot debe obedecer las órdenes de los humanos sin faltar a la primera ley.	2	10
El Robot debe proteger su existencia sin violar la 1ª y 2ª ley.	2	10
No Contestó.	13	61
Total		100



19. Asocias Robótica con:

	Cantidad	%
Humanoides estilo Robocops	3	18
Robot utilizados en la automatización de procesos industriales	3	18
Instrumento capaz de moverse con independencia	1	6
Brazo robótico	3	18
Tecnologías que ayudan al hombre a realizar tareas que se le dificultan	2	12
Animales, Maquinas	1	6

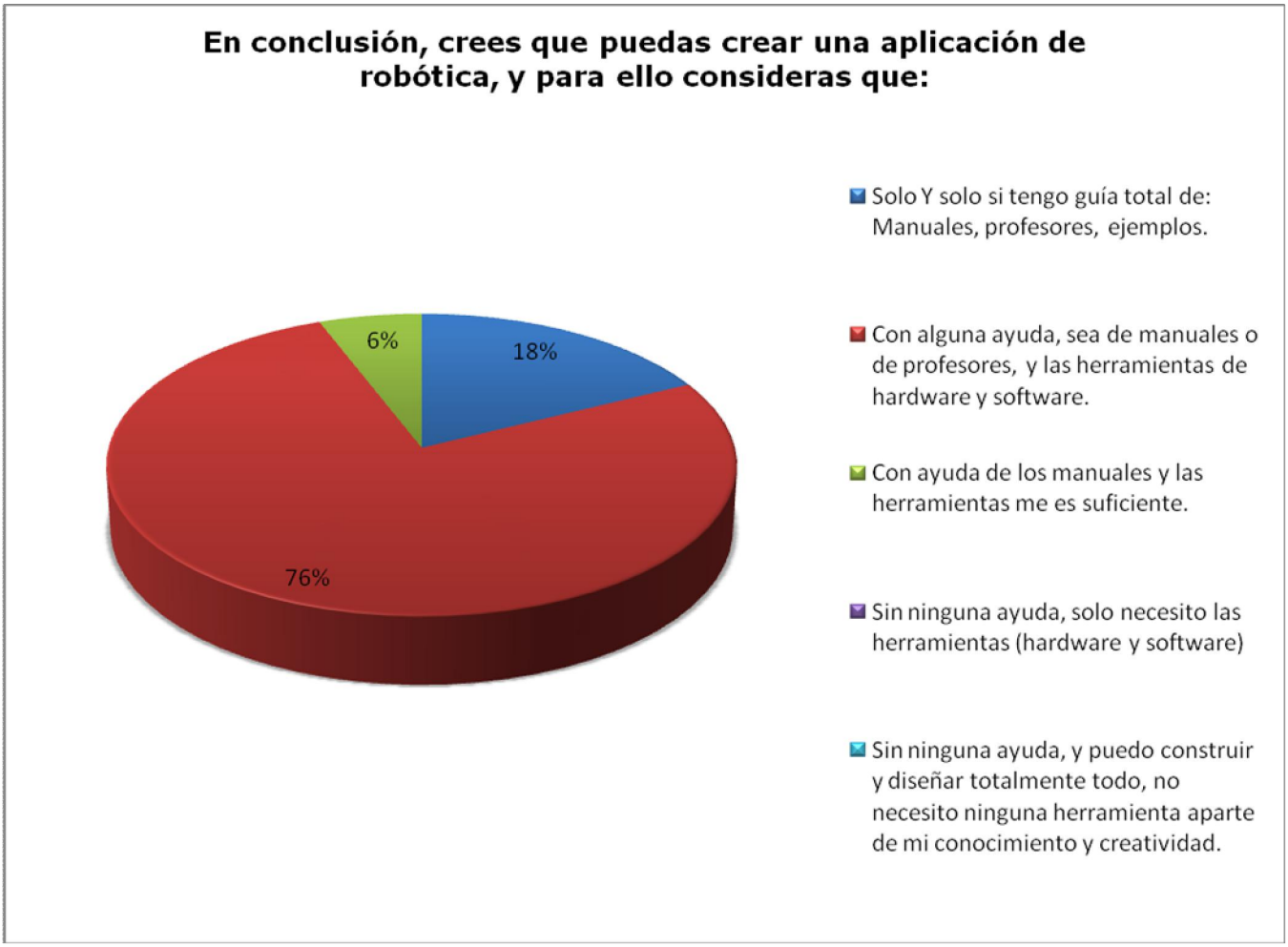
No Contestó	4	22
Total	17	100



20. En conclusión, crees que puedas crear una aplicación de robótica, y para ello consideras que:

	Cantidad	%

Solo Y solo si tengo guía total de: Manuales, profesores, ejemplos.	3	18
Con alguna ayuda, sea de manuales o de profesores, y las herramientas de hardware y software.	13	76
Con ayuda de los manuales y las herramientas me es suficiente.	1	6
Sin ninguna ayuda, solo necesito las herramientas (hardware y software)		0
Sin ninguna ayuda, y puedo construir y diseñar totalmente todo, no necesito ninguna herramienta aparte de mi conocimiento y creatividad.		0
Total	17	100



El análisis de esta encuesta nos arroja las siguientes conclusiones (ver resultados encuesta en “Diseño metodológico”).

- Los estudiantes consideran que necesitan la robótica para ser mejores profesionales. Pero sin embargo no la ven como un área lucrativa o interesante para dedicarse a ella como profesionales.

- No se sienten con las bases suficientes para afrontar la materia. Así como tampoco se consideran capaces de realizar una aplicación con los actuales conocimientos.

- Desconocen la mayoría de las aplicaciones de robótica realizadas en la universidad, y de las herramientas con las que la institución cuenta. Menos

de la mitad conoce que la universidad tiene varios kits de Lego Mindstorm como herramientas para estudiar la robótica, entre otras.

- La mayoría espera que, en la materia de robótica se les enseñe la teoría y a construir un robot.
- De manera general, opinan que la institución no los motiva e incentiva para profundizar en esta área.

Este análisis revela que la mayoría de estudiantes prácticamente llegan “desarmados” y desorientados al momento de iniciar la materia. Por lo que se hace necesario contar con una herramienta didáctica y de un apoyo metodológico fuerte, para aclarar estas dudas, y reforzar los conocimientos en general sobre robótica.

Además los resultados de la encuesta sirven de referencia al momento de determinar la profundidad que tendrían las guías, pues nos revela como se sienten los estudiantes antes de afrontar la materia, y al observar que los estudiantes creen que llegan con pocas o ningunas bases para enfrentar la materia. El objetivo fue encontrar un valor intermedio entre complejidad-orientación; ¿Como es esto? Suministrar las herramientas pedagógicas necesarias para que el estudiante pueda orientarse y comenzar en la robótica. Ofreciendo una guía, mediante, el estudiante es orientado, pero este también tiene que desarrollar y crear parte del conocimiento que le servirá para aprender. Entonces, se pretende “encender” la chispa de la curiosidad, y que sea el estudiante el que con estos medios construya sus propios caminos.

5.3. Pruebas a los sensores

Se sometieron a prueba los sensores que mas serian utilizados en la investigación, tales como sensores de color, luz y ultrasónico; con el fin de entender su funcionamiento antes de comenzar a usarlos en aplicaciones más complejas.

5.3.1. Sensor de color

La prueba del sensor de color se hace con el fin de comprobar su comportamiento a diferentes condiciones de luz y diferentes superficies. Para evaluar el funcionamiento del sensor se utilizó el programa “ColorDetector” hecho por los ingenieros de Hitechnic*(Hitechnic, 2009).

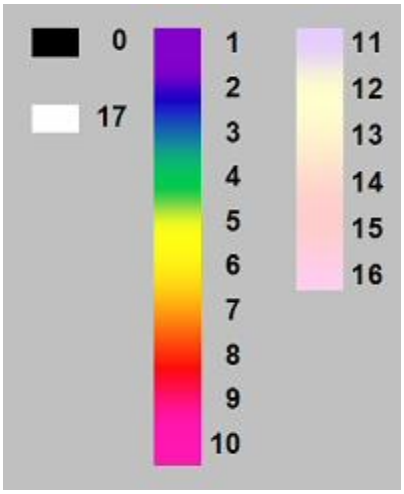


Tabla 12. Colores del sensor.

La tabla 12 muestra los valores que entrega el sensor cuando se utiliza el método “getColorNumber”, este es un número correspondiente a cada color de la tabla.

Para corroborar los valores de la tabla 12 se enumeran las diferentes pruebas realizadas, con diferentes condiciones de luz y tipos de superficies a ser sensadas.

1) Condiciones de luz: Media, bajo sombra, en las horas de la tarde (18 horas), entorno interior.

Objeto: Tabla de colores del libro de armado de lego. Superficie brillante.

Tabla de Resultados:

La tabla 13 muestra 4 columnas de diferentes valores arrojados por el sensor de color:

La columna "# Color" muestra el valor correspondiente arrojado por el sensor luego de usar el método "getColorNumber". Las columnas "R", "G", "B". Corresponden al valor regresado por el sensor de 0 a 255, luego de haber utilizado el método "getRed", "getGreen", "getBlue".

	# Color	R	G	B
Gris claro	4	5	40	0
Gris	4	3	25	0
Sepia	0	0	0	0
Gris oscuro	0	0	0	0
Negro	0	0	0	0
Azul Oscuro	2	0	0	24
Azul	3		24	92
Azul Claro	3	19	40	72
Verde Oscuro	0	0	0	0
Verde	4	0	25	0
Verde Claro	5	35	117	8
Rojo	9	51	0	0
Naranja	9	58	0	0
Amarillo	9	69	162	0

Tabla 13. Resultado de pruebas

Análisis: Al observar el comportamiento del sensor bajo las condiciones arriba mencionadas se concluye que el numero que entrega el sensor correspondiente a un color especifico (de 0 a 17 referente a cada color, ver tabla 12).no es confiable para diferenciar un color de otro, no se observa una clara diferenciación entre los diferentes colores, como se puede ver en la tabla 13 se ve que hay varios colores con el mismo número (ver rojo, naranja, amarillo).

Durante las pruebas se observó que para que el sensor detectara algunos colores (sobre todo los que menos reflejan la luz; tales como negros, grises oscuros, verdes oscuros), físicamente, había que colocarlo a un ángulo especifico, lo que es muy poco práctico para muchas aplicaciones, puesto un pequeño movimiento del sensor interfiere considerablemente en la medición, y se le dificulta diferenciar los

rangos. Si se quiere un buen comportamiento del sensor, hay que hacer un ajuste mecánico específico, para que este quede a una distancia y ángulo adecuado del objeto, para obtener una buena medición.

Se determina, que para diferenciar los colores, es más eficaz utilizar el valor de salida porcentual de RGB. Como se puede observar en la tabla 14, no hay ningún color, con valores de lectura igual.

1) Condiciones de luz: media; única fuente de luz, el sol directo, a las 15:00 horas, entorno en exterior.

La tabla 14 muestra 4 columnas de diferentes valores arrojados por el sensor de color:

La columna "# Color" muestra el valor correspondiente arrojado por el sensor luego de usar el método "getColorNumber". Las columnas "R", "G", "B". Corresponden al valor regresado por el sensor de 0 a 255, luego de haber utilizado el método "getRed", "getGreen", "getBlue".

	# Color	R	G	B
Gris claro	4	57	132	32
Gris	4	31	81	11
Sepia	5	16	42	0
Gris oscuro	0	0	0	0
Negro	0	2	12	0
Azul Oscuro	2	2	0	70
Azul	3	15	38	101
Azul Claro	4	36	90	95
Verde Oscuro	4	3	43	0
Verde	5	13	58	0
Verde Claro	4	45	103	7
Rojo	9	27	0	0
Naranja	8	66	31	0
Amarillo	5	65	171	0

Tabla 14. Resultado de pruebas

Análisis:

Al llevar el sensor a un ambiente de exterior, con luz natural (Sol), se observa que el comportamiento es un poco mejor que en interior y con poca luz. Los resultados dejan ver que en cuanto al Valor (ver tabla 14), se puede ver una diferencia entre la mayoría de los valores de salida correspondientes a los colores, y que este corresponde a la tabla de colores de la información técnica del sensor.

También se nota que en el factor RGB es más notoria la diferencia entre los diferentes sensores.

Se encontró que el problema en cuanto a la posición física del sensor frente al objeto a detectar, persiste (ver sección 5.7.1, donde se habla del problema).

5.3.2. Sensor de luz

La prueba del sensor de color se hace con el fin de comprobar su comportamiento a diferentes condiciones de luz. Para evaluar el comportamiento del sensor se realizó un programa de prueba (“PruebaLuz”), para observar el comportamiento del sensor de luz en dos ambientes diferentes; exterior con luz de ambiente de tarde, e interior de habitación con luz media.

Las pruebas arrojan los siguientes valores:

Tabla de resultados:

	Luz Ambiente	Luz interior
Gris claro	583	580
Gris	554	555
Sepia	518	515
Gris oscuro	487	475
Negro	449	443
Azul Oscuro	519	482
Azul	560	542

Azul Clara	585	581
Verde Oscuro	529	497
Verde	554	542
Verde Claro	616	620
Rojo	640	642
Naranja	627	650
Amarillo	611	623

Tabla 15. Resultado de pruebas

Análisis:

Se observó el comportamiento del sensor en ambos ambientes, y se concluyó que la diferencia de lecturas en luz ambiente y luz exterior no es significativa, a pesar de que en ambos casos difieren un poco los valores de lectura entre los dos entornos (ver tabla), la capacidad del sensor para diferenciar colores por índice de refracción, no se ve afectada por las condiciones de luz del entorno, pero para identificar el color del objeto, hay que previamente tener una lectura de todos los colores, y luego la identificación del color tiene que ser bajo las mismas condiciones de iluminación que se tenía al momento de recolectar los datos de lectura.

También se notó que al separar un poco el sensor del objeto, la lectura cambiaba considerablemente, hay que tener esto en cuenta al momento de la instalación mecánica del sensor, puesto se tienen que tomar los valores de referencia en la misma posición y condiciones de luz en las que se va a realizar la prueba.

Se puede separar colores en cuanto al índice de reflexión, en varios grupos, pero sería complicado establecer un valor de lectura para cada color, pues este varía dependiendo de la cantidad de luz, y de la posición del sensor frente al objeto. Por lo que es recomendable en aplicaciones muy especificas en cuanto a detección de color, utilizar preferiblemente el sensor de color en el modo RGB.

5.3.3. Sensor ultrasónico

Se busca observar el comportamiento del sensor ultrasónico frente a diferentes condiciones de entorno, tamaño y forma de objetos. Para llevar a cabo las pruebas se utilizó el programa “SonicTest” diseñado por los ingenieros de Hitechnic.

El programa es sencillo, lo que hace es mostrar en la pantalla LCD del ladrillo la lectura obtenida por el sensor.

El sensor se colocó frente a objetos planos, circulares, puntiagudos, de diferentes tamaños, y a diferentes distancias.

De estas pruebas se obtuvo que el rango real del sensor es de 4-180 cm (pero es recomendable trabajar con el rango del fabricante hasta 127 cm).

En si el sensor tiene un excelente comportamiento, en cuanto a tiempo respuesta, precisión, y aparte de la temperatura, no es influenciado por el medio. Lo que hace que hace sea muy fiable su medición. Con los únicos objetos que se observó tuvo problemas, fue con objetos muy puntiagudos, así como también lugares como esquinas de paredes o bordes, en los cuales se presentaban medidas erradas.

5.4. Seguidor de líneas:

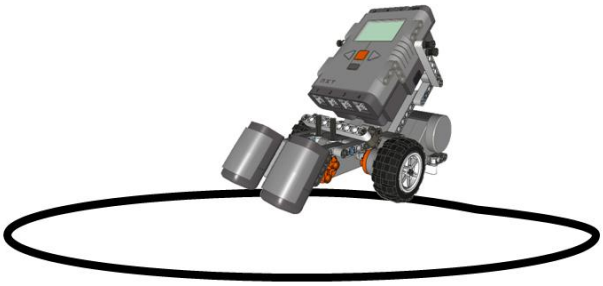


Figura 44. Seguidor de líneas

Una tarea bastante común en la robótica es la del robot seguidor de línea. Se busca que el robot logre seguir una línea negra, independientemente de la forma de ésta (Círculos, curvas, rectas, cuadrados). Se describen todas las pruebas llevadas a cabo para conseguir el objetivo, para lo cual se diseñaron programas y ejecutaron experimentos para determinar la velocidad de giro correcta, y el rango de detección de los colores negro y blanco.

5.4.1. Primera prueba:

Para determinar los rangos de detección se realizó la medición del color negro y del color blanco con un sensor de color HiTechnic; y se le otorgó una velocidad arbitraria a los motores. Luego de esto se procedió a colocar el robot sobre una pista como la de la figura 44. Se observó que el robot no era capaz de seguir la línea; debido a que los rangos de medida de los sensores variaban significativamente con el mínimo cambio en la iluminación (tal como se demostró en las pruebas a los sensores de color, en el ítem 5.3.1). Con respecto a la velocidad de giro también era bastante alta, haciendo que lo giros fueran descontrolados y ocasionando perdida del rumbo.

5.4.2. Segunda prueba:

Se modificó la estructura buscando que la luz no incidiera en la medición de los sensores de color y se disminuyó la velocidad de giro de los servo-motores. De esta prueba se obtuvo que el robot no era capaz de seguir la línea, a pesar de haber cambiado la posición de los sensores, las medidas aun se veían afectadas por el entorno.

5.4.3. Tercera prueba:

Se cambiaron los sensores de color por sensores de luz, se observó que la luz del ambiente afectaba en menor proporción la medida de éstos. Para mejorar el desplazamiento del móvil se le otorgó una velocidad de 500º/segundos para ir hacia adelante y de 100º por segundo para realizar los giros hacia los lados; se determinó el rango de detección del color negro midiendo la línea negra en

diferentes ambientes. Luego se probó el diseño, con las modificaciones realizadas sobre él, y de esta prueba se obtuvo que el robot era capaz de seguir una línea recta sin ningún inconveniente y el rango de medida del color negro era de 33[19].

5.4.4. Cuarta prueba:

Se probó el diseño en las pistas que se observan en la figura siguiente, en una habitación con luz artificial (entorno 1) y en el exterior, con luz natural (entorno 2).

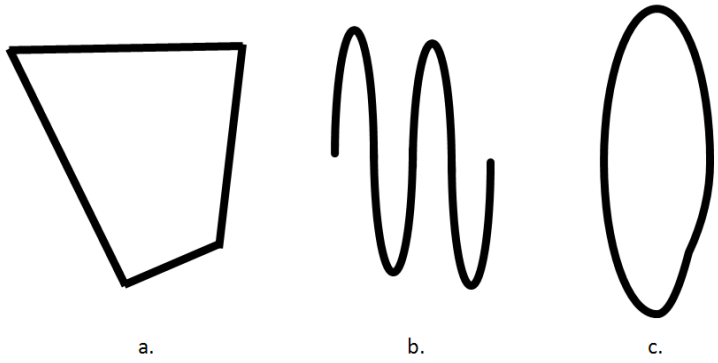


Figura 45. Pistas de prueba

Se obtuvo:

- En la pista “a”. se observó que el robot no fue capaz de realizar el recorrido con los ángulos que ésta presentaba.
- En la pista “b”. se observó que al iniciar el recorrido el robot era capaz de seguirla, pero en las curvas se extraviaba.
- En la pista “c”. se observó que el comportamiento era más regular pero aún se presentaban inconvenientes en las curvas.

[19] 33 es la lectura del sensor para la línea negra, el valor de esta lectura va de cero a 100.

5.4.5. Quinta prueba:

Se optó por reducir la velocidad de los servo-motores para observar el comportamiento del robot, las velocidades fueron asignadas según la siguiente tabla.

Velocidad hacia adelante (º/segundos)	200
Velocidad de giro (º/segundos)	300

Tabla 16. Valores de prueba

Con las velocidades de la tabla, se observó que el robot se desplaza sin ninguna dificultad en pistas sin ángulos rectos ni curvas muy acentuadas, logrando desplazarse en una pista como la ilustrada en la figura 45 c.

5.5. Seguidor de luz:

Otra aplicación interesante y común en la robótica son los robots seguidores de luz, el objetivo fue crear un seguidor de luz utilizando el sensor de luz de lego.

Se diseñaron programas y se realizaron pruebas en diferentes condiciones, para observar el comportamiento y encontrar la más eficiente.

5.5.1. Primera prueba:

Se realizó un controlador sencillo que le “ordenaba” al robot seguir hacia adelante con distintas velocidades, sobre una superficie plana y firme, y se procedió a colocarlo en funcionamiento con el fin de observar el desempeño del móvil en términos de la velocidad, es decir, cuan lento o rápido se movía; de forma cualitativa se obtuvo la siguiente tabla:

Velocidad de giro (º/segundos)	Desempeño
10	Muy lento
50	Muy lento
100	Muy lento
200	Lento
300	Lento
400	Medio
500	Medio
600	Medio
700	Rápido
800	Rápido
900	Rápido
1000	Muy Rápido
1200	Muy Rápido
1300 ó más	Muy Rápido

Tabla 17. Valores de pruebas

5.5.2. Segunda prueba:

Se asignó una velocidad de giro de los servo-motores única de 500º/segundos, siendo esta una velocidad media (según tabal anterior), adecuada para realizar el desplazamiento del móvil; se procedió a probar el robot en una habitación cerrada con luz tenue y una fuente de luz focalizada (lámpara de mano).

Movimiento	Velocidad de giro (º/segundos)
Adelante	500
Giro (derecha/izquierda)	500
Búsqueda	500

Tabla 18. Valores de prueba

De esta prueba se obtuvo que el robot perdía la ubicación de la fuente de luz muy rápidamente y cualquier cambio representativo de luz hacía que se extraviara fácilmente

5.5.3. Tercera prueba:

Con base en las dos primeras pruebas se procedió a cambiar las velocidades de giro, avance y búsqueda en el controlador, experimentando diferentes velocidades así:

Movimiento	Velocidad de giro (º/segundos)				
Adelante	50	100	150	200	250
Giro (derecha/izquierda)	100	200	250	300	400
Búsqueda	20	40	90	130	150

Tabla 19. Valores de pruebas

Después de realizar cada cambio, se procedió a probar el funcionamiento del robot en una habitación oscura y con la misma fuente de luz utilizada en la primera prueba. Se observó que en los primeros 2 ensayos el robot seguía la luz pero sus movimientos eran demasiado lentos; se cambiaron las velocidades una última vez logrando un mejor resultado.

5.5.4. Cuarta prueba:

Basándose en las pruebas anteriores se modificaron las velocidades de búsqueda, giro y avance, así:

Movimiento	Velocidad de giro (º/segundos)
Adelante	300
Giro (derecha/izquierda)	500
Búsqueda	200

Tabla 20. Valores de pruebas

Se observó un comportamiento más eficiente del robot, que logró seguir la luz sin mayores inconvenientes.

5.6. Control de distancia usando control difuso.

Se busca analizar el comportamiento de la aplicación “control de distancia por difuso”, para ello se realizan simulaciones en matlab, se diseña un programa con menor cantidad de conjuntos (fase beta) para observar el comportamiento antes de diseñar el programa completo, finalmente se observa el comportamiento del programa terminado, colocando el robot en diferentes entornos y frente a diferentes tipos de objetos.

Conjuntos de entrada

A continuación, graficas de los conjuntos difusos del programa.

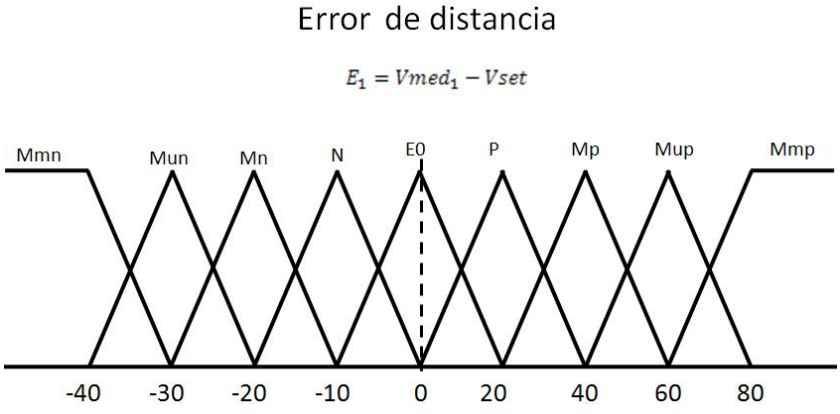


Figura 46. Conjuntos difusos de Error de distancia

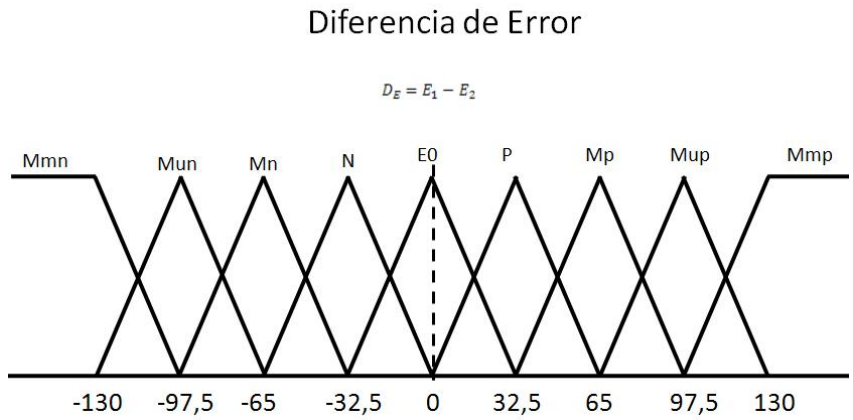


Figura 47. Conjunto difuso de Diferencia de Error

Variable de salida

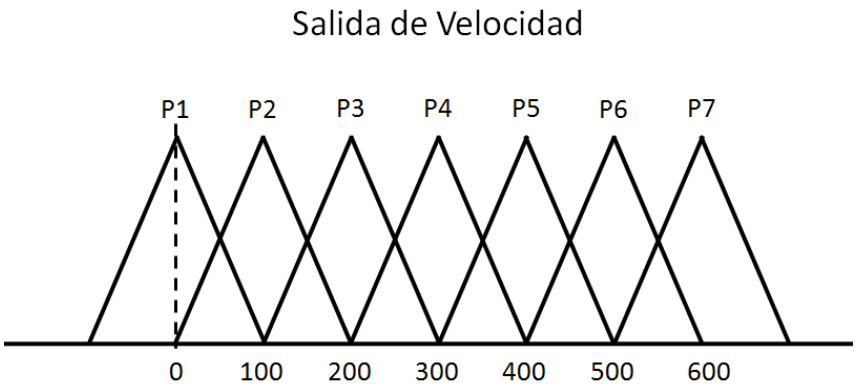


Figura 48. Conjunto difuso de Salida de Velocidad

Simulaciones en matlab

De las simulaciones realizadas en matlab, difuso logic tool box, se pueden observar las siguientes capturas de pantalla.

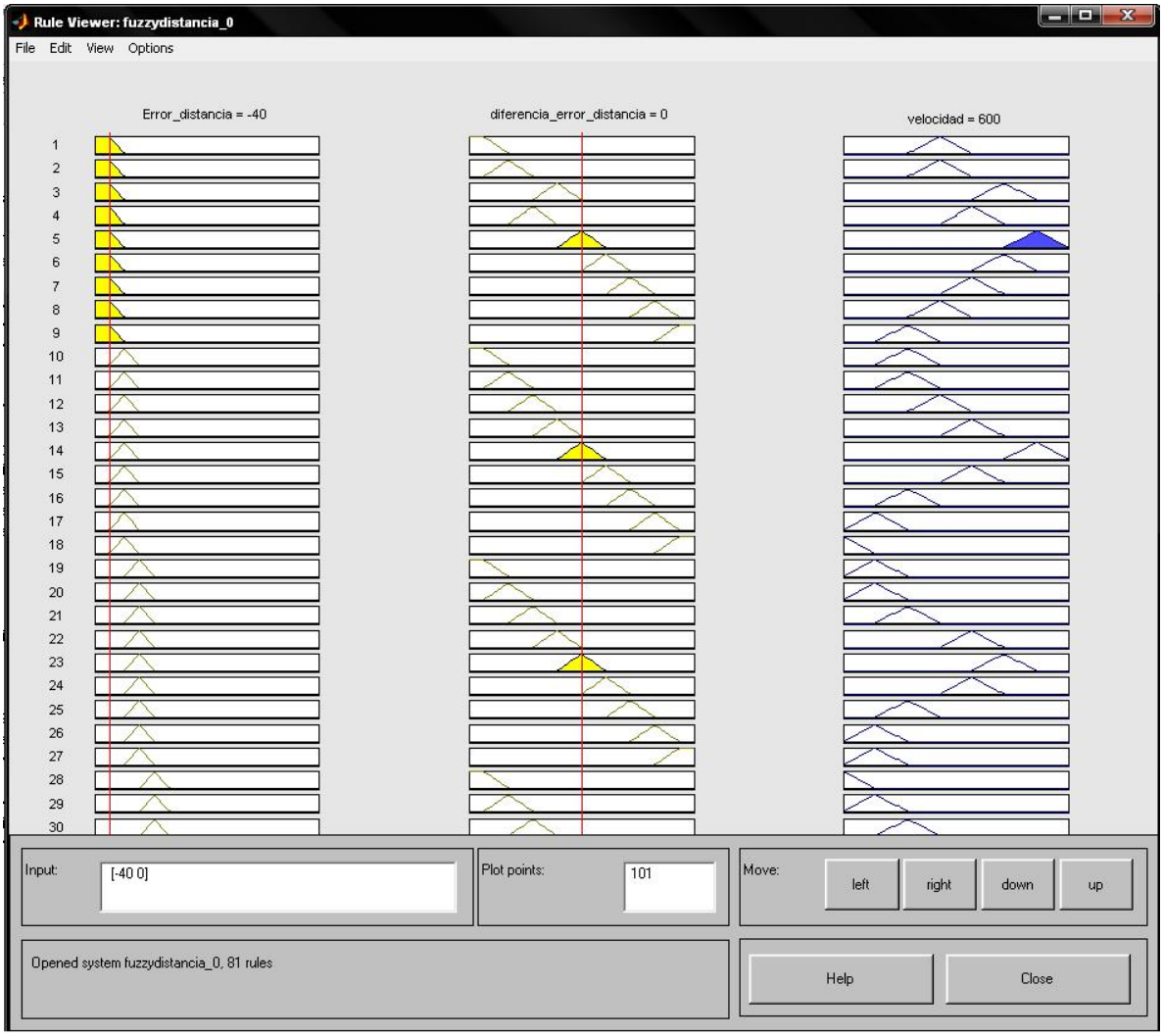


Figura 49. Simulación matlab # 1

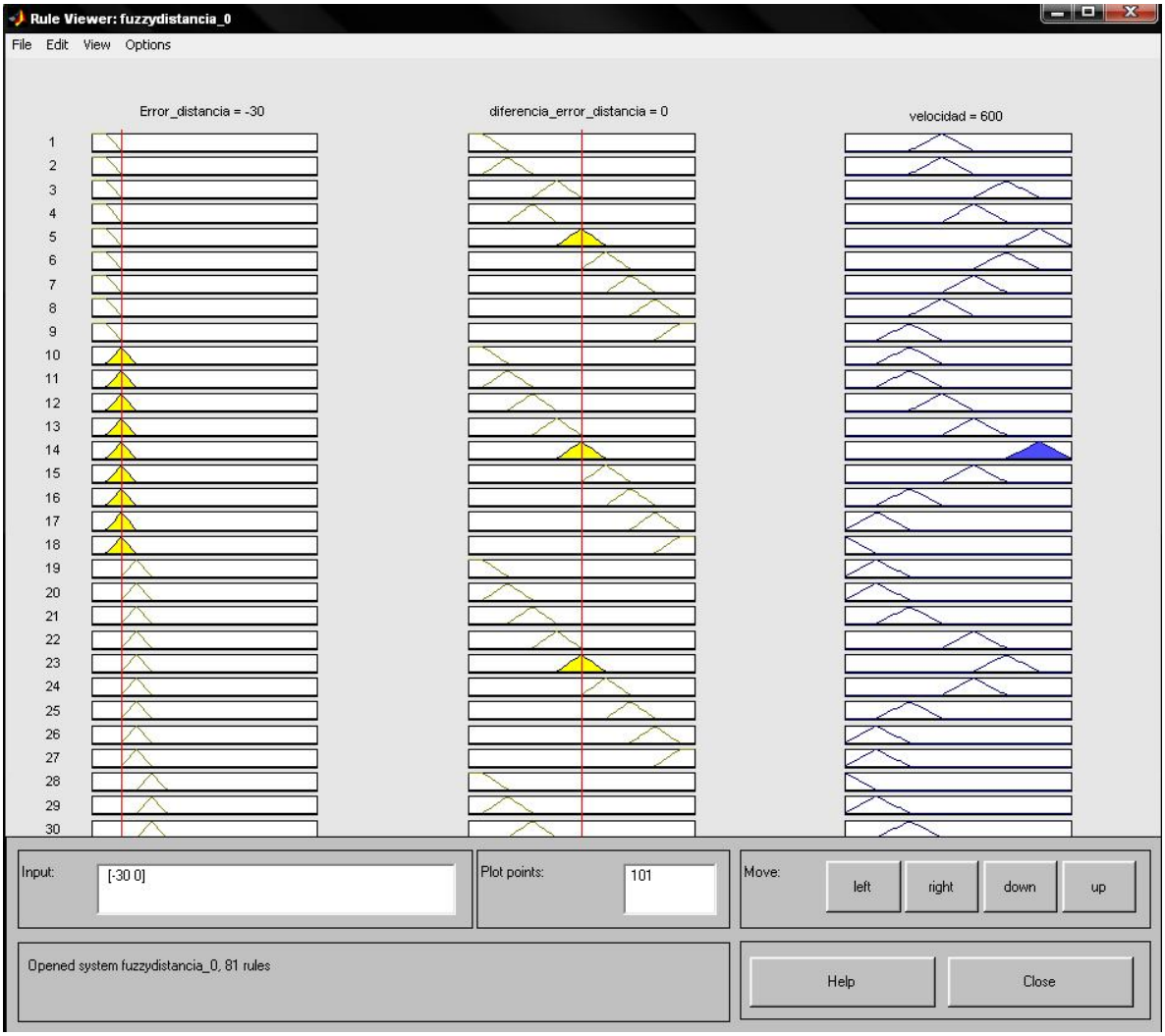


Figura 50. Simulación matlab # 2

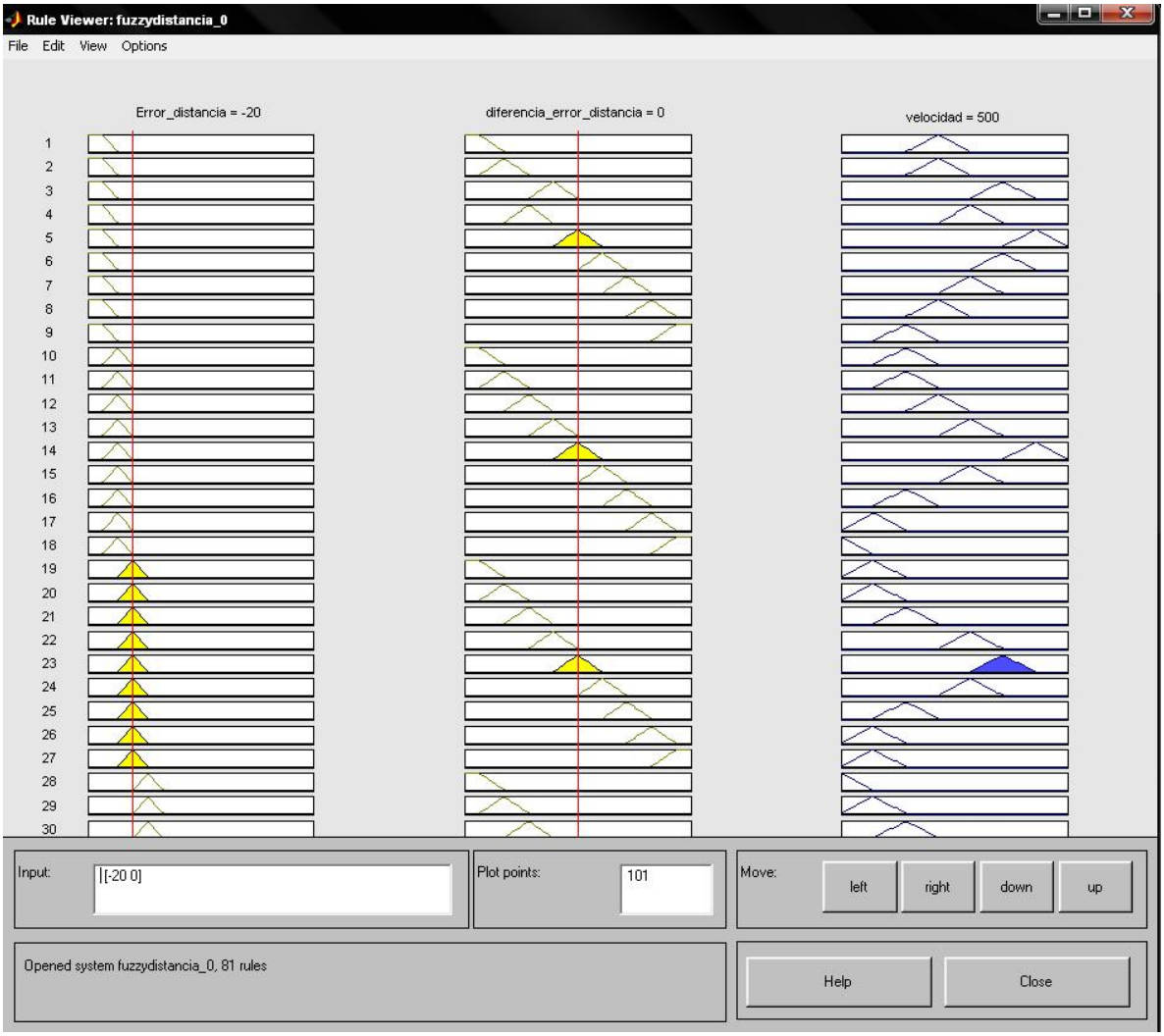


Figura 51. Simulación matlab # 3

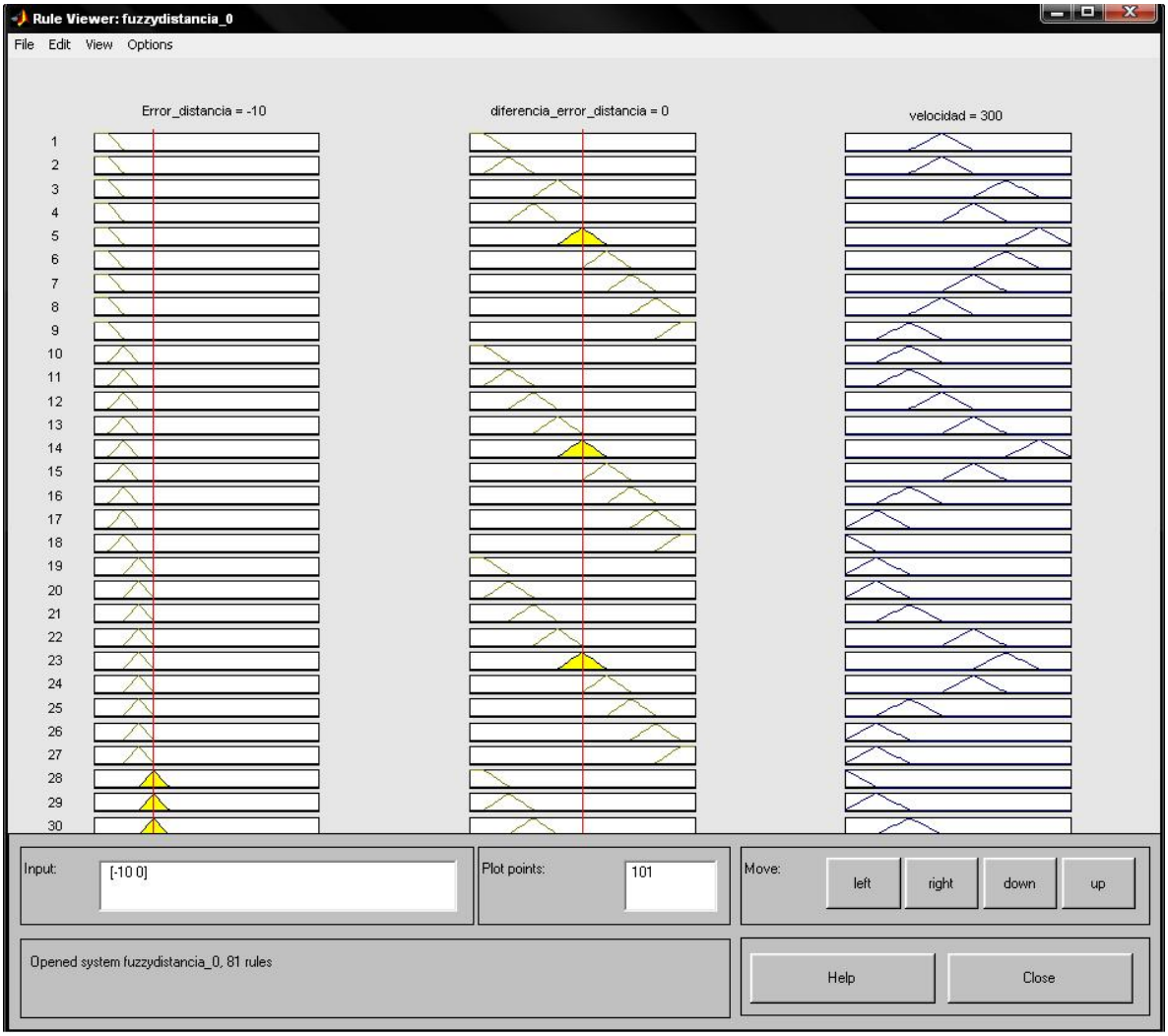


Figura 52. Simulación matlab # 4

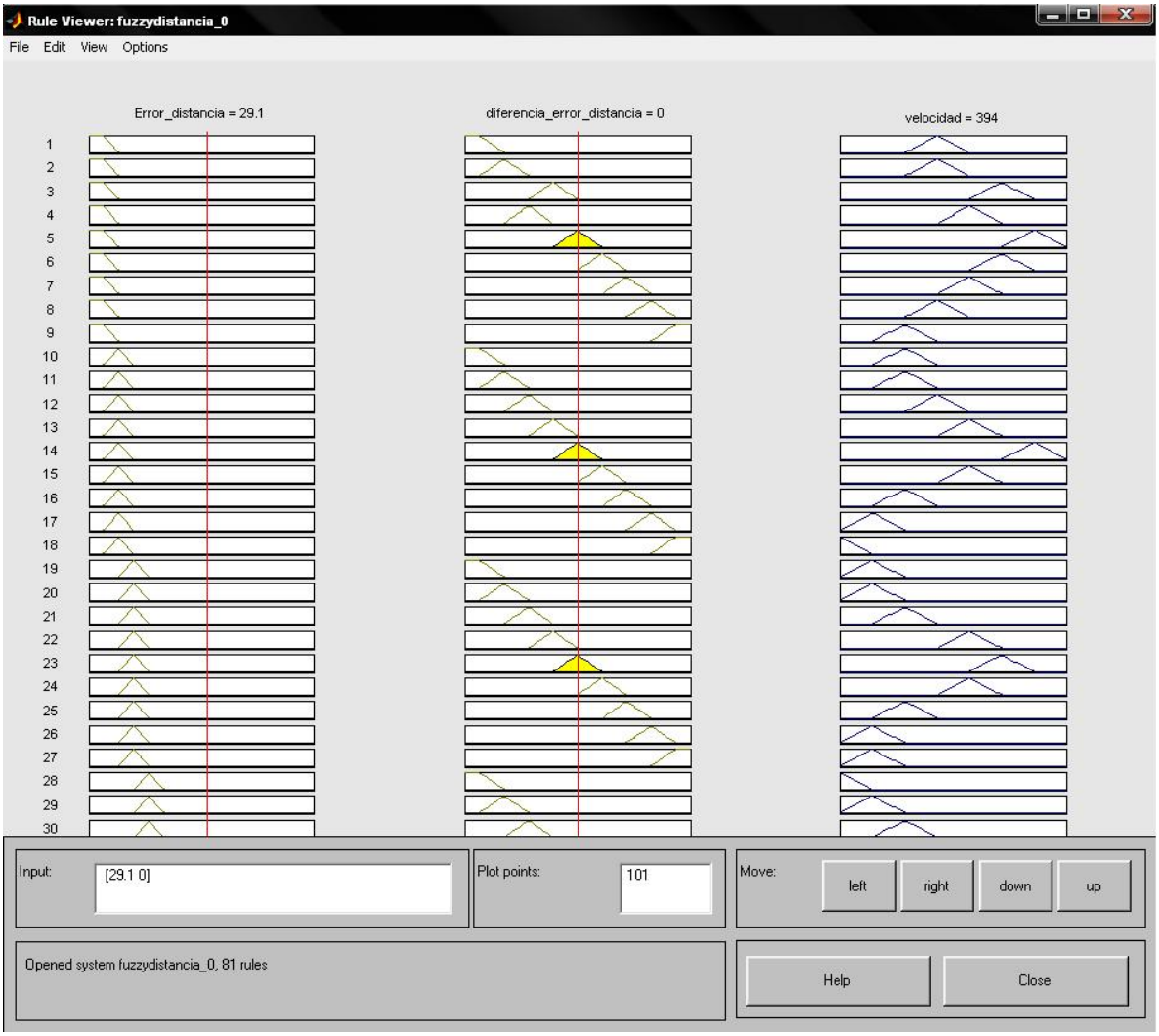


Figura 53. Simulación matlab # 5

5.6.1. Análisis simulación

En las capturas de pantalla se puede observar la variable de salida (velocidad) de acuerdo a diferentes valores de entrada (error de distancia, y diferencia de error de distancia).

Cuando el robot se encuentra a cero centímetros del objeto y la diferencia de error de distancia es cero, el robot debe alejarse del objeto con la mayor velocidad;

según la tabla de reglas (ítem 4.2.2, tabla 27). La máxima velocidad, según se muestra en la grafica de conjuntos de salida es 600º/min.

A medida que esta distancia se va acortando puede verse que la variable velocidad disminuye su valor.

5.6.2. PROGRAMA EJEMPLO O PRUEBA:

Se hizo un experimento previo, con un programa con un número reducido de conjuntos, para realizar pruebas físicas con una aplicación sencilla, y así poder detectar posibles errores, observar las características de la aplicación, y en general el comportamiento del robot con múltiples entornos, y objetos.

La cantidad de conjuntos, los rangos y la tabla de reglas, surge al analizar la aplicación, así como también de estudiar el funcionamiento del sensor ultrasónico. Puesto con esto se obtuvo el rango de trabajo verdadero del sensor ultrasónico (0 - 127 cm) y así se delimitaron los rangos máximos de la grafica de los conjuntos de entrada, por ejemplo los intervalos de cada valor de “error de distancia” los negativos van de 10 en 10, los positivos de 20 en 20, puesto los valores negativos son menos que los positivos, y se quiere una resolución mayor.

Observando el funcionamiento de las combinaciones en el programa de prueba se desarrolló la tabla de reglas de forma empírica. Así como el número de conjuntos se determinó pensando en que hubiera un equilibrio entre números de conjunto y tiempo de ejecución y de respuesta.

5.6.2.1. ANÁLISIS GENERAL

Del análisis en general, se observa un comportamiento bueno con respecto a la distancia a controlar. El robot se detiene a una distancia exacta de 50 cm según la lectura del ultrasónico. Cuando se compara la distancia a la que se detiene el robot frente a un objeto con una medida externa , (regla o metro), esta puede variar un poco con respecto a la distancia set-point, pero esto es por el error en lectura del sensor ultrasónico, y no por error de control del programa.

El robot algunas veces pierde el rumbo a la hora de alejarse de algún objeto muy cercano, puesto arranca con una velocidad muy alta.

En forma general, se observó que los resultados concuerdan con las graficas de la simulación, para corroborar esto. Se uso el LCD del kit, se envían los valores de "velocidad" y "error de distancia".

La velocidad se colocó teniendo en cuenta la información técnica, de tal forma que la máxima salida no fuera superior a un valor medio de velocidad, para evitar posibles errores en la velocidad entre ambos motores.

Se observó además que la variable de entrada "diferencia de error" siempre daba cero, esto quiere decir que el robot realiza todos los cálculos antes de obtener otra lectura en los sensores, por lo que se puede considerar su omisión en futuras versiones del programa, para mejorar eficiencia del código.

5.7. Control de distancia utilizando un controlador PID:

Se realizó un controlador PID que posiciona al robot a una distancia deseada de un obstáculo para regular la velocidad en función de la distancia. Se procedió a realizar las pruebas reales con el robot.

5.7.1. Primera prueba:

Se realizó el algoritmo del controlador proporcional y al ponerlo en marcha se observó:

- El móvil es capaz acelerar, desacelerar y detenerse dependiendo de la distancia.
- Cuando la distancia entre el robot y un obstáculo (Las pruebas fueron realizadas con objetos del tamaño del robot o más grande que este) cambia de manera súbita el móvil acelera y desacelera de forma brusca haciendo que la estructura se estremezca y pierda el rumbo, esto sucede porque el controlador proporcional depende únicamente del error y este error es proporcional a la distancia ($Error = SP - d$).

- El móvil se detenía 4 cm más adelante que el set point programado, esto sucedía debido a que el sensor de ultrasonido no se encontraba en el frente de la estructura sino un poco más atrás. Se modificó la ubicación del sensor, se puso en el frente, logrando mejores resultados.

5.7.2. Segunda prueba:

Se agregó el término integral al controlador y se observó que tanto en un entorno con y sin obstáculos la velocidad se incrementaba demasiado rápido alcanzando el valor máximo de la velocidad de los motores y no lograba retroceder, esto se debe a que el termino integral se incrementaba indefinidamente.

5.7.3. Tercera prueba:

Se optó por quitar el término integral, y se añadió el término derivativo al controlador. Se observo que:

- El robot sigue siendo capaz de detenerse, acelerar y desacelerar como en la primera prueba.
- Cuando la distancia entre robot y obstáculo varía de forma brusca el cambio de velocidad es gradual y no se presenta ningún tipo de inconveniente, esto se logró porque el término derivativo responde al cambio del error de forma instantánea evitando que la magnitud del error se vuelva demasiado grande.

5.7.4. Cuarta prueba:

El algoritmo básico del controlador aprendido es:

$$P_{out} = K_p * e(t) \cong P = K_p * (ErrorN)$$

$$D_{out} = Kd * \frac{de}{dt}(t) \cong D = K_d * (ErrorN - Error1)$$

$$MV(t) = P_{out} + D_{out}$$

Hasta este momento las constantes K_p y Kd no se habían introducido al controlador; se introdujeron y se probaron diferentes valores como se observa en la siguiente tabla, obteniendo los siguientes resultados:

Kp	Kd	Resultado (Velocidad de motores)
0	0	Sin movimiento de los motores – velocidad cero
0,5	0	La velocidad se reduce a un poco más de la mitad de la esperada[20]
0,5	0,5	La velocidad se reduce a un poco menos de la mitad de la esperada
0,9	0	No alcanza la velocidad máxima
0,9	0,9	No alcanza la velocidad máxima
1	0	No alcanza la velocidad esperada
1	1	Alcanza velocidad esperada de forma óptima
2	0	La velocidad alcanza valores de casi el doble de la esperada
2	2	La velocidad alcanza el doble de la esperada
3 ó más	0	La velocidad alcanza valores extremadamente altos
3 ó más	3 ó más	La velocidad alcanza valores extremadamente altos

Tabla 21. Pruebas de velocidad de motores

[20] Velocidad esperada = 500º/segundos.

De estas pruebas se obtuvo de que si a las constante Kp y Kd se les daba valores superiores a uno superan la velocidad esperada provocando en ocasiones que el error se incrementara, si eran menores no se lograba alcanzar la velocidad esperada. También se pudo concluir que la Kp nunca puede ser cero porque la salida del controlador siempre sería cero.

La combinación óptima de las constantes es Kp=1 y Kd=1, ya que se alcanza la velocidad esperada y la respuesta es idónea.

5.8. Seguidor de pared

El objetivo es diseñar un robot que sea capaz de seguir el contorno de una pared, a una distancia establecida previamente, y que también pueda evitar los obstáculos frente a él, para evitar colisionar.

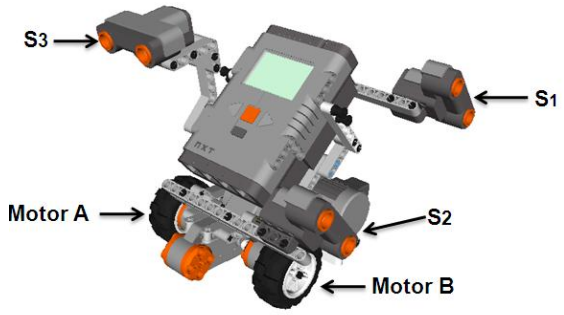


Figura 54. Diseño físico del robot seguidor de pared

5.8.1. Fase diseño:

Para diseñar los conjuntos, combinaciones, tablas, valores de cada conjunto, etc. Se diseñó un programa de prueba, el cual permite que el usuario le seleccione diferentes tipo de “regulación”.

Programa experimental para obtención de valores óptimos de regulación.

El objetivo del programa es poder realizar pruebas con el Robot a diferentes distancias, velocidades y valores de regulación, con el fin de observar como es el comportamiento de este, y así adquirir la base de conocimientos del controlador

difuso. Cabe resaltar que el objetivo con el programa experimental para la obtención de los valores y con el programa del control de distancia, es mejorar y afinar cada vez más el programa de “seguidor de pared” realizando experimentos y pruebas que lleven a realizar.

Para ver más detalladamente el programa (Anexo A guía de laboratorio 5, “seguidor de pared”)

Tabla de datos.

Los siguientes datos fueron obtenidos ejecutando pruebas con el Robot, a diferentes distancias y diferentes valores de regulación.

Distancia	Max	Min	Regulación
60	60	37	10
60	61	42	10
50	50	36	10
50	50	31	10
40	40	29	8
40	40	29	8
30	30	20	5
30	31	23	5
25	25	24	3
25	25	21	4

30	30	19	6
30	29	23	6
60	X	X	15
60	X	X	15

Tabla 22. Datos obtenidos mediante pruebas realizadas al Robot

	muyMuyCerca2	muyCerca2	cerca2	E02	pocoLejos2	lejos2	muyLejos2	muyMuyLejos2
muyMuyCerca,	n7	p3	p4	p9	p9	p9		
muyCerca	n5	n6	p4	p4	p7	p7		
cerca	n14	n11	n3	p6	p8	p9		
E0	n16	n11	n8	c	p8	p10		
pocoLejos	n15	n14	n13	n10	p2	p10	p10	
lejos	n14	n14	n10	n11	n11	p4	p11	
muyLejos	n14	n14	n11	n12	n11	n9	p4	p11
muyMuyLejos	n15	n14	n12	n12	n11	n10	n5	p5

Tabla 23. Tabla de combinaciones

Análisis:

En el valor de distancia 60 a regulación de 15, no tiene datos máximos ni mínimos de distancia, debido a que el comportamiento era muy inestable, el Robot se aproximaba demasiado a la pared, saliéndose del rango de medida del sensor.

Por lo que basándose en los valores obtenidos en el programa experimental se descubrió inicialmente que el rango de valores de regulación es [-10,10].

Del análisis de estos datos se obtienen los valores de la variable regulación, para cada combinación de los conjuntos, y se colocó el robot en ejecución.

Con las primeras pruebas se observa que el robot demora muchísimo en estabilizarse. Por ejemplo: si se encuentra muy inclinado hacia la pared (como en la figura 55, lado izquierdo), demora demasiado en alejarse de ella, y termina colisionando contra esta, antes de cruzar. Al igual si se encuentra alejándose de la pared (como en la grafica 55, lado derecho) demora demasiado en acercarse a la pared, y termina saliéndose de rango

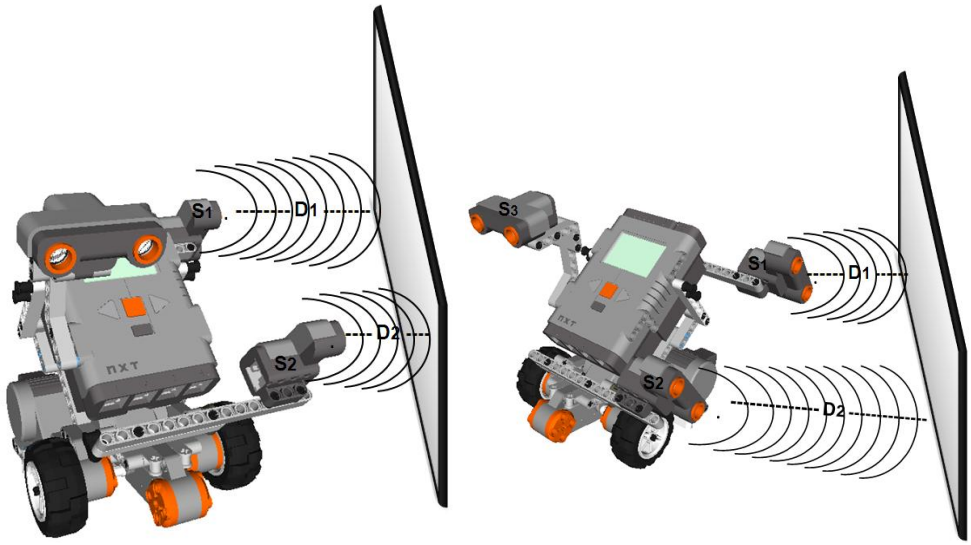
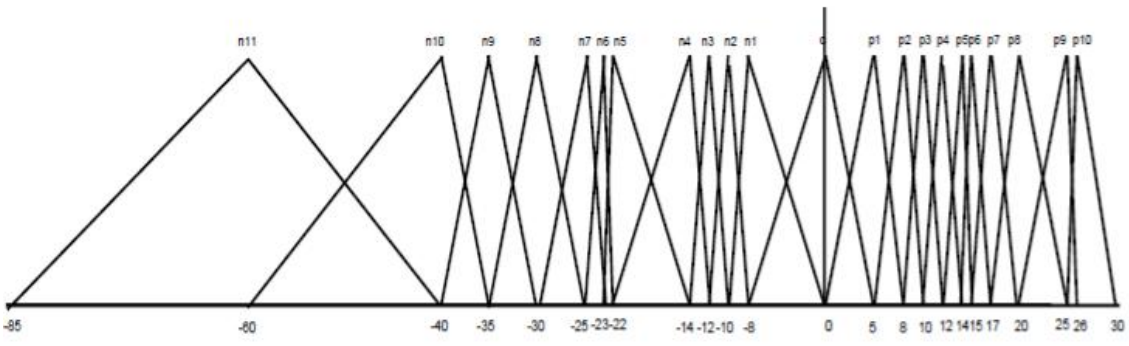


Figura 55. Prueba experimental

Por lo que estos valores de regulación, comienzan a cambiarse, de forma ensayo y error, hasta que se logra establecer la tabla de combinaciones (ver tabla 23), y la grafica de conjunto de salida queda así como se muestra en la figura siguiente (56).



Variable de salida – Regulación

Figura 56. Conjuntos difusos de salida

5.8.2. Simulación en matlab:

Se realizó simulación en matlab para el seguidor de pared. Con este se observa el valor de la variable de salida “regulación” que se les entregaría a los sensores. Cabe resultar que hasta el momento se encontró un error en la simulación. Cuando el robot se encuentra en el set-point el controlador arroja una salida, (aunque es muy baja), pero de todas formas sería bueno tenerlo en cuenta.

A continuación graficas de la simulación realizada:

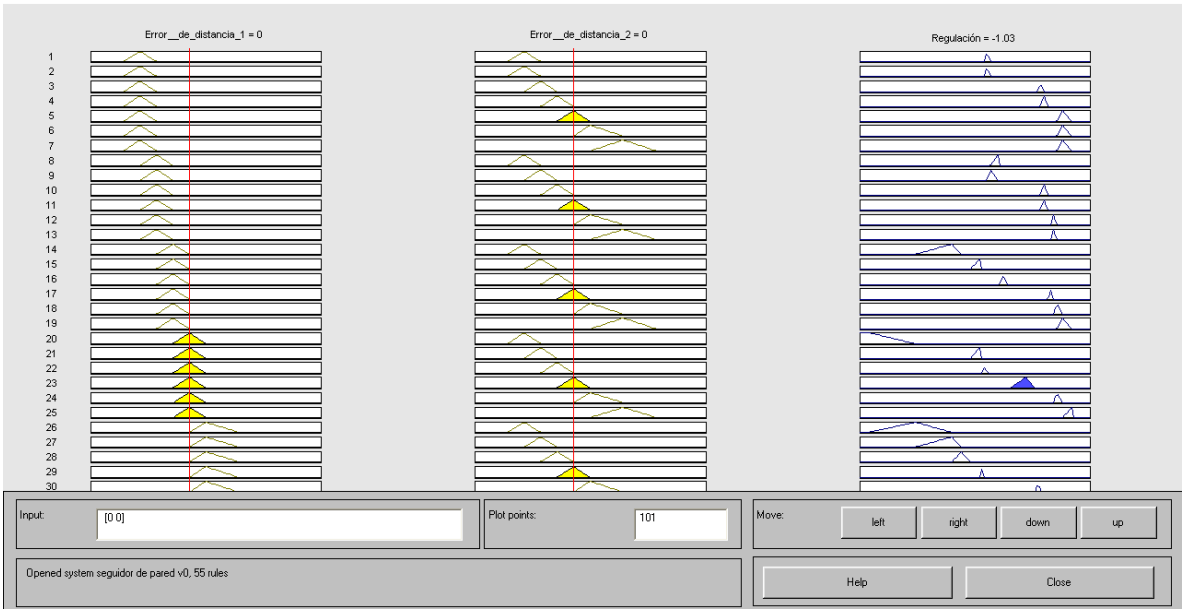


Figura 57. Gráfica de la simulación # 1

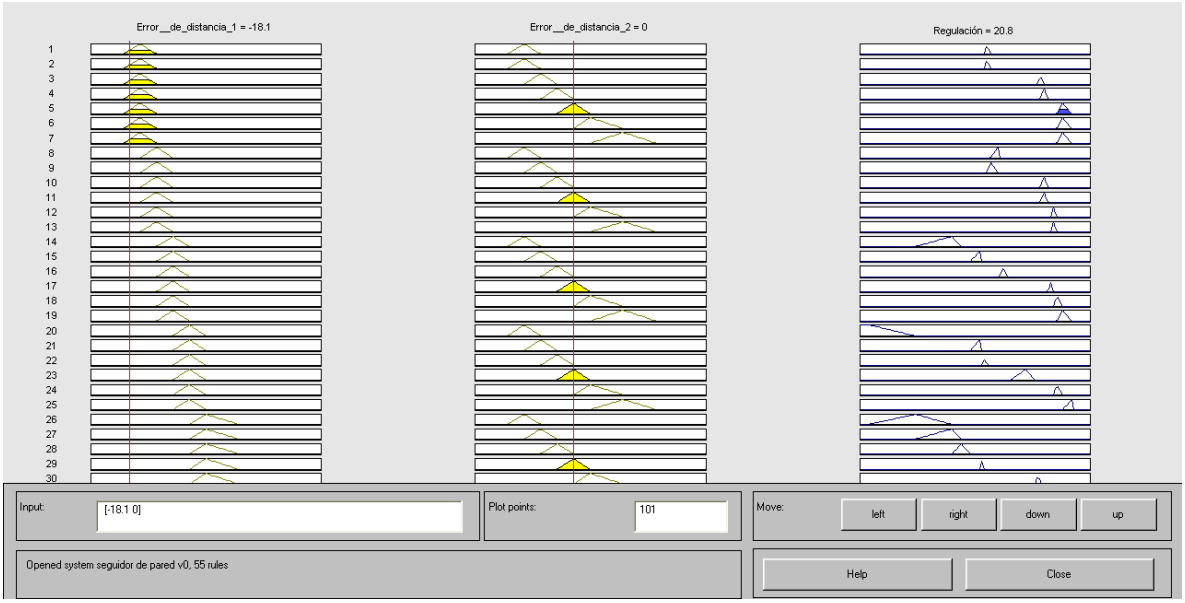


Figura 58. Gráfica de la simulación # 2

- En este caso el robot se encuentra así: el sensor 1 a 2 centímetros de la pared, y el sensor 2 a 20 cm, el programa arroja una salida de "regulación" de 20.8, así el robot avanzará un poco hasta enderezarse.

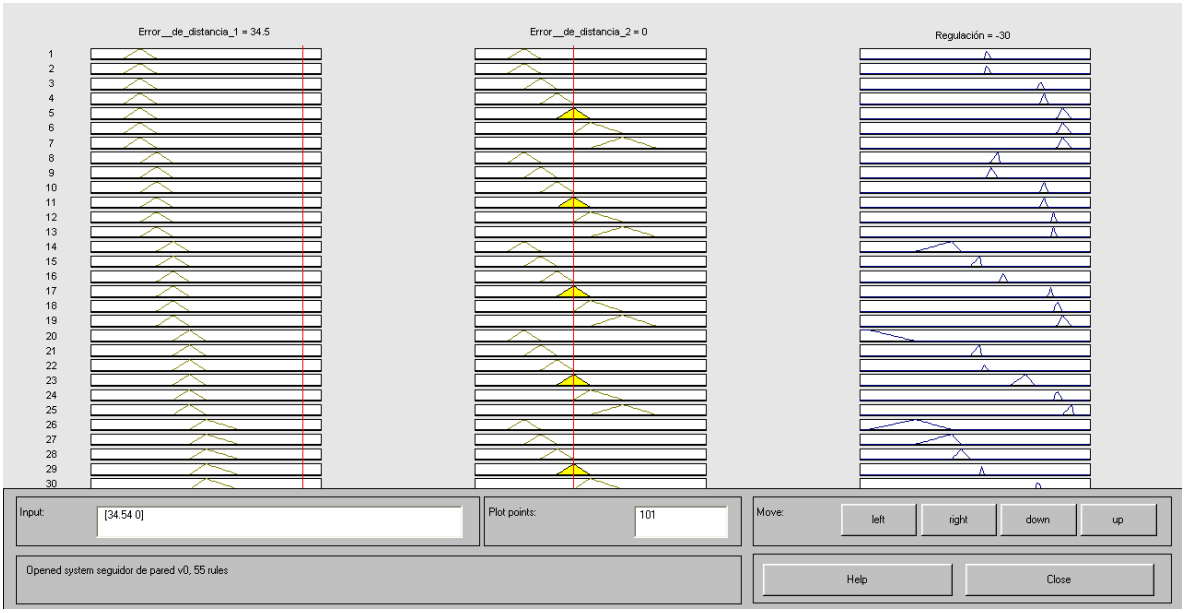


Figura 59. Gráfica de simulación #3

- El robot se encuentra: sensor 1 a 50 cm de la pared, sensor 2 a 20 cm. El controlador entrega una salida de “regulación” de – 30, y así este se irá acercando paulatinamente a la pared.

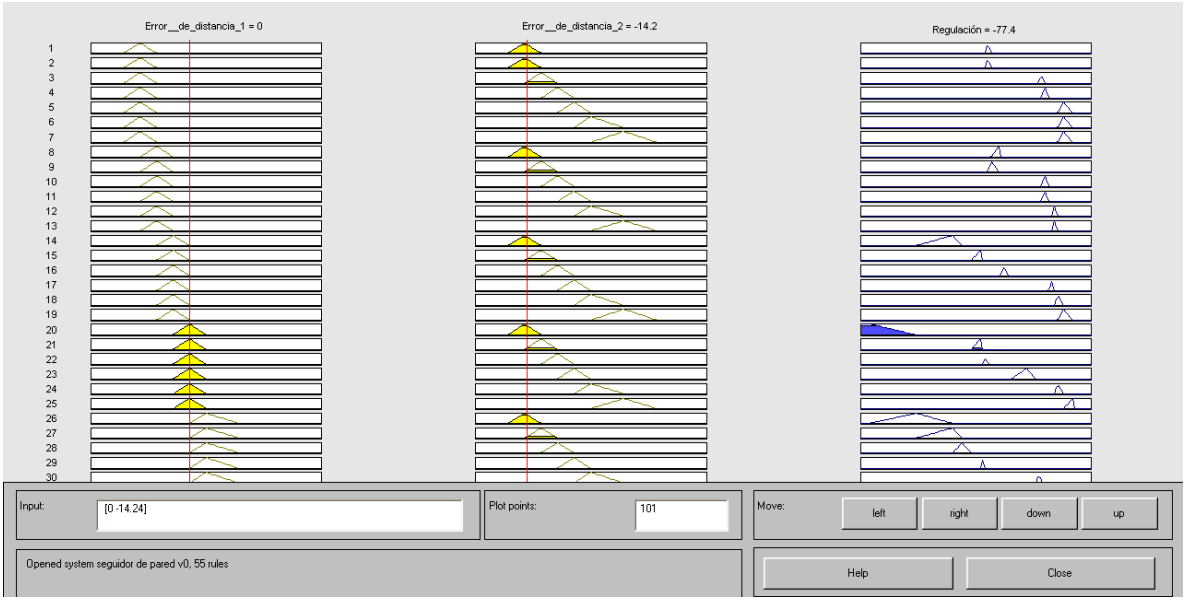


Figura 60. Gráfica de simulación #4

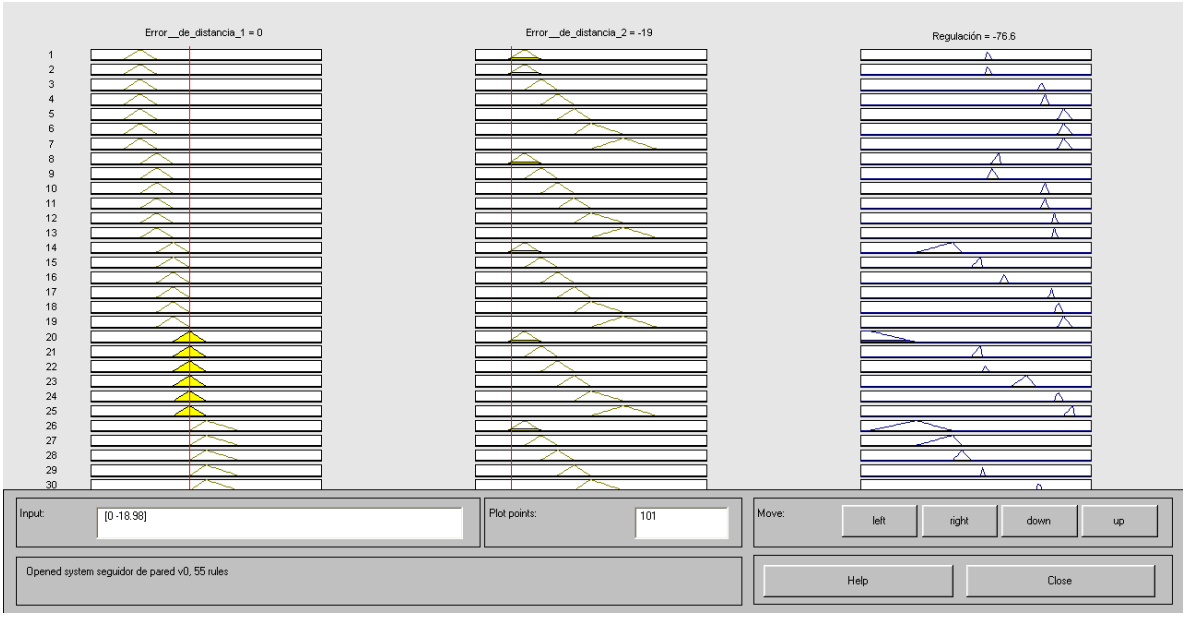


Figura 61. Gráfica de simulación #5

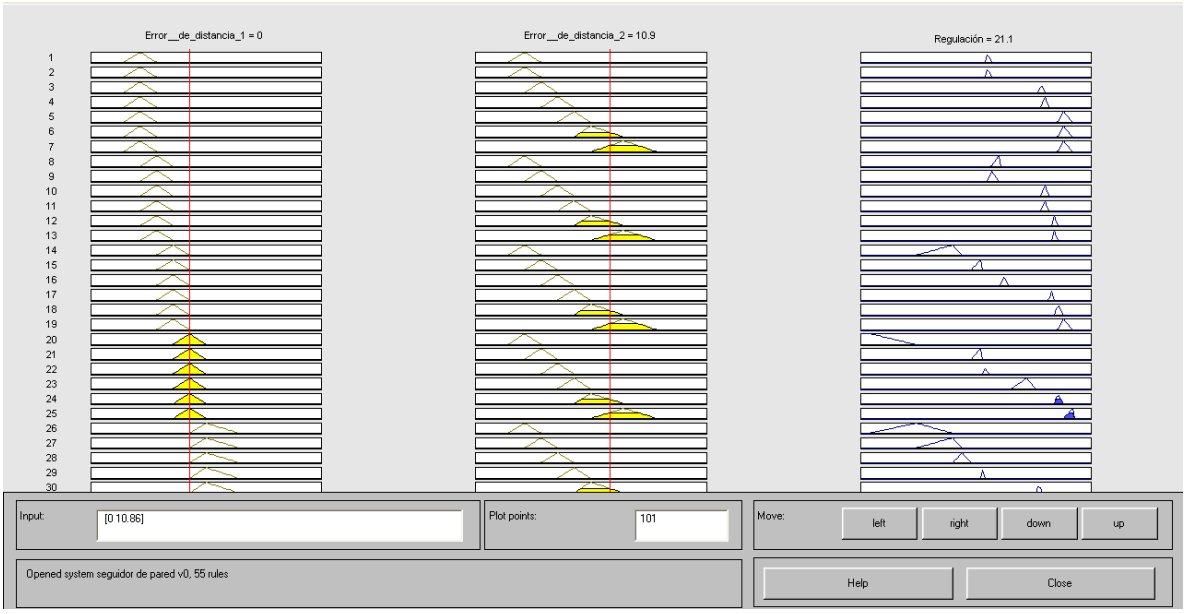


Figura 62. Gráfica de simulación #6

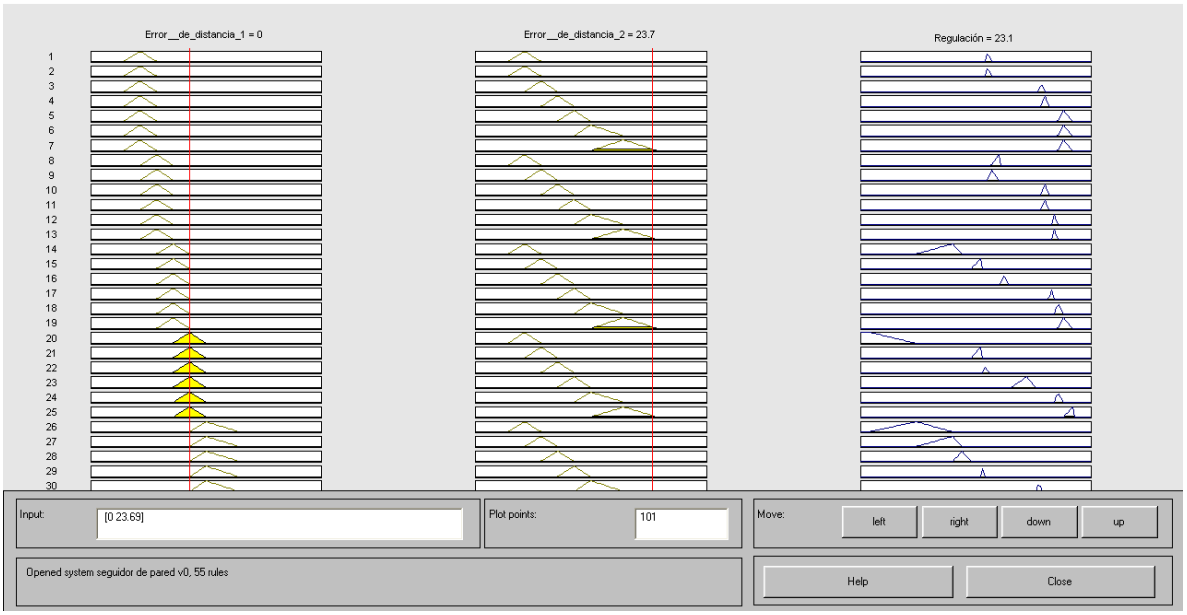


Figura 63. Gráfica de simulación #7

Figura 64. Gráfica de simulación #8

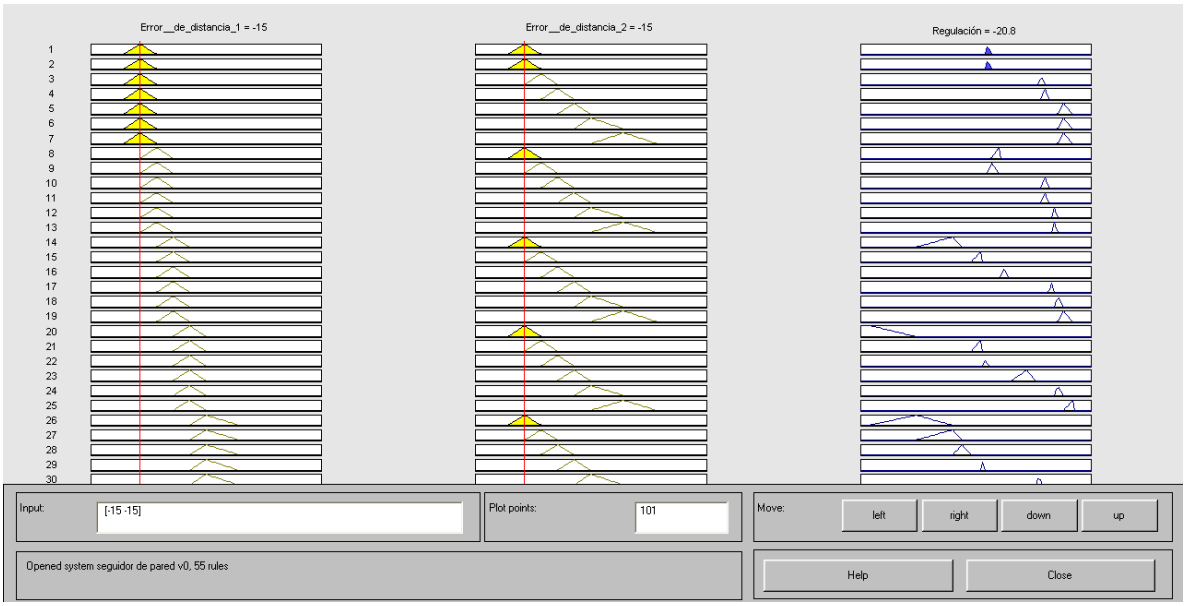


Figura 65. Gráfica de simulación #9

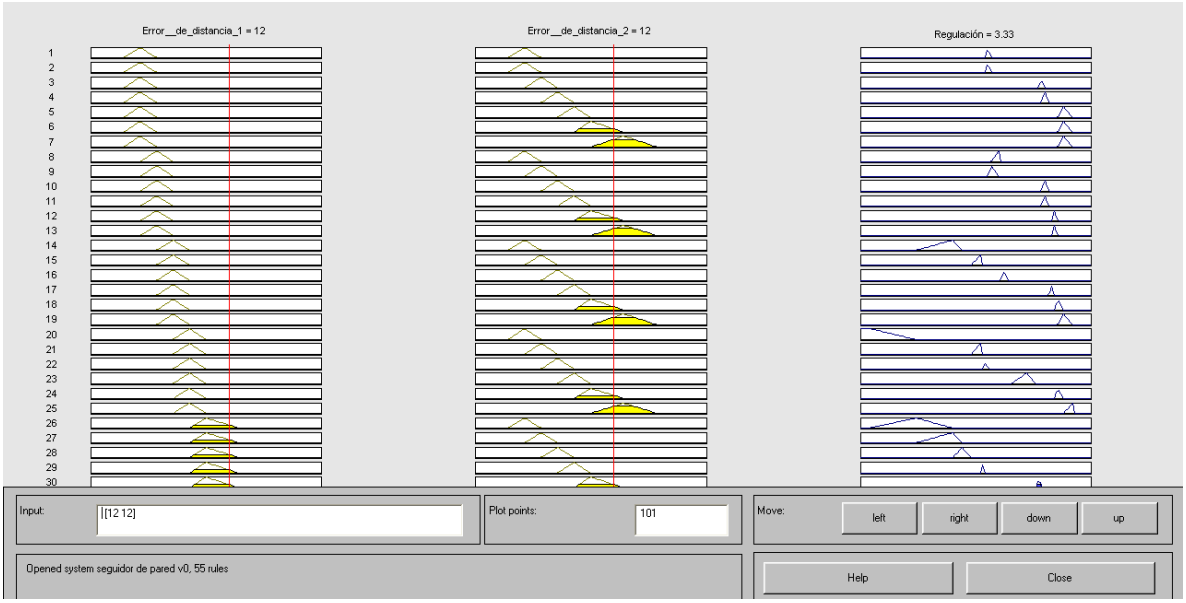


Figura 66. Gráfica de simulación #10

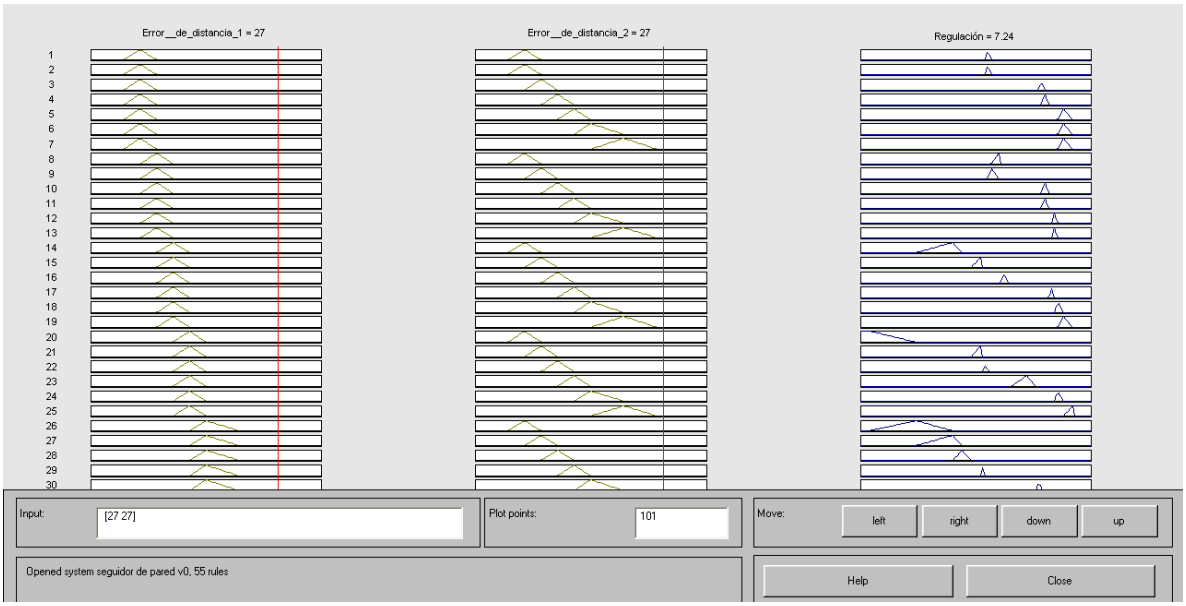


Figura 67. Gráfica de simulación #11

5.8.2.1. Análisis de graficas:

Se puede observar en las graficas que el comportamiento de la variable "regulación", para cada uno de los casos es el esperado según la lógica y la tabla de combinaciones empleada (ver tabla 23).

Análisis:

Durante la elaboración e implementación del programa de este seguidor de pared. Se fue cada vez haciendo más eficiente el programa. Por ejemplo se borraron combinaciones de reglas, que se observó en el entorno real no suceden, eliminando así líneas de código innecesarias; lo que finalmente se ve reflejado en mejor tiempo de ejecución.

El robot solo puede seguir paredes totalmente planas. Si tiene algún borde, hay posibilidad de que no funcione de acuerdo a lo esperado, esto es por la ubicación física de cada sensor, al ser las paredes con borde, el sensor detectará un corrimiento con respecto a su alineación. Esto quiere decir que cuando está alineado con la pared, este registrará una diferencia entre la lectura del sensor 1 con respecto al sensor 2, lo que le dificulta en sobremanera cumplir su objetivo (seguir el contorno de la pared).

En algunos entornos muy cerrados como pasillos o callejones pequeños (como el doble de ancho que el robot) ocurre mucha interferencia por el eco en las paredes, por lo que se le agregó un comportamiento que luego de varios “loops”, retrocede y gira, para salir del lugar o entorno. Por ejemplo luego de que el robot se encuentra con un objeto a una distancia no permitida frente a él, retrocede, y continua normal, pero si esta situación se repite más de 3 veces, retrocede y gira varias veces, de tal forma que pueda salir del lugar que le impide tener correctas lecturas, así puede reaccionar adecuadamente cuando se encuentra recibiendo ecos falsos, de lugares muy confinados.

De forma general se considera que esta aplicación funcionaria mejor con otro principio de funcionamiento de sensor, como el óptico.

6. Conclusiones.

Con el K.R.O.P.E.L (Kit Robótico Pedagógico De LEGO) se consiguió un ambiente de aprendizaje con herramientas básicas de hardware y software que permiten crear y concebir aplicaciones importantes en el ámbito de la robótica móvil en ambientes reales, esto se logró combinando los planteamientos de la robótica pedagógica con el uso del kit educativo de lego NXT.

Con K.R.O.P.E.L se observan los principales y clásicos diseños de la robótica móvil, a partir de un modelo tipo triciclo, el cual tiene la flexibilidad suficiente para permitir realizar cambios a la estructura física, y así obtener diferentes diseños finales con pequeñas modificaciones al modelo de la base principal.

Los diferentes diseños presentados son:

- Seguidor de pared.
- Seguidor de luz.
- Seguidor de línea.
- Pinza Sujetadora.

Con estas aplicaciones se logra obtener un ambiente natural mediante el cual se incentiva al estudiante a iniciar y/o ahondar en el campo de la robótica, haciendo el camino del aprendizaje más entretenido y práctico, así como también se le brinda al docente las herramientas necesarias para orientar a los estudiantes en el ámbito real, permitiendo con esto comprobar lo necesario de la praxis, sin desmeritar lo importante de la teoría.

El desarrollo de la guía acompaña al estudiante, desde el diseño mecánico hasta el diseño lógico y la programación. Para el diseño físico se ofrecen guías de construcción paso a paso, donde se indican cómo construir cada una de las aplicaciones nombradas, lo que facilita el armado y construcción de cada una de las

aplicaciones presentadas, también sirven para orientar al estudiante sobre cómo crear sus propios diseños. En la parte de software se logró crear guías de instalación, configuración del software, descripción de las principales clases y métodos de leJOS; facilitando la tarea del aprendizaje y comprensión del mismo, permitiendo a cualquier estudiante de ingeniería electrónica (o afín) con conocimientos básicos en programación en java, aprender a crear aplicaciones para lego NXT.

Se creó un esquema general de la estructura de un controlador difuso, con lo que se orienta al estudiante acerca de la lógica utilizada para la creación de una aplicación con lógica difusa, para afianzar el proceso de diseño de un controlador difuso, se presentó un ejemplo de un controlador difuso con el K.R.O.P.E.L en donde se integra el diseño teórico, las simulaciones con software, y finalmente la implementación.

En el proceso de diseño y elaboración del controlador difuso, se logró:

- Estudiar correctamente el proceso a modelar, esto influyó en el comportamiento adecuado del controlador, puesto que entre más cercana sea la representación de los conjuntos difusos en relación con el proceso, este realizará un mejor control.

- Se resalta la importancia de la fase de simulación en software (matlab), debido que esto reduce considerablemente el tiempo de detección y corrección de errores, antes de llegar a la etapa de la implementación.

Se logró crear un controlador P.D para el control de la velocidad del motor con el fin de comparar el sistema clásico con el control difuso, durante el proceso de creación y comparación de ambos controladores se observo lo siguiente:

- Con el control difuso, se corrobora la respuesta rápida y suave del robot a cambios en el ambiente poco estructurado.

- Los planteamientos lógicos son más sencillos de realizar en el proceso de modelado puesto que son sintaxis que el hombre realiza en su vida cotidiana.

- En el controlador difuso es más complicado cambiar los set points. Puesto esto implica nuevamente crear los conjuntos desde un principio.

- Con el control P.D se observo la ventaja en cuanto al cambio del set point, era un poco más sencillo, pero las respuestas y acciones eran un poco más bruscas.

Finalmente se logró entregar una guía donde se muestra como crear un controlador difuso, y un controlador P.D con el fin de que el estudiante y docente realicen pruebas y aplicaciones para determinar cuándo es necesario emplear determinado tipo de control.

7. Recomendaciones y trabajo futuro.

Crear una serie de demostraciones para realizar pruebas orientadas a incentivar el interés en la robótica, orientadas a estudiantes de bachillerato, con el fin de promocionar la robótica y la ciencia en los colegios. Ya que un robot puede ser una herramienta poderosa al momento de usarlo para enseñar diferentes áreas, aparte de la robótica, tales como matemática, física, informática, entre otras.

Ahondar en metodologías de enseñanza aprendizaje, con el fin de desarrollar guías metodológicas para la robótica orientada a docentes, y alumnos.

Diseñar un brazo robótico para modelar los utilizados en las industrias en los procesos de manufactura. Que permita la ejecución de pruebas y tareas sencillas de control, de tal forma que se pueda aprender, enseñar y observar el principio de funcionamiento de un brazo robótico en general. Debido a que en el proyecto solo se trató con robots móviles.

7.1. Realizar un estudio de campo donde se compruebe el impacto del proyecto en el grupo aplicado, puede ser mediante encuestas. El análisis de estas encuestas puede ser usado como retro-alimentación para mejorar las guías de laboratorio (ver ítem 5.2).

Crear aplicaciones más interactivas en el ladrillo.

Crear un programa que permita al robot “aprender” de su entorno y modifique sus bases del conocimiento, para así tener un robot que sea capaz de funcionar en entornos menos estructurados.

Realizar un programa con la técnica de control difuso, para lograr y calcular trayectorias desde el punto donde se encuentra el robot y la bola de IR, usando par ello IR Seeker.

Implementar algún software o aplicación que sea capaz de visualizar en un PC la trayectoria recorrida por el robot. Con esta herramienta se podría analizar de mejor manera el comportamiento de las aplicaciones ejecutadas en el robot.

Implementar y mejorar un programa de control de distancia mediante PID que involucre la componente integral y mejorar la sintonización de éste.

Bibliografía.

Libros consultados.

[ANGELES-03] Angeles, J. (2003). Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms, New York (U.S.): Springer-Verlag.

[BARRIENTOS-97] Barrientos, A. (1997). *Fundamentos de Robótica*, España: McGraw Hill.

[BREÑA-09] Breña, M. J. C. (12-Apr-2009). E-Book *Develop leJOS Programs Step by Step*. Extraído del sitio: http://www.juanantonio.info/lejos-ebook/.

[BREÑA-08] Breña, M. J. C. (24-Ago-2008). *Using I2C with Java leJOS. Extraído del sitio:* http://www. juanantonio.info/jab_cms.php?id=146.

[CABRERA-96] Cabrera, O. L. (1996, Diciembre 15). La Robótica Pedagógica. *Soluciones Avanzadas*, 40, 1-10.

[CHEN-01] Chen, G. (2001). *Introduction to fuzzy sets, fuzzy logic, and fuzzy control systems.* U.S.: CRC Press LLC.

[ECKEL-02] Eckel, B. (2002). *Piensa en Java.* Madrid (España): Pearson Educación S.A.

[FENG-94] Feng, L., Borenstein, J. y Everett, H. R. (1994). *Artículo Técnico: Sensors and Methods for Autonomous Mobile Robot Positioning*. Michigan (U.S): Universidad de Michigan.

[FERRARI-02] Ferrari, M., Ferrari, G., Hempel, R. (2002). *Building robot with LEGO MINDSTORMS*, U.S.: Syngress Publishing Inc.

[GARCÍA-00] García de Jalón. J. et al. (2000). *Aprenda Java Como Si Estuviera En Primero*. San Sebastián (España): TECNUN.

[GASPERI-07] Gasperi, M., Philippe H., Isabelle H. (2007). *Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level*. U.S.: Technology in Action Press™.

[GIAMARCHI-01] Giamarchi, F. (2001). *Robots Móviles: Estudio y Construcción.* Madrid (España): Thomson Paraninfo.

[GOEBEL-03] Goebel, G. (2003). *An Introduction to Fuzzy Control Systems*. Extraído el 1 de septiembre de 2009 desde el sitio: http://www.faqs.org/docs/fuzzy/.

[HERNÁDEZ-03]. Hernández, R. S, Fernández, C. C, Baptista P.L. (2003). *Metodología de la Investigación*. México: McGraw Hill. pp. 58–71.

[HOLLANS-04] Hollans, J.M., (2004). *Designing Autonomous Mobile Robots*. Maryland(U.S.): Elsevier Inc.

[LEMAY-99] Lemay, L. (1999). *Aprende Java 2 en 21 días.* México: Prentice-Hall.

[MIGLINO-99] Miglino O., Hautop Lund H. y Cardaci M. (1999). *La Robótica como Herramienta para la educación*. Extraído el 1 de junio de 2008 del sitio: http://www.mip. sdu.dk/~hhl/.

[MILSANT-72] Milsant, F. (1972). *Servosistemas Lineales*. Tomo I. Barcelona (España): Editores Técnicos Asociados.

[MUÑOZ-06] Muñoz, N., Andrade, C. y Londoño, N. (2006). Diseño y Construcción de un Robot Móvil Orientado a la Educación. *Resumen Tesis del Grupo de Investigación en Robótica y Áreas Afines*, Universidad de Antioquia.

[MUÑOZ-06] Muñoz, D, N., (2006). *Diseño y Construcción De Un Robot Móvil Orientado a la Enseñanza e Investigación*. Colombia: Universidad del Norte, Nº 19.

[MURPHY-00] Murphy, R. (2000). *Introduction to AI Robotics.* Massachusetts (U.S.): Cambridge University Press.

[OLLERO-01] Ollero, A. (2001). *Robótica, Manipuladores y Robots Móviles*. Barcelona (España): Marcombo Boixareu Editores.

[OGATA-98] Ogata, K. (1998) *Ingeniería de control moderna*. México: Pearson Editorial.

[PALACIOS-06] Palacios, E., Remiro, F. y López, L. (2004). *Microcotroladores PIC16f84, Desarrollo de Proyectos*, Segunda Edición, México: Alfaomega Grupo Editor.

[PRESTON-05] Preston, S. (2005). *The Definitive Guide to Building Java Robots*. New York (U.S.): Apress®.

[RUIZ-VELASCO-89] Ruiz-Velasco, E. (1989). Ciencia y Tecnología a través de la Robótica Cognoscitiva. Perfiles Educativos.

[RUIZ-VELASCO-02] García, R., López, D. U., Valencia, J. M. y Ruiz-Velasco, E. (2002). *Herramientas Lúdico-tecnológicas para la Enseñanza, los Mecanismos Robóticos y sus Aplicaciones en el Aula.* Trabajo presentado en el VIII Congreso Internacional de Informática en la Educación INFOREDU, Febrero, La Habana – Cuba. pp. 1-ss.

[RUIZ-VELASCO-08] Ruiz-Velasco, E., García, J. V. y Rosas, L (s.f.). Robótica Pedagógica Virtual para la Inteligencia Colectiva. Extraído el 28 de septiembre de 2008 desde el sitio:

http://www.virtualeduca.info/ forumveduca/index.php?option=com_content&task=view&id=108&Itemid=26

[SÁNCHEZ-08] Sánchez, M. M. (s.f.). *Ambientes de Aprendizaje con Robótica Pedagógica*. Extraído el 10 de Agosto de 2008 desde: http://guaica.uniandes.edu.co:5050/ dspace/bitstream/1992/ 369/1/mi_1253.pdf.

[SÁNCHEZ-03] Sánchez, M. M. (2003). *Implementación de Estrategias de Robótica Pedagógica en las Instituciones Educativas*. Extraído el 3 de julio de 2008 del sitio http://www.eduteka.org/ RoboticaPedagogica.php.

[TORRES-02] Torres, F. (2002). *Robot y Sistemas Sensoriales.* Madrid (España): Prentice Hall.

[VALLEJO-04] Vallejo, E. (2004). Tesis Doctoral: *Aprendizaje Evolutivo de Reglas Fuzzy en un Sistema Clasificador Modificado para Control de Agentes Móviles.* Universidad Politécnica de Valencia – España.

[VERGEL-07] Vergel, C.G. (2007). *Metodología: Un Manual para la Elaboración de Diseño y Proyectos de Investigación*, Barranquilla (Colombia): Mejoras Ltda. Editores.

Sitios web consultados.

[WEB01] Tutorial leJOS NXT, extraído el 10 de diciembre de 2008 del sitio: http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm.

[WEB02] LEGO® MINDSTORMS®. (2009). Traducción realizada del documento *LEGO® MINDSTORMS® NXT Hardware Developer Kit*, extraído el 7 de marzo de 2009 del sitio web: http://mindstorms.lego.com/Overview/ NXTreme.aspx.

[WEB03] Pagina oficial de LEGO NXT: http://mindstorms.lego.com/

[WEB04] Software leJOS, descargado del sitio: http://lejos.sourceforge.net/p_technologies/nxt/nxj/downloads.php.

[WEB05] Software libusb.win32, descargado del sitio: http://sourceforge.net/project/showfiles.php?group_id=78138.

[WEB06] Software Eclipse, descargado del sitio: http://www.eclipse.org/downloads/

[WEB07] Software Java JDK, descargado del sitio: http://www.java.com/en/download/manual.jsp.

[WEB08] Hitechnic. (2009). Información Técnica extraída del sitio web: http://www.hitechnic.com/.

[WEB09] LEGO® MINDSTORMS® (2009). *Información Técnica de Motores Lego NXT*. Citado en Hurbain, P. (2009). Extraído el 6 de junio de 2009 desde el sitio: http://www.philohome. com/nxtmotor/nxtmotor.htm.

[WEB10] LEGO® MINDSTORMS® (2009). Tienda LEGO Education. http://www.legoeducation.us/store/default.aspx?CategoryID=178&by=9&c=1.

[WEB11] leJOS API, extraído del sitio: http://lejos.sourceforge.net/nxt/nxj/api/index.html

[WEB12] Información Técnica MLCad, extraída del sitio: http://www.ldraw.org/Downloads-req-viewdownload-cid-1.html.

[WEB13] Fuzzy Logic Toolbox. Extraído el 10 de septiembre de 2009 desde el sitio: http://www.mathworks.com/access/helpdesk/help/toolbox/ fuzzy/.

[WEB14] Bobcat. (2009). *Imagen Cargadora Bobcat*. Extraído el 30 de octubre de 2009 del sitio:http://www.bobcat.com/publicadmin/ getFile.do?id=23739.

[WEB15] LEGO® MINDSTORMS® (2009). Software NXT-G extraído el 1 de enero de 2009 desde el sitio: http://mindstorms.lego.com/ overview/NXT_ Software.aspx.

[WEB16] Robotc. (2009). Extraído el 1 de enero de 2009 desde el sitio: http://www.robotc.net/content/lego_over/lego_over.html.

[WEB17] Next Byte Codes & Not eXactly C. (2008). Extraído el 1 de enero de 2009 desde el sitio: http://bricxcc.sourceforge.net/nbc/

[WEB18] NXT-Python. (2009). Extraído el 4 de febrero de 2009 desde el sitio: http//code.google.com/p/nxt-python/

[WEB19] Gostai. (2009). Extraído el 5 de febrero de 2009 desde el sitio: http://www.gostai.com/ tutorials.html

[WEB20] nxtOSEK Project. (2009). Extraído el 5 de febrero de 2009 desde el sitio: http://lejos-osek.sourceforge.net/whatislejososek.htm

[WEB21] leJOS, Java for Lego Mindstorms. (2009). Extraído el 5 de febrero de 2009 desde el sitio: http://lejos.sourceforge.net/nxt/nxj/ tutorial/Preliminaries/Intro.htm.

[WEB22] National Instruments. (2009). Extraído el 5 de febrero de 2009 desde el sitio: http://www.ni.com/pdf/manuals/372573a.pdf

[WEB23] Microsoft Developer Network, msdn. (2009). Extraído el 6 de febrero de 2009 desde el sitio: http://msdn.microsoft.com/en-us/library/bb483027.aspx.

[WEB24] The MathWorks Inc. (2009). Extraído el 6 de febrero de 2009 desde el sitio: http://www.mathworks.com/programs/mindstorms/.

[WEB25] The MathWorks Inc. (2009). *Fuzzy Logic Toolbox*. Extraído el 10 de septiembre de 2009 desde el sitio: http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/.

ANEXOS

ANEXO A

Laboratorio 1
Levantamiento de la plataforma de desarrollo en leJOS.

1. Objetivos.

1.1. Generales.

Levantar la plataforma de trabajo leJOS para NXT.

1.2. Específicos.

- Conocer las diferentes plataformas de desarrollo de programación de alto nivel para LEGO Mindstorms NXT.
- Instalar y configurar todo el software necesario para desarrollar programas en Java para LEGO Mindstorms NXT.
- Descargar el primer programa al ladrillo NXT usando eclipse y lejos.

2. Metodología.

El día de la realización del Laboratorio el estudiante debe haber leído y analizado la presente guía.

Con el conocimiento adquirido de la lectura el estudiante estará en capacidad de seguir los pasos para el levantamiento de la plataforma de desarrollo, y posteriormente cargar su primer programa en el ladrillo de LEGO NXT, el cual, se encuentra al final de la guía, este programa tiene como propósito comprobar el éxito de la instalación.

Se deberá entregar un informe con las normas ICONTEC actuales en la próxima clase.

3. Materiales.

- Ladrillo LEGO NXT.
- Cable conector NXT (cualquier tamaño).
- Computador con Windows XP o en su defecto que dicho computador posea una maquina virtual del sistema operativo mencionado.
- 6 Batería AA (preferiblemente recargables).
- Java JDK.
- Eclipse.
- leJOS.
- libusb.
- Driver lego NXT.

4. Introducción.

El firmare que trae el ladrillo originalmente funciona bajo el software de lego, NXT-G. Es un software de licencia privada (esto quiere decir que hay que pagar por obtenerlo) Este programa es adecuado para programación básica. Presenta algunas desventajas, como que sus programas pueden ser muy largos comparados con otros lenguajes de programación, los programas se toman más tiempo en ser descargados al ladrillo, cuando se crean programas extensos tiende a colapsar y perder datos no guardados, el software NXT-G generalmente corre lento, incluso en computadores potentes.

Existen actualmente múltiples alternativas en cuanto a la escogencia del software con el que se quiera programar el ladrillo NXT de Lego, a continuación se presentan los más conocidos.

4.1. Software comercial:

- NXT-G: es el software basad en LabVIEW con el que viene con el kit de LEGO NXT, este software es adecuado para programadores principiantes (LEGO® MINDSTORMS®, 2009).

- ROBOTC: es un poderoso lenguaje de programación basado en C con el entorno de Windows para escribir y compilar programas (Robotc, 2009).

4.2. Software de libre uso:

- NBC/NXC (Next Byte Codes/Not eXactly C): NBC es un simple lenguaje basado en el lenguaje ensamblador; NXC es un lenguaje similar al lenguaje C, como una simplificación de C, diseñado especialmente para los robots de Lego. Cuenta con el Bricx Command Center (BricxCC), herramienta para escribir los programas, descargarlos al robot (Next Byte Codes & Not eXactly C, 2009).

- NXT-Python: es un driver/interface de Python para los robots Lego Mindstorms NXT (NXT-Python, 2009).

- URBI (Universal Real-time Behavior Interface): es un lenguaje de programación diseñado para robots de cualquier tipo (Gostai, 2009).

- leJOS OSEK: es una plataforma de código abierto para LEGO MINDSTORMS NXT con lenguaje de programación “C”, consta del controlador de dispositivo de LEJOS NXJ C / Código fuente ensamblador, Toppers / ATK (antes conocido como Toppers / OSEK) y Toppers / JSP código fuente del sistema operativo en tiempo real que incluye ARM7 (ATMEL AT91SAM7S256) parte específica de puerto, un código pegamento para hacerlos trabajar juntos (nxtOSEK Project, 2009).

- leJOS NXJ: es el remplazo del firmware para el NXT que contiene una maquina virtual de Java , la cual, incluye todas las clases en la API de NXJ

como también herramientas usadas para descargar el código de programación al ladrillo NXT (leJOS, Java for Lego Mindstorms, 2009).

4.3. Software comercial y de libre uso (plugins):

- LabVIEW: es posible programar directamente los Lego Mindstorms desde LabVIEW. Utiliza un lenguaje gráfico (National Instruments, 2009).
- Microsoft Robotics Developer Studio: entorno basado en Windows para el control de robots (Microsoft Developer Network msdn, 2009).
- MATLAB y Simulink: MATLAB es un entorno de programación y de computación numérica. El MINDSTORMS NXT Toolbox para MATLAB, se trata de un proyecto en código abierto. Simulink utiliza las librerías de MATLAB para el desarrollo de los programas (The MathWorks Inc., 2009).

A continuación se presenta una tabla comparativa que contiene los principales software de programación y sus ventajas.

Software	Ventajas
NXT-G	NXT-G es fácil de instalar en Windows XP y Vista, y está soportado también para Mac OS X.
	NXT-G puede transferir datos vía Bluetooth o por el cable USB incluido.
	NXT-G es fácil de usar, ambiente grafico de tomar y poner.
	Los gráficos incluyen los hilos que muestran el dato de flujo bloque a bloque.

ROBOTC	Es ideal para usuarios principiantes y avanzados.
	Permite la educación en programación avanzada, ingeniería y sistemas embebidos.
	Su IDE incluye revisión de sintaxis en tiempo real, compilador y autocompletar funciones y variables ROBOTC es un excelente entorno que puede programar diferentes tipos de robots a demás del LEGO NXT, su principal desventaja es que es un software pago.
	Su colección de instrucciones y los programas de ejemplo es incomparable en el mundo de Mindstorms.
NBC/NXC	Gratuito
	Permite comunicación vía Bluetooth y vía USB
	Utiliza BricxCC, la cual permite escribir los programas, descargarlos al robot, comenzar y terminarlos, navegar en la memoria flash del NXT, convertir archivos de sonido para usar con el ladrillo.
NXT-Python	Gratuito.
	Multiplataforma.
	Comunicación Bluetooth y USB.

URBI	Gratuito
	Su SDK es de código abierto (distribuido y desarrollado libremente), esta soportado por GNU GPL y es independiente del robot utilizado.
leJOS OSEK	Gratuito
	Entorno de programación en ANSI C y C++
	API de C para sensores y accionamientos finales
	Rápida ejecución y menor consumo de memoria
leJOS NXJ	Gratuito
	Usa el lenguaje estándar industrial de Java
	Proporciona lenguaje de programación orientada a objetos.
	Es un proyecto de fuente abierta con muchos contribuidores.
	Permite escoger un profesional IDE como Eclipse o Netbeans con soporte de sintaxis y otras características.
	Tiene soporte para las plataformas de Windows, Linux, Mac OS X.

	Mucho más rápido que el NXT-G.
	Tiene completo soporte para Bluetooth.
	Proporciona control de motor de alta precisión.
	Tiene soporte para navegación avanzada.
	Proporciona clases de comportamiento que soportan la arquitectura para el desarrollo de robots con comportamientos complejos.
	Apoyo a sensores externos, como los Sharp por ejemplo.
	Soporta monitoreo remoto y seguimiento del programa desde el PC.
	Establece funciones de trigonometría y otras funciones matemáticas.
	Apoya la interfaz gráfica.
	Apoya multi-hilos.
	Ofrece docenas de programas de ejemplos, entre otras ventajas.
LabVIEW	Ofrece licencias para uso no profesional gratuitas.

	Con LabVIEW y LabVIEW Toolkit para Lego Mindstorms NXT es posible crear nuevos bloques para el NX-G.
Microsoft Robotics Developer Studio	Ofrece licencias gratuitas.
	Permite hacer simulaciones en 3D.
	Dirigido tanto para la educación, para aficionados como a desarrolladores comerciales.

4.4. ¿Por qué leJOS?

leJOS NXJ es un entorno de programación de Java para LEGO MINDSTORMS NXT, el cual, permite programar los robots de LEGO en Java. Esto es de gran ventaja debido a que se trabaja con un lenguaje de programación de alto nivel orientado a objetos, trae consigo todas sus ventajas, como es el polimorfismo, uso de clases, reutilización de clases, siendo una de las ventajas más significativa el no tener que comenzar los programas desde cero, pues todos los programas se basan en clases ya existentes que proveen java y leJOS, así como también el usuario puede crear sus propias clases, para luego ser reutilizadas (revisar concepto de reutilización de clases en cualquier libro de P.O.O),la sintaxis es la misma que utiliza un programador o desarrollador en sus proyectos con java. Java tiene la gran característica de manejar la memoria por sí solo, es decir que el programador no tiene que preocuparse por los espacios de memoria que se estén o no utilizando, pues éste tiene un recolector de basura, que borra las variables, datos e información que deja de ser necesaria en el programa.

Además leJOS es un programa de código abierto, lo cual, permite que sea estudiado y mejorado por los aportes de sus usuarios, cuentan con un foro que permite el contacto directo con sus creadores y desarrolladores. Actualmente está soportado en tres sistemas operativos, Microsoft Windows, Linux y MAC OS X. Trabaja con eclipse – que también es un software gratuito – esto proporciona todas las ventajas de las potencialidades de este entorno en cuanto a su manejo, al manejo del código, etc.

También cuenta con librerías creadas por los desarrolladores para el manejo de dispositivos, lo cual, facilita aún más la programación por medio de la reutilización de código.

leJOS NXJ presenta las mismas ventajas de leJOS OSEK y proviene de los mismos creadores, lo que los diferencia es el lenguaje de programación; la escogencia entre el uno y el otro depende de las preferencias del programador. Otro de los lenguajes con el que es posible programar el ladrillo es URBI (lenguaje de programación de

alto nivel), el cual, es compatible con MATLAB, C++ o Java pero cuenta con poco soporte y es poco popular.

5. Practica.

5.1. Instalando Lejos Y Configurando Eclipse.

Para la realización de este punto de la guía se requieren los siguientes programas.

- Java JDK
- Eclipse
- leJOS
- libusb
- Driver lego NXT

Tanto el JDK y el JRE se pueden descargar desde internet buscando en cualquier buscador web, o desde la pagina de sun microsystem de java.

El driver de lego se puede descargar desde el siguiente link:

http://mindstorms.lego.com/Support/Updates/

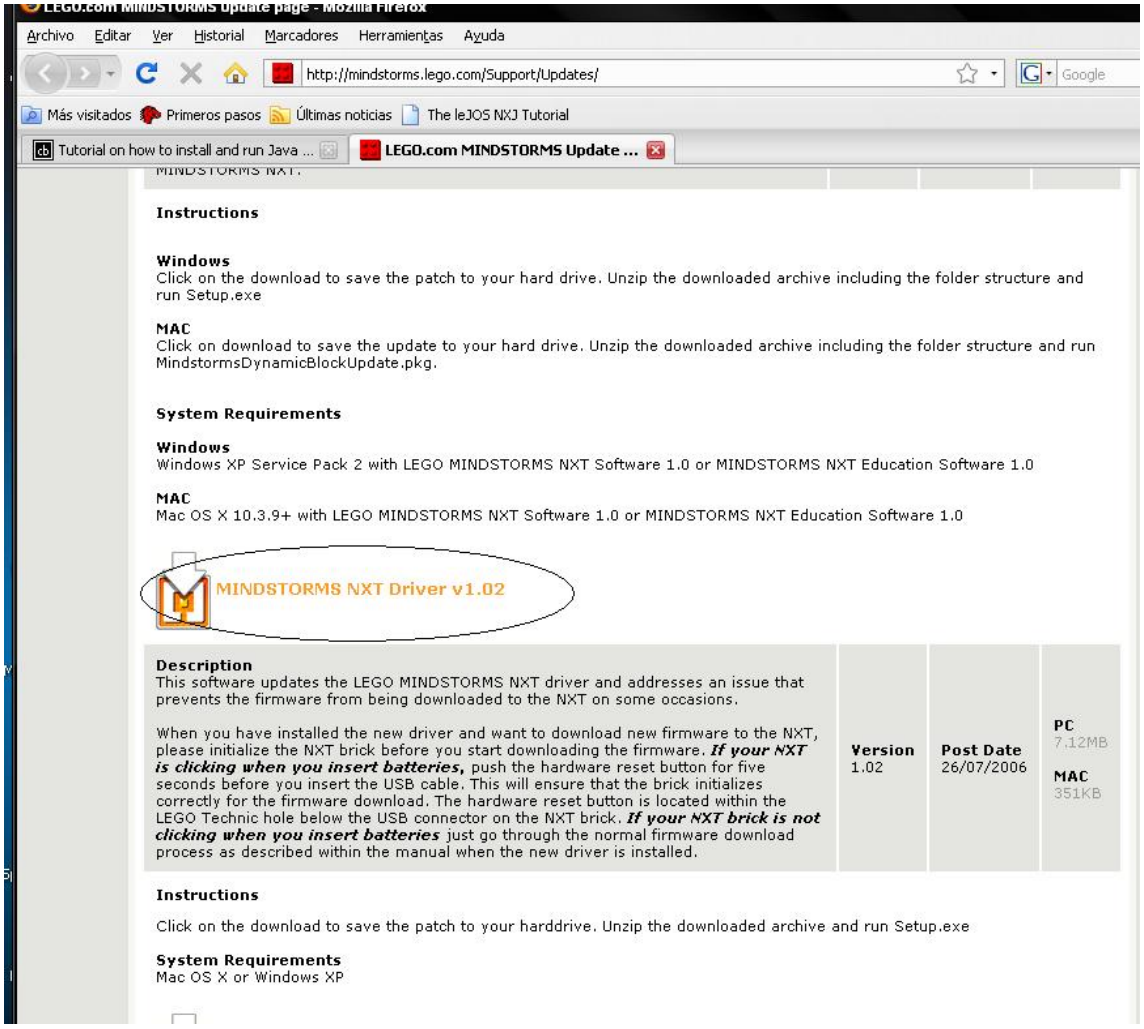


Figura 68. Pagina de descarga driver para lego NXT

Luego de descargarlo, descomprimir el archivo y dar click en setup.exe

5.2. Pasos para la instalación.

Figura 69. LEGO MINDSTORMS NXT Driver Software

Figura 70. Pantallazo final de instalación del software

Luego de instalarlo:

Conecta el ladrillo mediante el USB al ordenador.

Ve a mi PC, y dar click derecho en propiedades.

Figura 71. Propiedades del sistema

Estando ahí se le da click a Hardware y luego Administrador de Dispositivos debe aparecer uno que diga “lego devices”, como se muestra en la siguiente imagen:

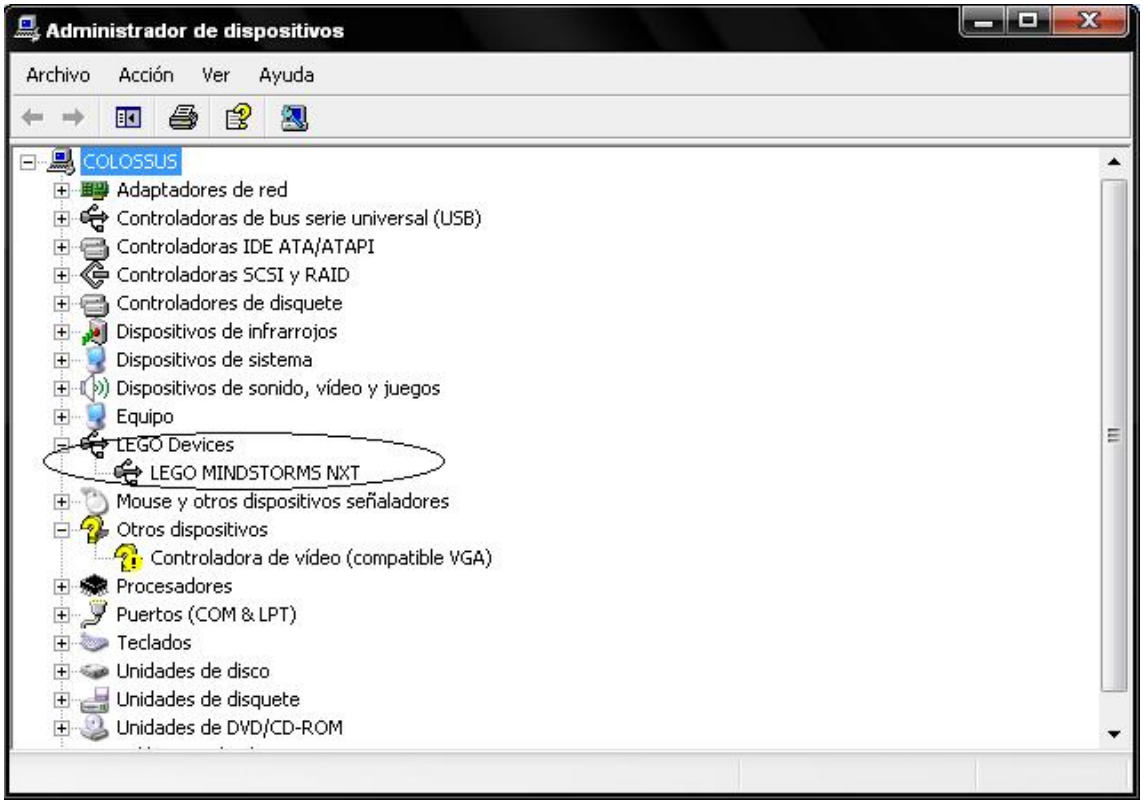


Figura 72. Administrador de dispositivos

5.2.1. Instalando Lejos.

Para descargarlo ir al link:

http://lejos.sourceforge.net/p_technologies/nxt/nxj/downloads.php

Ir a Mi PC, Disco Local (C:), y crear una nueva carpeta que se llame lejos_nxj en esta carpeta descomprimir el archivo descargado de la pagina de lejos.

Ahora configurar las variables de entorno del sistema, necesarias para que esté avise a eclipse sobre la existencia de la librería de lejos y que así pueda funcionar el programa.

Luego ir a Mi PC, dar click derecho y entrar a la pestaña opciones avanzadas y dar click en variables de entorno, como nuestra la figura 6.

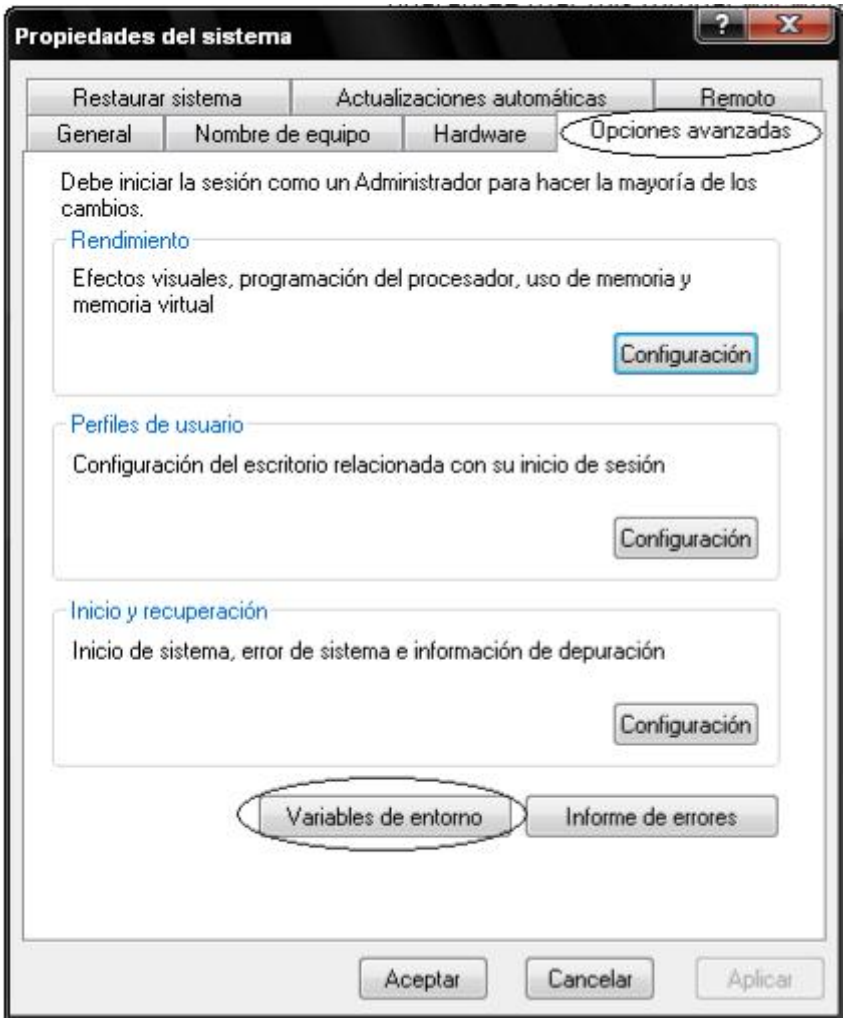


Figura 73. Configuración variables de entorno

En nueva ventana abierta buscar la variable de sistema Path y se da click en editar, agregar; %JAVA_HOME%\bin sin borrar las demás figura 42.

Figura 74. Modificar la variable Path

Ahora en variable de usuario se debe dar click en nueva. Otorgándole en nombre de variable; LEJOS_HOME. En la etiqueta valor de la variable, colocar la carpeta C:\lejos_nxj. Figura 43.

Figura 75. Variable de usuario para administrador

Finalmente ir a la variable path y dar modificar nuevamente y agregar; %LEJOS_HOME%\bin . El valor de la variable path quedaría algo similar a esto; Valor de path:

%SystemRoot%\system32; %SystemRoot%; %SystemRoot%\System32\Wbem; %JAVA_HOME%\bin; %LEJOS_HOME%\bin.

Aunque el valor de la variable path puede ser diferente en cada PC, acá la coloco en caso de que por error hayan borrado el anterior valor. Para verificar que todo este correcto, vamos a inicio, ejecutar escribimos cmd.

Nos debe aparecer un entorno de DOS Ahí se debe escribir lejosdl y dar click a la tecla enter, luego debe aparecer lo siguiente:

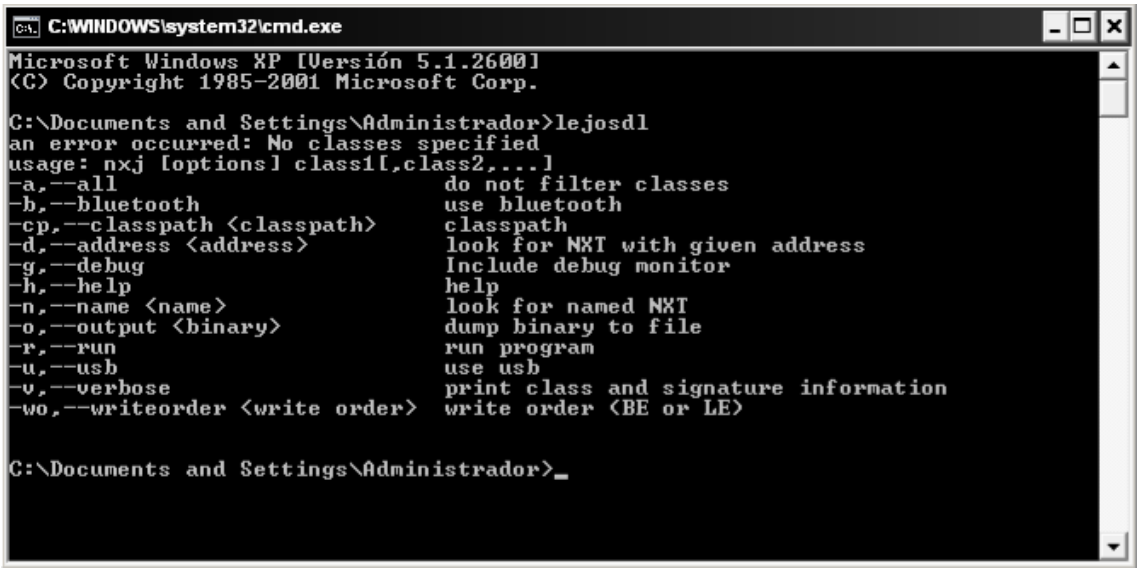


Figura 76. Entorno DOS para confirmar la instalación

Hay dos maneras de instalar el firmware en el ladrillo, la primera y más sencilla es simplemente cuando se instala leJOS, dar click a la opción OK cuando muestre instalar firma, lo siguiente es salir y debe aparecer leJOS en el ladrillo cuando se encienda.

La otra manera es acceder a la ventana de DOS y escribir el comando lejosfirmdl y aparece un mensaje que dice libus no ha sido instalado. Figura10.

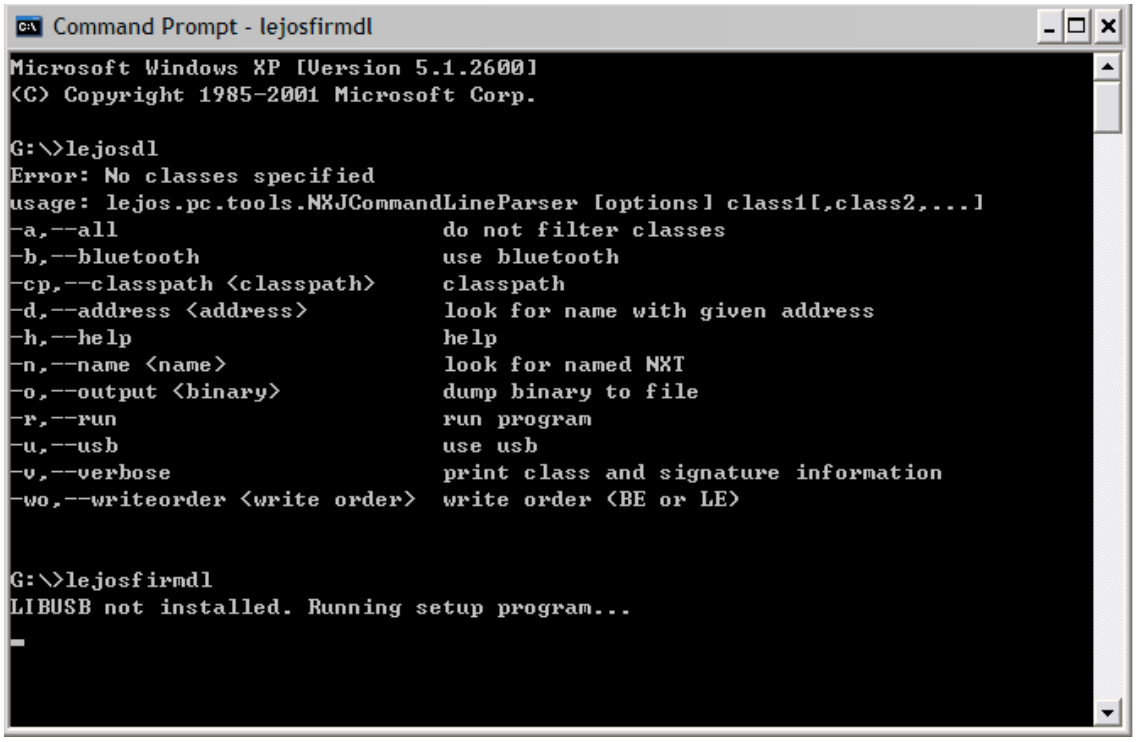


Figura 77. Comando lejosfirmdl

En este link se descarga:

http://sourceforge.net/project/showfiles.php?group_id=78138

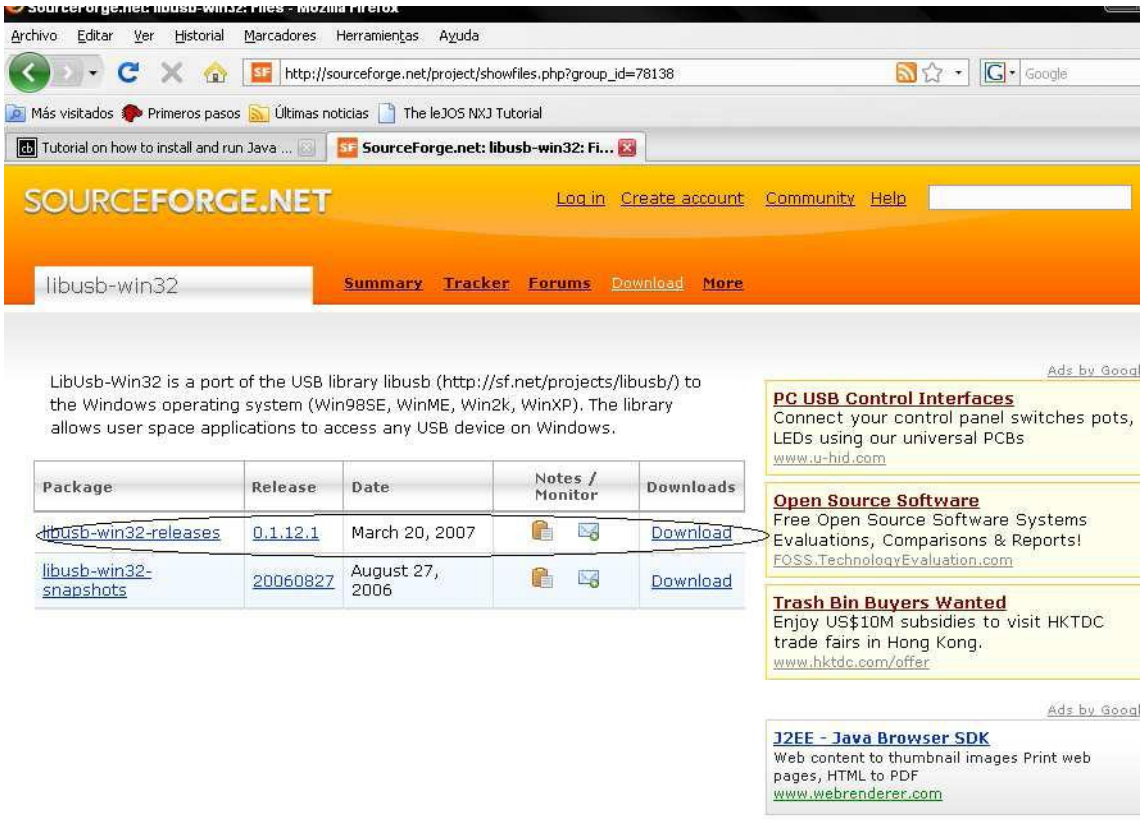


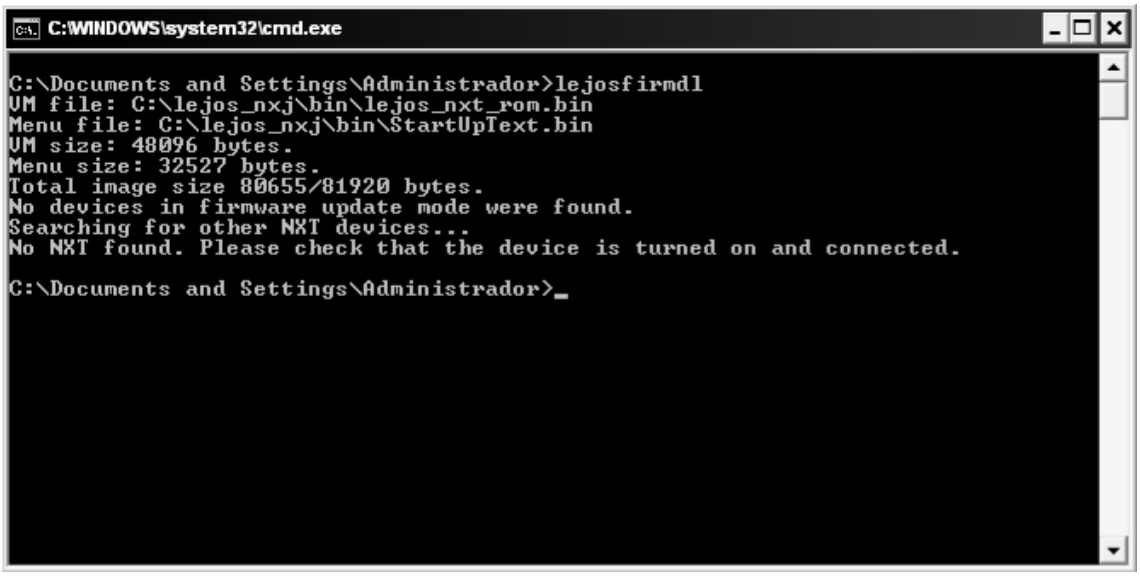
Figura 78. Pagina de descarga de libusb.win32

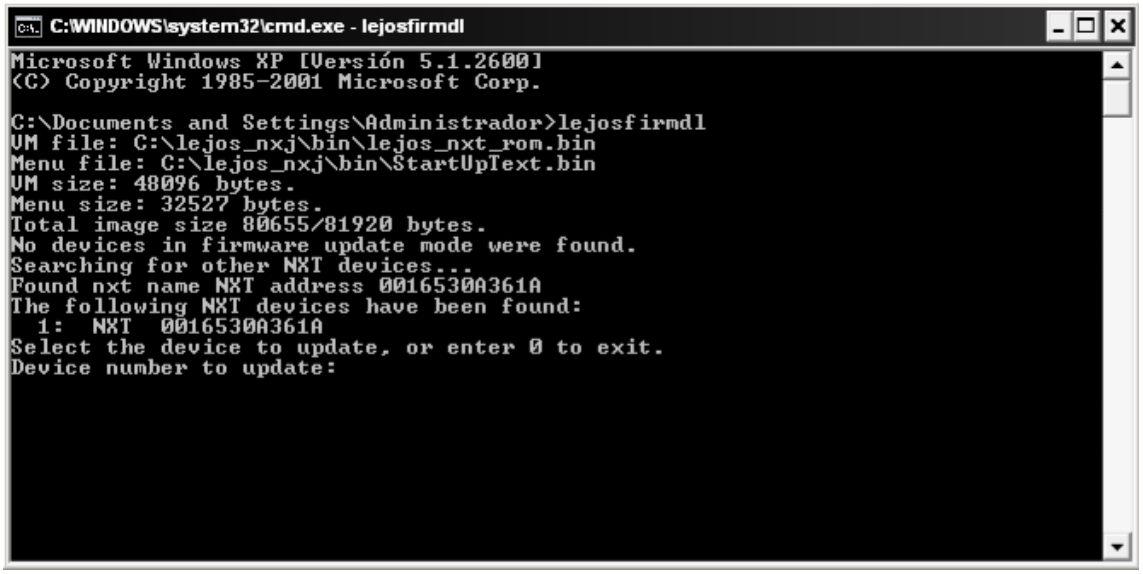
5.2.2. Instalando libusb.

Figura 79. Dar click en Next, y en la siguiente ventana deseleccionar el cuadro de texto.

Figura 80. Desactivar el test de prueba

Ahora se debe conectar al puerto USB del computador y encender el ladrillo, ir a la ventana de DOS, escribir lejosfirmdl. Si aparece la siguiente ventana:





Es porque no se ha conectado adecuadamente el dispositivo. Esperar y escribir nuevamente el comando. Y debe aparecer la siguiente, información:

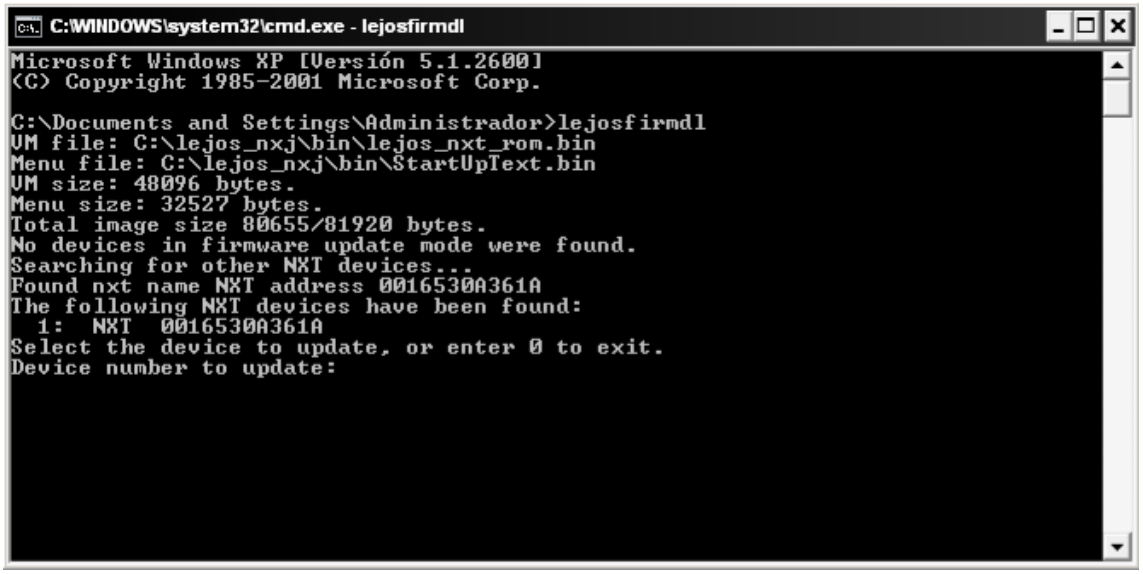


Figura 81. Etapa final de la instalación de leJOS

Escribir 1 y presionar la tecla enter. Con esto culmina la instalación de leJOS. Ahora cada vez que se encienda el ladrillo deberá aparecer leJOS en vez de LEGOS en la pantalla.

5.2.3. Instalando Y Configurando Eclipse.

Para descargar eclipse entrar en el siguiente link:

http://www.eclipse.org/downloads/

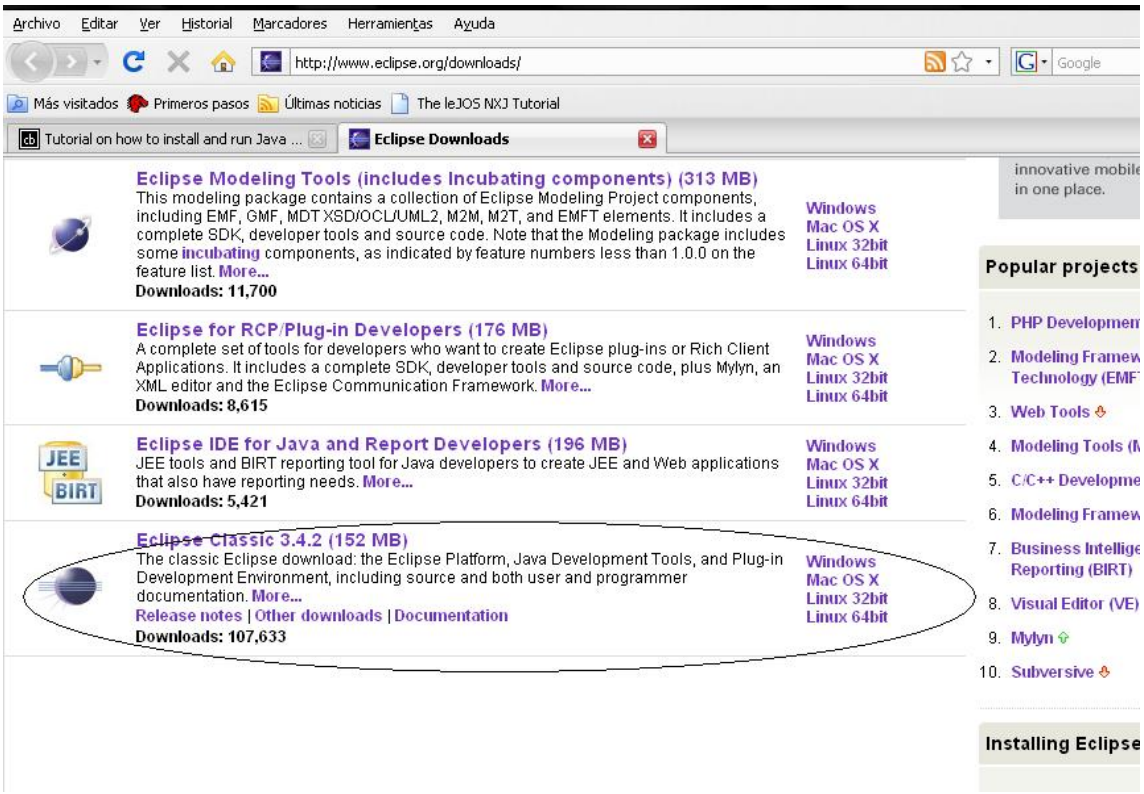


Figura 82. Pagina de descarga de eclipse

Luego de descargarlo descomprimirlo en la carpeta eclipse en el Disco Local (C:) algo parecido a la siguiente imagen:

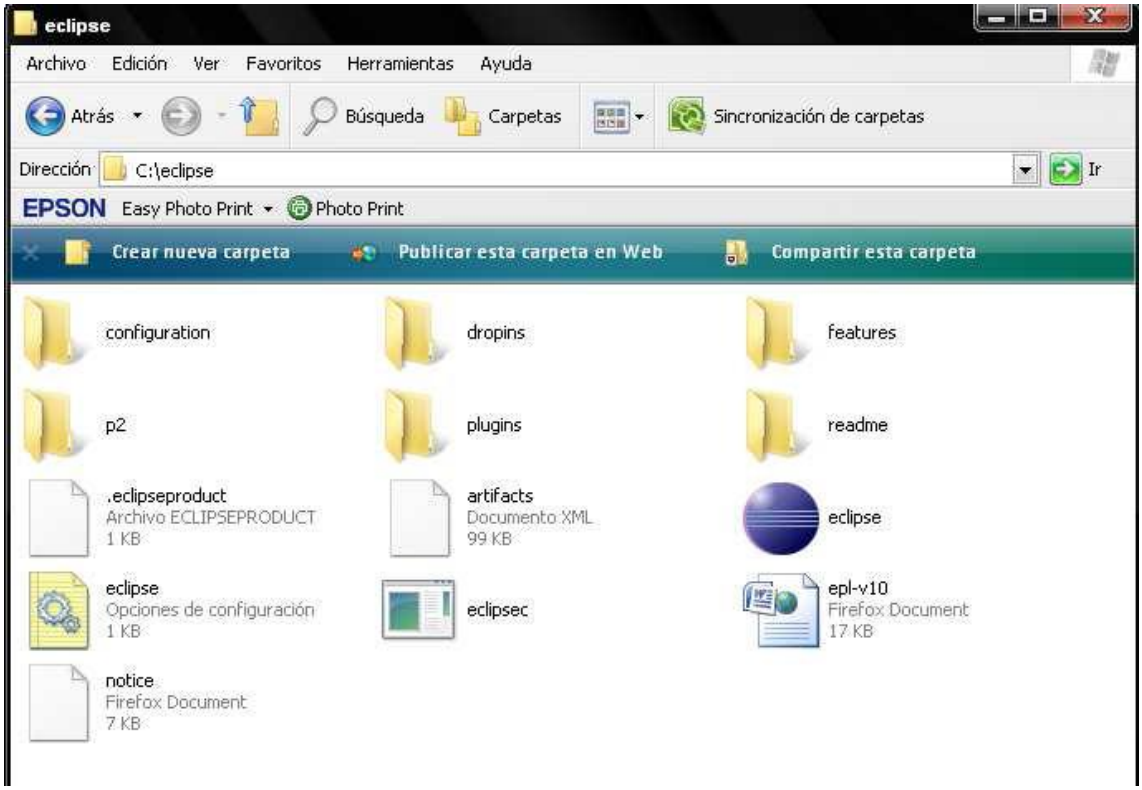
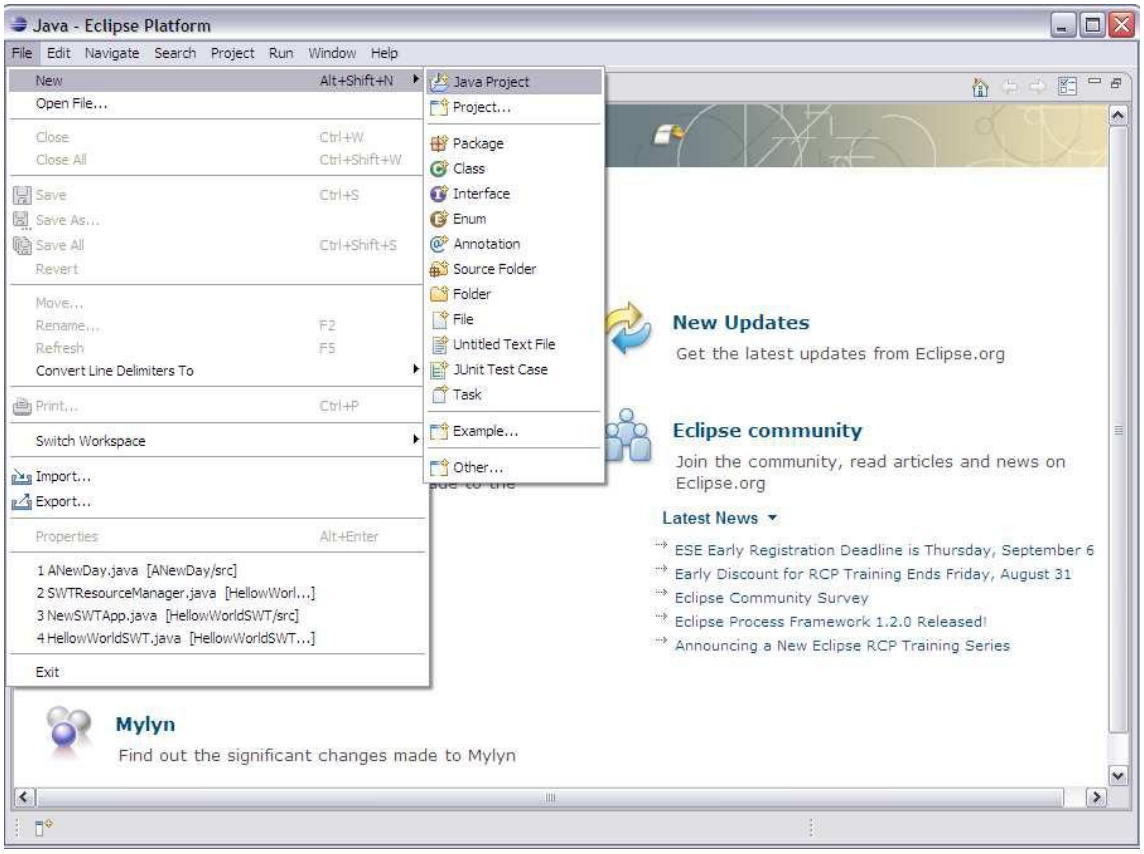


Figura 83. Carpeta eclipse

Dar doble click en el ícono de eclipse que es el archivo eclipse.exe y el se realiza la configuración de cuál va a ser la carpeta de trabajo para los proyectos de leJOS; lo mejor es crear una nueva carpeta de trabajo, que no tenga espacios en blanco en el nombre, (pues esto puede causar errores). Para comenzar a trabajar se crea la carpeta de trabajo. Colocarle el nombre leJOS_NXJ.



Luego de crear la carpeta leJOS_NXJ dar click derecho sobre ésta y luego abrir las propiedades, figura 51.

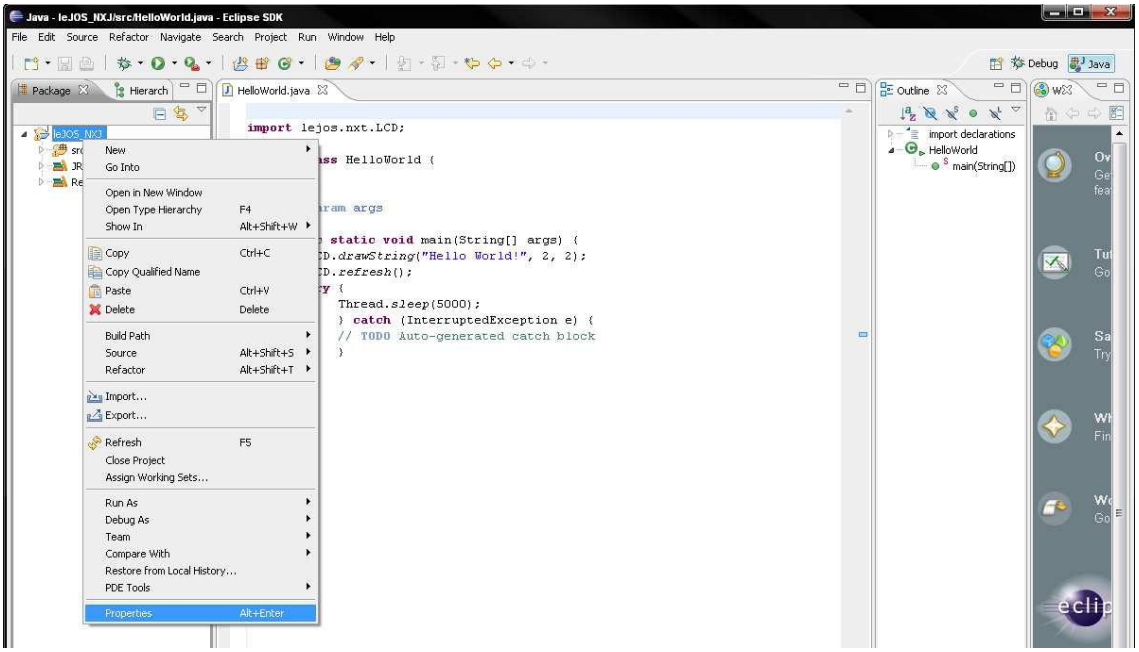


Figura 84. leJOS_NXJ – propiedades

En la ventana Properties dirigirse a Java Built PAth y luego a la pestaña Libraries. Figura 52.

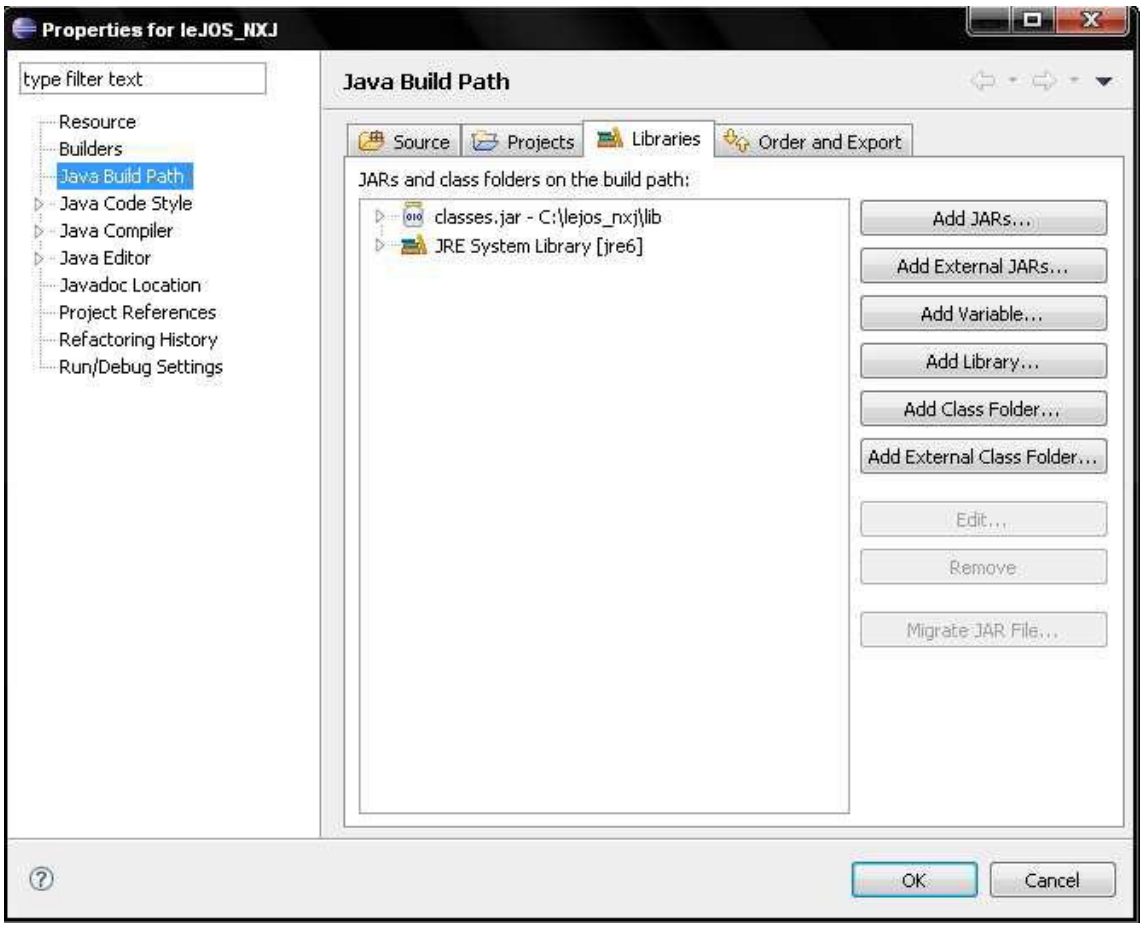


Figura 85. Java Built PAth – Libraries

Allí dar click donde dice Add External JARs, ir a la carpeta lejos_nxj y a agregar la carpeta lib; debe aparecer tal cual como en la imagen. Luego dar click en aceptar.

En la misma ventana propiedades ir al ítem Java Compiler en la parte izquierda de la ventana y dar click en enable Project specific setting ir donde dice compilar compliance level, y colocar 1.3, es recomendado por leJOS para optimizar los programas.

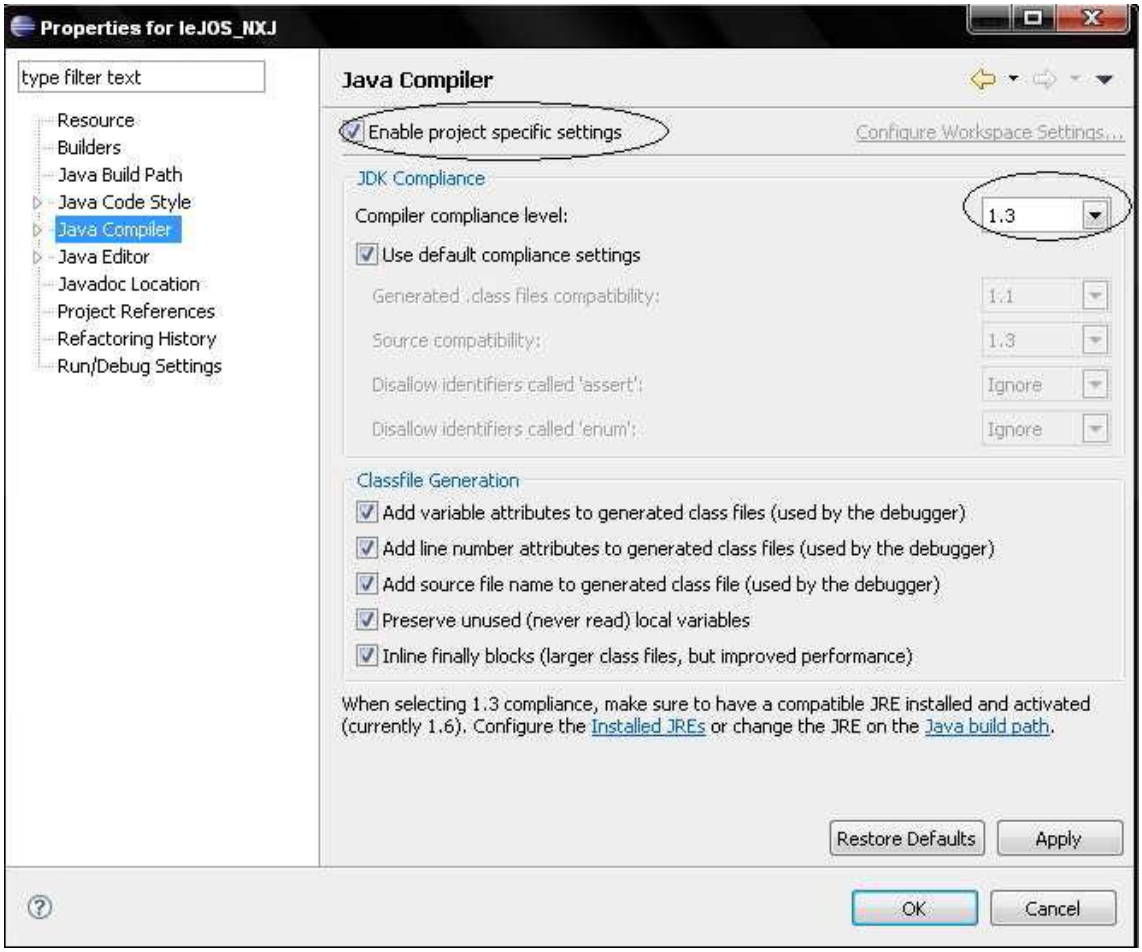


Figura 86. Java Compiler - enable Project specific setting

Dar click en OK y listo.

Ahora hay que configurar a leJOS para usarlo con eclipse, para esto dar click en la lista desplegable run, ahí dar click a External Tools y luego acceder a External Tools Configurations. Figura 54.

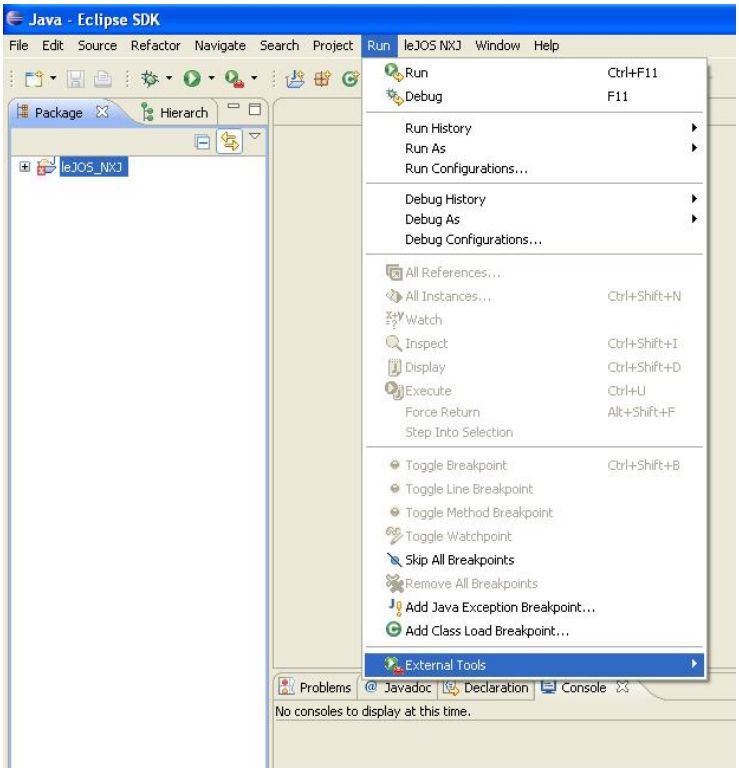


Figura 87. Run – External Tools

Aparece la siguiente ventana (figura 55). Dar clic en Program, después presionar el botón ‘New’ para crear una configuración del tipo seleccionado.

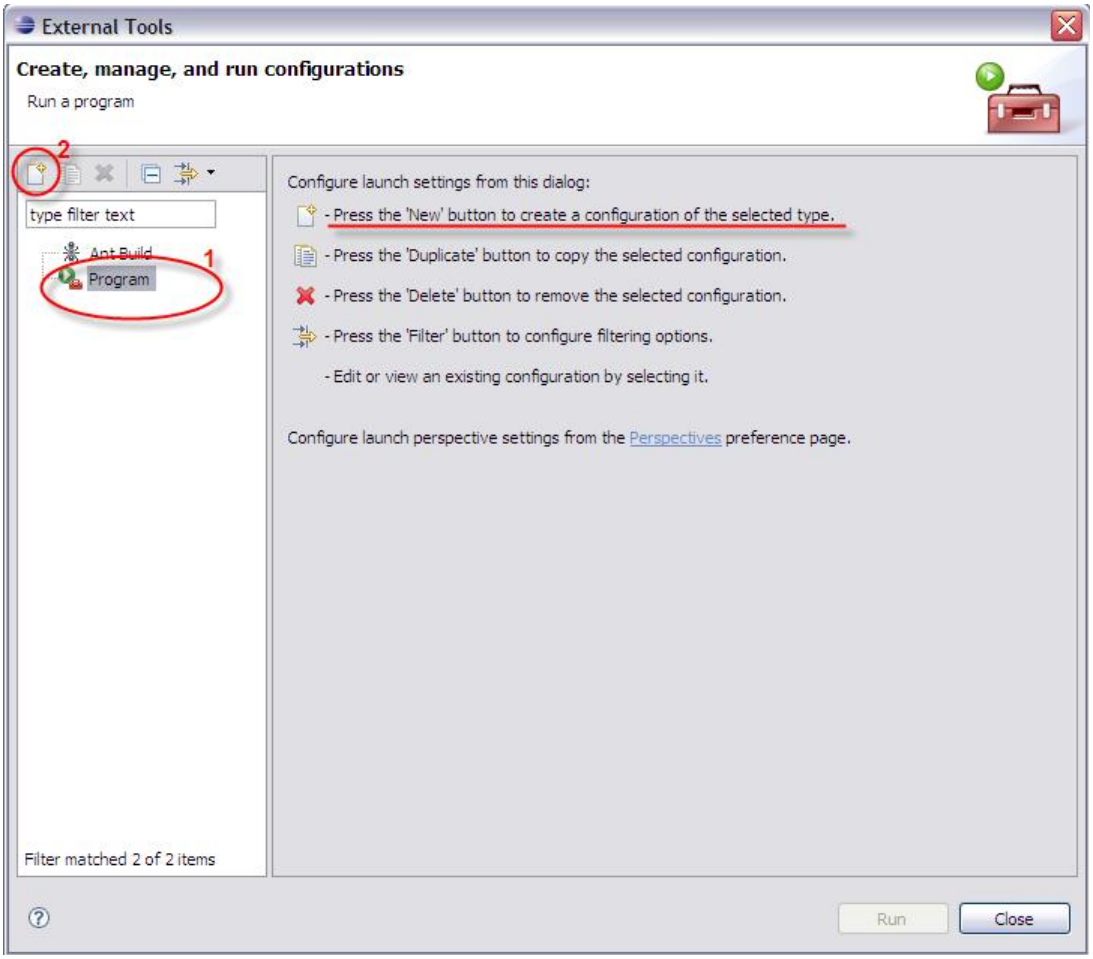


Figura 88. Crear, administrar y correr configuraciones

Darle el nombre de leJOS Download y en location dar click en lejosdl.bat buscándole en la carpeta lejos_nxj bin.

En working directorio escribir ${project_loc}\bin

En la sección arguments section escribir:

${java_type_name}

Dar click en apply y close. Ver figura 21

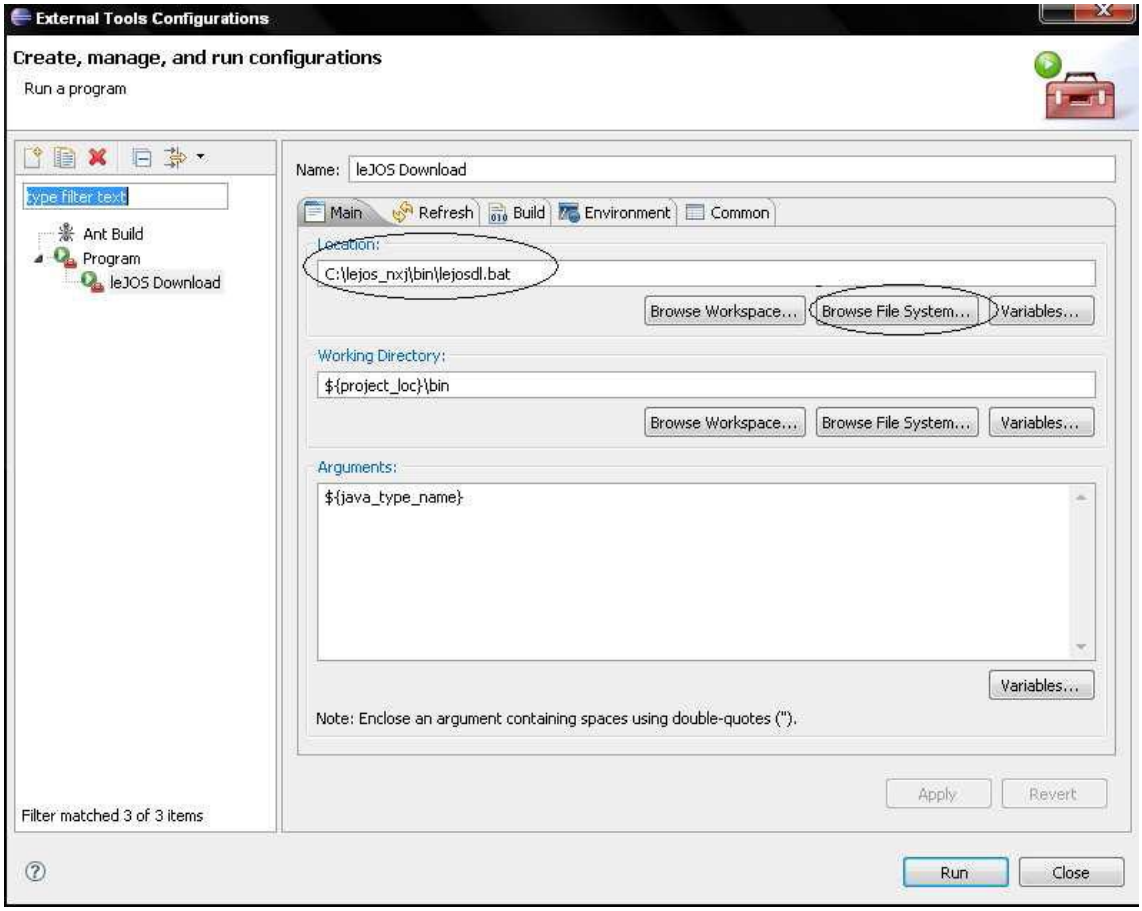


Figura 89.

Ahora se crea un acceso directo a la opción de run que se acaba de crear; para realizar esto se debe dirigir al ícono run y le dar organize favorites. Figura 57.

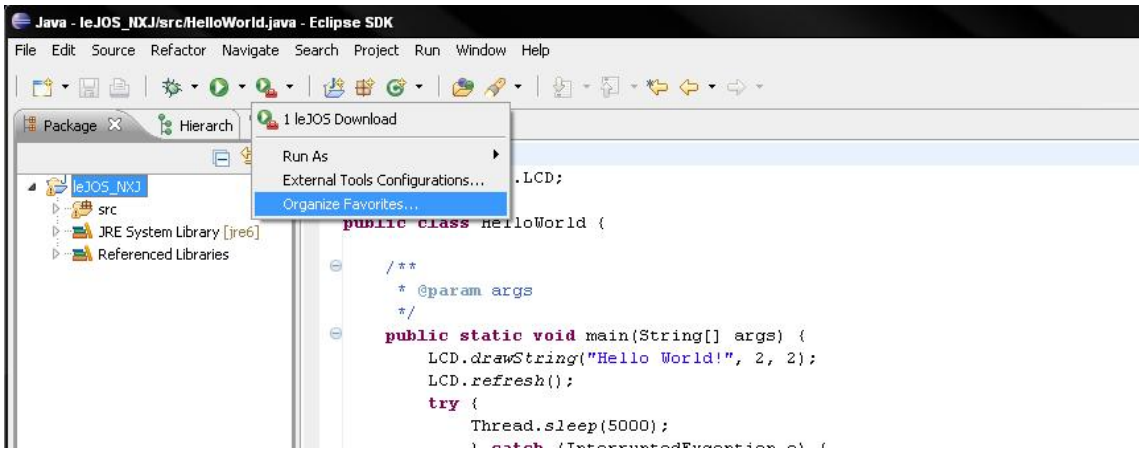
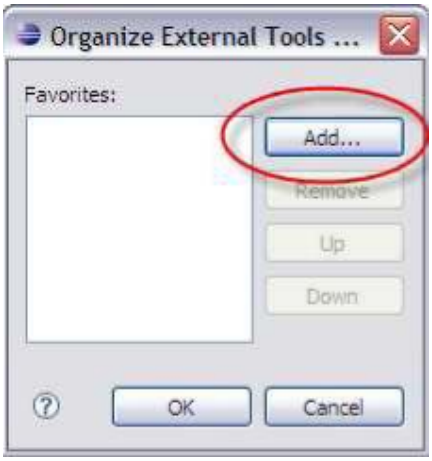
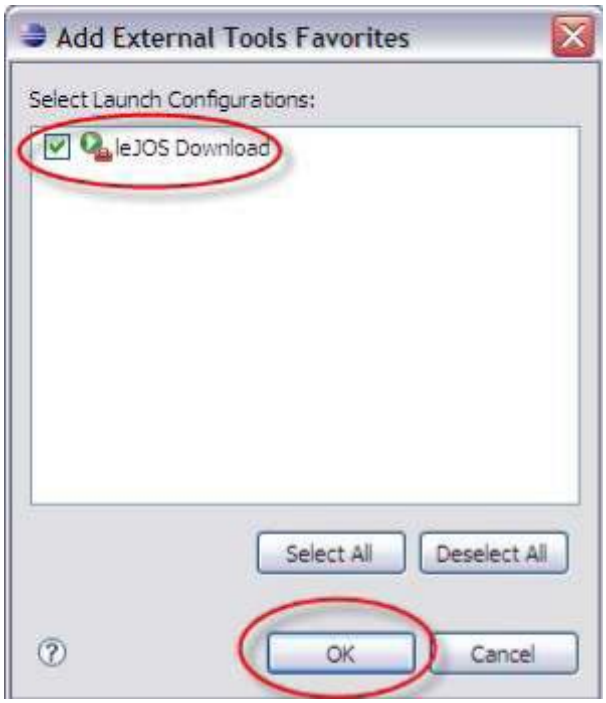


Figura 90.

Dar clic en Add.



Seleccionar leJOS Download y dar click en OK.



5.3. Cambiado el firmware del ladrillo a NXT G.

Para volver al firmware original de lejos se debe abrir el programa LEGO MINSTORMS NXT-G, luego acceder al menú desplegable Tools> Update NXT Firmware y aparecerá una ventana similar a la de la figura 58, con la última versión del firmware.

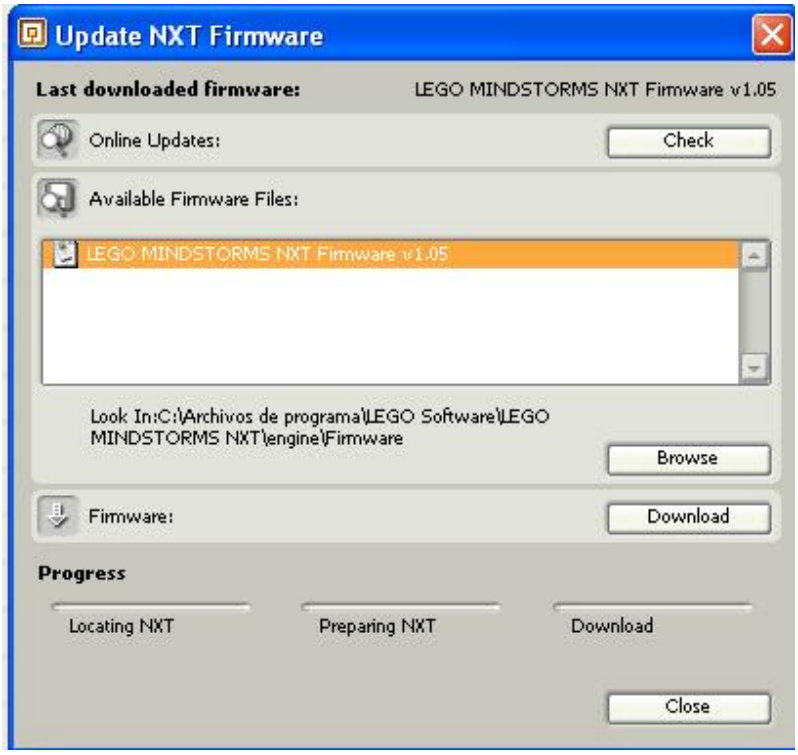


Figura 91. Update NXT Firmware

Se debe verificar que el ladrillo esté conectado y encendido, luego de haber verificado, se puede proceder a descargar el Firmware dando click en el botón Download.

5.4. Primer Programa Para El Ladrillo.

Ahora para probar que todo esté bien se escribe el típico hola mundo.

Para esto vamos a crear primero una nueva clase y hacemos click derecho sobre el proyecto, se escoge la opción New > Class >.

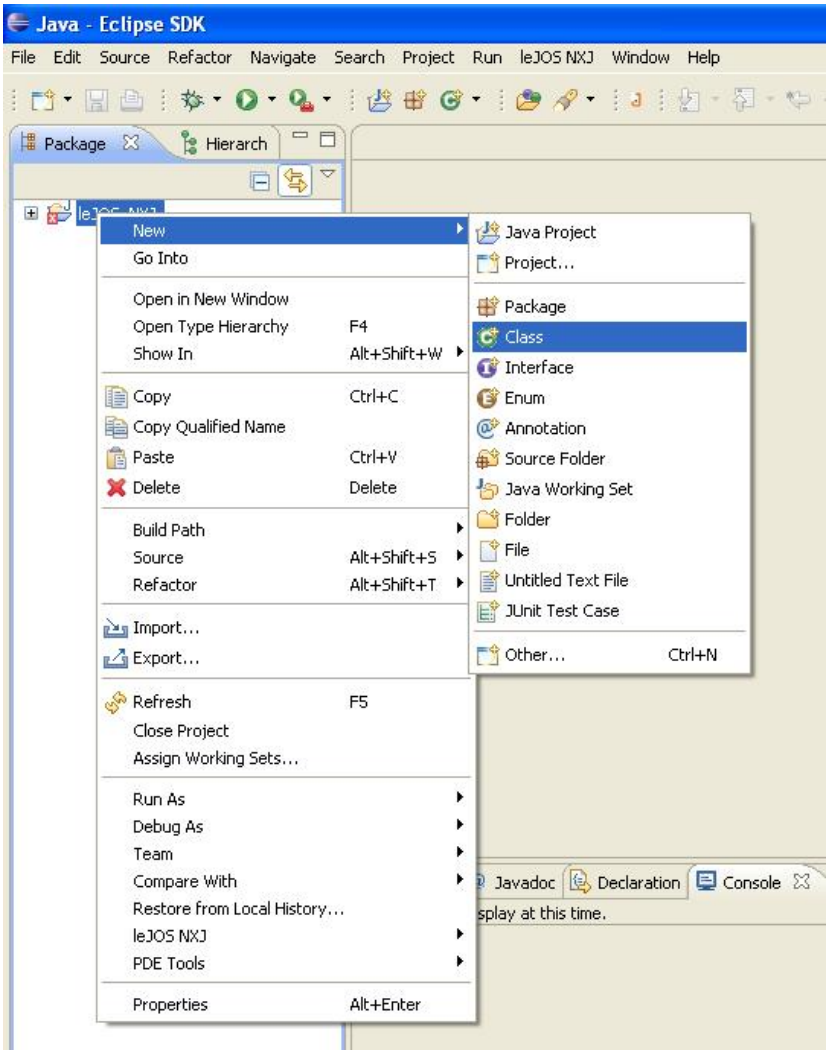


Figura 92. Crear clase para el programa

Darle el nombre de HolaMundo. Figura 60

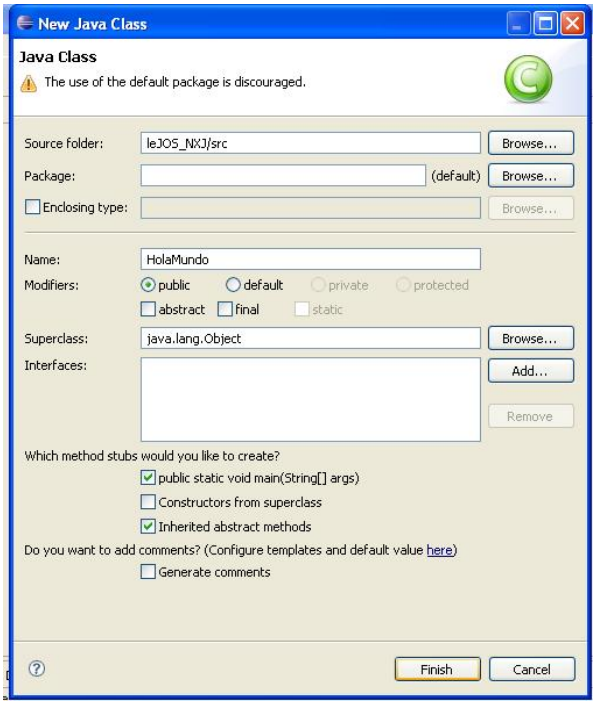


Figura 93. Registrar la clase

En la primera línea se debe colocar la importación de la clase LCD, así:

```
import lejos.nxt.LCD
```

Dentro del método main colocar las siguientes líneas:

```
LCD.drawString(“Hola Mundo”, 3, 4);
LCD.refresh();
try{
Thread.sleep(5000);
} catch(InterruptedException e){
        }
```

Lo que al final resulta es:

```
import lejos.nxt.LCD;

public class HolaMundo{

public static void main(String[] args){

LCD.drawString(“Hola Mundo”, 3, 4);

LCD.refresh();

try{

Thread.sleep(5000);

} catch(InterruptedException e){

            }

      }

}
```

Al tener todo el programa escrito, ya es posible descargarlo al ladrillo dando click a botón Run leJOS Download.

Laboratorio 2

Manejo de botones y pantalla LCD.

1. Objetivos.

1.1. General.

- Comprender el funcionamiento de la pantalla LCD y los botones del ladrillo NXT de lego.

1.2. Específicos.

- Utilizar los principales comandos para enviar datos a la pantalla LCD.
- Aprender a utilizar las principales clases, para uso de los botones del ladrillo NXT.
- Conocer las especificaciones técnicas de la pantalla LCD.
- Realizar los ejercicios al final de la guía, para afianzar los conocimientos.
- Diseñar una aplicación sencilla donde se aplique el uso de botones y la pantalla LCD del ladrillo NXT.

2. Metodología.

En esta práctica el grupo deberá estar familiarizado con el entorno de trabajo de eclipse, tener los conocimientos fundamentales de java y saber descargar un programa al ladrillo NXT.

Antes de realizar la práctica los estudiantes deben haber leído todos los métodos y clases necesarias que se usarán en ésta guía; durante el desarrollo de la guía se encontrarán aportes teóricos, ejercicios y pequeños ejemplos de programas.

La práctica consta de cinco programas que deberán ser resueltos y presentados la siguiente clase. Para la presentación del laboratorio se debe comentar detalladamente cada programa desarrollado.

Se recomienda a los grupos descargar al ladrillo cada ejemplo y programa mostrado en la guía, con el fin de afianzar los conocimientos.

3. Materiales.

Los materiales necesarios para esta práctica son:

- 6 baterías AA o batería de lego NXT (recargable preferiblemente).
- Ladrillo lego NXT.
- Cable USB.
- Computador con eclipse configurado, instalado leJOS, y todos los drivers necesarios.

4. Manejo del LCD.

La interfaz de comunicación con el usuario de lego es una pantalla LCD con un área de visión 26 x 40,6 mm, con una resolución de 100 x 64 píxeles. Esta pantalla permite mostrar 16 caracteres por línea (0 -15) y un número máximo de 8 líneas (0-7). La posición de los caracteres se realiza mediante coordenadas (x, y), como podemos apreciarlo en la siguiente figura:

Figura 94. Sistemas de coordenadas del LCD

El display se usa para interactuar con el menú del software, instalado en el ladrillo, y para mostrar información que el usuario quiera visualizar en pantalla en el momento de ejecución del programa.

4.1. Métodos para manejo de LCD usando la clase LCD de leJOS:

El manejo de la pantalla puede hacerse a través de la clase estática LCD (esto quiere decir que todos sus métodos son declarados estáticos). Esta clase contiene todos los métodos que se requieren para el envió de datos a la LCD, la ruta de la clase es:

C:\Documents and Settings\Administrador\leJOSNXJProjects\classes\lejos\nxt

Los métodos más utilizados de esta clase de lejos son:

- void drawString(String str, int x, int y): Este método se utiliza para mostrar una cadena de texto, en la pantalla y a su vez ubica el texto, en las coordenadas(x, y) establecidas.

  Ejemplo:

  Se requiere sacar la palabra resultado, en la posición (0,0) de la pantalla. El método quedaría de la siguiente forma:

  LCD.drawString ("resultado", 0, 0);

- void drawString(String str, int x, int y, boolean invert): Este método permite la salida, de una cadena de caracteres en la pantalla. A diferencia con el anterior método, este ubica el texto en una posición aleatoria y no puede ser definida por el programador, además el texto sale invertido, es decir, si boolean es false el dato sale normal, si es true el texto sale con un fondo oscuro. En resumen sus parámetros son:

  str – String(texto) para la pantalla

x – posición en X

y – posición en Y

invert – para el valor de true el texto aparece invertido

Ejemplo:

```
LCD.drawString ("resultado”, 0, 0, false);
```

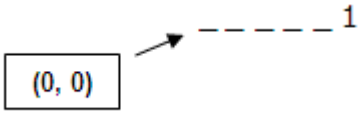
Este mostrara la palabra “resultado”, si se coloca en donde esta false, true sale invertido.

- void drawInt (int i, int x, int y): Este método es usado cuando se requiere sacar una variable tipo entero en la pantalla LCD, luego de ser llamado muestra un entero en la pantalla, en las coordenadas (x, y) establecidas.
- void drawInt (int i,int places, int x, int y): Este método saca un valor entero en las coordenadas (x ,y) establecidas .Su diferencia está en el argumento *int places*, con el cual se determina el número de caracteres reservados a partir de la coordenada x.

Ejemplo:

```
import lejos.nxt.*;
public class LCDPrueba0 {
        public static void main(String[] args) throws Exception {
           LCD.drawInt (1, 6, 0, 0);
           Thread.sleep (10000);
         }
}
```

Este ejemplo muestra en pantalla el entero 1 en la posición 6 a partir de la coordenadas x, y (0,0). Es decir:



- void drawChar(char c, int x, int y, boolean invert): Con este método se puede dibujar un carácter tipo char en una posición cualquiera de la pantalla. Sus parámetros son los siguientes:

  c – es el carácter para la pantalla

  x – posición en X

  y – posición en Y

  invert - Si boolean es false el dato sale normal, si es true el texto sale con un fondo oscuro.

  Ejemplo:

  char c; //se declara c

  LCD.drawChar('c', 0, 0, false);

Los dos métodos siguientes se utilizan para actualizar y limpiar la pantalla LCD, esto se hace con el fin de que el valor a enviar se mantenga actualizado, pues, si solo se envía el dato una vez y no se borra ni se actualiza, en la pantalla éste se mantendrá constante y no variará. El intervalo para esperar la actualización y borrado de la pantalla depende del programador, suelen utilizarse tiempos de 200 ms o 500 ms, entre más rápido se actualice, el dato será más cercano a tiempo real, pero el LCD parpadeará más al actualizar la información, así que un buen valor intermedio es 500ms.

- void clearDisplay (): Método es utilizado para limpiar la pantalla.

- void refresh (): Este método actualiza la pantalla LCD.

4.2. Ejemplos de manejo del LCD.

Ejemplo 1:

```
import lejos.nxt.*;

public class LCDPrueba {

        public static void main(String[] args) throws Exception {

           LCD.drawString ("Ejemplo:”, 0, 0);

           LCD.drawInt (1, 0, 1);

           Thread.sleep (10000);

         }

}
```

Comentando el ejemplo 1:

En el ejemplo uno se tiene una clase llamada LCDPrueba.

Se envía un dato tipo string “Ejemplo” a la posición (0,0).

Luego se envía un dato tipo entero “1” a la posición (0,1).

Finalmente se llama a un hilo de java que contiene un sleep, esto se hace para poner a dormir el programa durante un tiempo especificado en ms, (esto es para ver el dato durante un tiempo, antes de que se borre), en caso de no colocar un sleep, el pantallazo de visualización en el LCD será tan rápido que el ojo humano no detectará lo mostrado en pantalla.

Ejemplo 2:

```
import lejos.nxt.*;
```

```
public class metodosLCD {
        public static void main(String[] args)throws Exception  {
        LCD.clearDisplay ();
        LCD.drawString("mostrar int:", 0, 0);
        LCD.drawInt (1, 13, 0);
        Thread.sleep(2000);
        LCD.clearDisplay();
        LCD.drawString("mostrar char:", 0, 0);
        Thread.sleep(2000);
        LCD.clearDisplay();
        LCD.drawChar('A', 13, 2,true);
        Thread.sleep(2000);
        LCD.clearDisplay();
        LCD.drawString("mostrar entero", 0, 0);
        LCD.drawString("con arg place", 0, 1);
        LCD.drawInt(1, 2, 13, 1);
        LCD.refresh();
        Thread.sleep (10000);

    }
```

}

Comentando el ejemplo 2:

Prueba de los diferentes métodos descrito hasta el momento.

5. Manejo de botones.

El ladrillo consta con 4 Botones (izquierda, derecha, escape y enter), que pueden ser usados como contadores, o como modificadores de flujo en los programas.

5.1. Clase Button.

Para el manejo de los botones del ladrillo se utiliza la clase BUTTON. La ruta de la clase BUTTON es:

C:\Documents and Settings\Administrador\leJOSNXJProjects\classes\lejos\nxt

Por medio de esta clase se puede saber cuándo se presiona un botón específico, cuando se presiona cualquier botón, cual es valor de un contador que puede ser incrementado por el botón izquierdo, o cual es el valor del contador incrementado por el botón derecho. Para ello esta clase utiliza cuatro campos para cada uno de los botones, estos son:

ENTER

ESCAPE

LEFT

RIGHT

Los métodos más utilizados de esta clase de leJOS son:

- `boolean isPressed ()`: Este método se utiliza para comprobar si un botón está presionado. Retorna un `true` si el botón está presionado y `false` en caso contrario.

  Para saber cuándo se presione alguno de los botones se utiliza el método

isPressed() de la siguiente forma:

Button.ENTER.isPressed ();

Button.RIGHT.isPressed ();

Button.LEFT.isPressed ();

Button.ESCAPE.isPressed ();

- static int waitForPress(): Este método detiene el flujo del programa a la espera de que algún botón especificado sea presionado.

  Ejemplo:

  Button.ENTER.waitForPress ();

- void waitForPressAndRelease (): El método siguiente se utiliza para detener el flujo del programa hasta que un determinado botón sea pulsado y soltado.

5.2. Clase ButtonCounter.

Esta clase se usa para utilizar los botones del ladrillo lego NXT como contadores. Si se presiona el botón izquierdo o derecho inicia un conteo, si se presiona el botón ENTER junto con algunos de los botones izquierdo o derecho, el contador presenta un decremento. El botón escape finaliza el conteo. Ver ejemplo 2, de la sección 4.3.

La ruta de esta clase es:

C:\Documents and Settings\Administrador\leJOSNXJProjects\classes\lejos\util

Los métodos más utilizados de esta clase de lejos son:

- int getRightCount (): este método se utiliza para regresar un valor entero cuando el conteo del botón derecho es completado.

- int getLeftCount (): este método se utiliza para regresar un valor entero cuando el conteo del botón izquierdo es completado.

Para usar los métodos getRightCount y getLeftCount es necesario crear la referencia al objeto, por medio del constructor.

Ejemplo:

```
ButtonCounter bc = new ButtonCounter();
```

5.3. Ejemplos de manejo de botones.

Ejemplo 1:

```
import lejos.nxt.*;

public class EjemploButton{

public static void main (String[ ] args) throws Exception {

        LCD.clear ();

        LCD.drawString ("Pulse ENTER", 0, 0);

        Button.ENTER.waitForPressAndRelease ();

        LCD.clear ();

        LCD.drawString ("Pulse DERECHA", 0, 0);

        Button.RIGHT.waitForPressAndRelease ();

        LCD.clear ();

        LCD.drawString ("Pulse IZQUIERDA", 0, 0);

        Button.LEFT.waitForPressAndRelease ();

        LCD.clear ();
```

```
            LCD.drawString ("Pulse SALIR", 0, 0);
            Button.ESCAPE.waitForPressAndRelease ();
            LCD.clear ();
            LCD.drawString ("Adios", 6, 4);
            Thread.sleep (2000);
                    }
            }
```

Ejemplo 2:

```
import lejos.nxt.*;
import lejos.util.*;
public class countBotton {
        int cound;
        public static void main (String[] args) throws Exception{
                ButtonCounter bc = new ButtonCounter ();
                //cound=bc.getRightCount();
        bc.count("signo y regula");
        }


}
```

6. Practica.

Programa 1

Escribir un programa que realice lo siguiente:

Crear una variable tipo string con el nombre de num1

Crear una variable tipo string con el nombre de num2

Crear una variable tipo string con el nombre de resultado:

Crear una variable tipo int con el valor de 1

Crear una variable tipo int con el valor de 5.

Crear una variable tipo int con el nombre de x.

El programa debe realizar la operación sencilla de x=1+5

Se mostrará en pantalla lo siguiente

En la posición (0,0) se debe mostrar un mensaje “y=num1+num2”

En la posición (0,1) la variable x (debe mostrar el resultado de la operación)

En la posición (0,2) la variable num1 (esta debe mostrar el valor de 1)

En la posición (0,4) la variable num2 (esta debe mostrar el valor de 5)

Nota: una manera de mostrar los valores en pantalla LCD es por medio de los cast de java.

Programa 2

Modificar el programa anterior de tal forma, que antes de que comience a enviar datos a la pantalla, salga en pantalla un mensaje diciendo:

Secuencialmente:

21. “Programa de suma de dos números”

22. Al presionar cualquier botón comience y muestre lo mismo del programa 1.

23. Al presionar cualquier botón se borre la pantalla y se refresque.

Programa 3

Realizar una modificación al programa del ejemplo 1, de tal forma que las variables sean introducidas por medio del uso de los botones izquierda para la variable “a” y derecho para la variable “b”, de tal forma que se seleccione el valor deseado. Al terminar la introducción de las variables.

La suma será “y=a+b”

En la pantalla LCD debe aparecer el valor de “a” en la coordenada (0,0), el valor de “b” en la coordenada (0,1), y el valor de “y” en la coordenada (0,3).

Programa 4

Hacer una modificación al programa anterior de tal forma, que luego de que acabe el proceso del ingreso de las variables con los contadores, y de mostrar el resultado en la LCD, se llame a un método que refresque la pantalla cada 500 ms.

Por medio de algunos de los métodos de botones, colocar la opción de que cuando la persona presione el botón enter pueda nuevamente ingresar un nuevo dato a las variables, por medio del conteo, y que el proceso se pueda realizar cada vez que se desee.

Programa 5

Crear un programa que cumpla los siguientes requerimientos.

- Muestre el conteo del botón derecho en la posición de la LCD (0,0)
- Muestre el conteo del botón izquierdo en la posición de la LCD (0,5)

Laboratorio 3

Uso de sensores y accionamientos finales.

1. Objetivos.

1.1. Generales

Comprender el funcionamiento de los principales sensores compatibles con LEGO Mindstorms y el funcionamiento básico del servo motor LEGO NXT.

1.2. Específicos.

- Comprender el proceso lectura de datos de los sensores: color, tacto, ultrasónico, luz, sonido, giroscopio, acelerómetro, brújula.
- Aprender a utilizar las clases pertinentes en el entorno eclipse para el uso de cada uno de los sensores.
- Aprender a usar las clases de leJOS para el control y manejo de motores.
- Diseñar unas aplicaciones para cada uno de los sensores, y clases[21] de los motores mencionado.

2. Metodología.

En esta práctica el grupo deberá estar familiarizado con el entorno de trabajo de eclipse, tener los conocimientos fundamentales de java y saber descargar un programa al ladrillo NXT.

[21] Teniendo en cuenta el concepto de clases de la P.O.O, y la A.P.I referente a leJOS.

Leída y comprendida la teoría de sensores, accionamientos finales en la robótica móvil, y de estar familiarizados con los métodos y clases descritos en ésta guía, los estudiantes deben realizar la práctica.

Durante el desarrollo de la guía se encontrarán aportes teóricos, ejercicios y ejemplos de programas, sobre cada uno de los sensores y motores usados.

La práctica consta de once programas que deberán ser resueltos y presentados la siguiente clase.

En el informe entregado se debe documentar detalladamente cada programa desarrollado.

A los grupos se les recomienda, que los programas mostrados en la guía sean descargados al ladrillo con el fin de observar cada ejemplo y afianzar los conocimientos.

3. Materiales.

- Ladrillo LEGO NXT.
- Servo motor lego NXT.
- 6 baterías AA o batería recargable.
- Sensores: De color, ultrasónico, giroscopio, detector de luz, detector de sonido, acelerómetro, compás (brújula).
- Cable USB o dispositivo bluetooth.
- PC con leJOS NXJ, eclipse instalado y configurado[22].

[22] El computador donde se trabajará tiene que tener eclipse configurado para descargar programas al ladrillo a través de leJOS NXJ.

4. Uso de los sensores[23].

Los sensores son dispositivos usados para medir de forma objetiva variables físicas (como velocidad, posición, aceleración, etc.). Son ampliamente usados en robótica, tanto móvil como industrial ya que son los que permiten que el robot tenga conocimiento de sí mismo y del entorno, para poder realizar tareas con adecuada precisión, velocidad, exactitud y eficiencia. Para conocer cómo se encuentra su propio estado el robot debe contar con sensores internos, como encoders, resolvers, giroscopios, acelerómetros, etc. Mientras que para estar al tanto de su entorno debe contar con sensores enfocados al exterior como sensores de luz, color, temperatura, sonido, infrarrojos, de contacto, de aceleración, y ultrasónicos.

Ejemplos de sensores internos de posición son los encoders y resolvers3, para el control angular y los denominados sensores lineales de posición, como los giroscopios y compases magnéticos. Otros sensores importantes son los de posición son los de velocidad, los cuales le permiten al robot saber cuál es su velocidad y regularla a través de un bucle cerrado de control. Para realizar control sobre su estado, el robot también cuenta con sensores de presencia, que le permiten tener un rango de acción determinado, sin obstruir su propio movimiento (Palacios, E., Remiro, F. y López, L., 2004).

En cuanto al entorno el robot puede utilizar los siguientes tipos de sensores, explicados brevemente:

Sensor pasivo de luz: Se utiliza para detectar ausencia o presencia de luz, teniendo en cuenta la intensidad de ésta.

Sensor infrarrojo: Es un tipo de sensor de luz, que utiliza el espectro infrarrojo para la detección de obstáculo. Puede ser pasivo (contiene un fototransistor) o

[23] Barrientos, 1997.

activo (consta de un emisor y receptor). Este sensor realiza las tareas de medir distancia, seguir y evadir obstáculos.

Final de carrera mecánico o sensor de contacto: Este sensor funciona como un interruptor mecánico con los estados de abierto y cerrado, puede ser utilizado para la detección de presencia y proximidad, siendo los más comunes los finales de carrera.

Sensor ultrasónico: Este dispositivo es útil para medir distancias y la detección de obstáculos, funciona enviando a través del emisor una breve ráfaga de sonido ultrasónico, la cual, tras reflejarse en los obstáculos es captada por el receptor. Para determinar la distancia el sensor mide el tiempo transcurrido entre el envío y la recepción de la onda ultrasónica.

4.1. Sensores lego NXT.

leJOS cuenta con clases para el manejo y uso de sensores compatibles con LEGO Mindstorms. Dependiendo del tipo del sensor, se utilizará una interfaz adecuada; habiendo sensores con convertidor análogo digital, de contacto, o sensores que utilizan el protocolo I2C.

El set predeterminado de LEGO Mindstorms consta de cuatros sensores que son: luz, ultrasónico, sonido y contacto.

4.1.1. Interfaces implementadas para el uso de sensores.

Las interfaces son descripciones de comportamientos; como también son colecciones de métodos, los cuales a su vez pueden incluir declaraciones de constantes. A continuación se describen las interfaces para el manejo y uso de los sensores.

4.1.1.1. Interfaz ADSensorPort.

Esta interfaz es implementada cuando se utiliza un sensor Análogo/Digital, como por ejemplo el sensor de luz, sonido y giroscopio.

En esta encontramos los siguientes métodos:

public boolean readBooleanValue();

public int readRawValue();

public int readValue();

Con los que se puede obtener valores de lectura de acuerdo a la necesidad, valor booleano, o valor directamente del sensor respectivamente.

4.1.1.2. Interfaz SensorConstants.

Esta interfaz contiene las constantes empleadas por los diferentes sensores usados en el ladrillo NXT, así como su posición de memoria y sirve para establecer el modo de acuerdo al tipo de sensor (contacto, luz, sonido, etc.) y su modo de trabajo.

Por ejemplo en el caso del sensor de luz, y sus modos de operación con luz activa o inactiva, en la interfaz se puede apreciar lo siguiente.

```
public interface SensorConstants {

    public static final int TYPE_NO_SENSOR = 0x00;
    public static final int TYPE_SWITCH = 0x01;
    public static final int TYPE_TEMPERATURE = 0x02;
    public static final int TYPE_REFLECTION = 0x03;
    public static final int TYPE_ANGLE = 0x04;
    public static final int TYPE_LIGHT_ACTIVE = 0x05;
    public static final int TYPE_LIGHT_INACTIVE = 0x06;
    public static final int TYPE_SOUND_DB = 0x07;
    public static final int TYPE_SOUND_DBA = 0x08;
    public static final int TYPE_CUSTOM = 0x09;
    public static final int TYPE_LOWSPEED = 0x0A;
    public static final int TYPE_LOWSPEED_9V = 0x0B;

    public static final int MODE_RAW = 0x00;
    public static final int MODE_BOOLEAN = 0x20;
    public static final int MODE_TRANSITIONCNT = 0x40;
    public static final int MODE_PERIODCOUNTER = 0x60;
    public static final int MODE_PCTFULLSCALE = 0x80;
    public static final int MODE_CELSIUS = 0xA0;
    public static final int MODE_FARENHEIT = 0xC0;
    public static final int MODE_ANGLESTEP = 0xE0;
```

Figura 95. Interfaz SensorConstants

4.1.1.3. Interfaz I2CPort.

Esta interfaz implementa a la interfaz BasicSensorPort por lo que hereda todos sus métodos, variables y propiedades, y agrega los siguientes.

public void i2cEnable(); habilita el puerto I2C

public void i2cDisable(); deshabilita el puerto I2C

public int i2cBusy(); regresa valor booleano de verdadero cuando el puerto I2C está ocupado y falso en caso contrario.

public int i2cStart(int address, int internalAddress, int numInternalBytes, byte[]buffer,int numBytes, int transferType); se inicializa el puerto I2C.

4.1.1.4. Interfaz I2CSensor.

Esta interfaz puede ser implementada por un sensor de comunicación I2C tales como: ultrasónico, compás, color, aceleración. Esta implementa la interfaz I2CPort.

4.1.2. El sensor de luz.

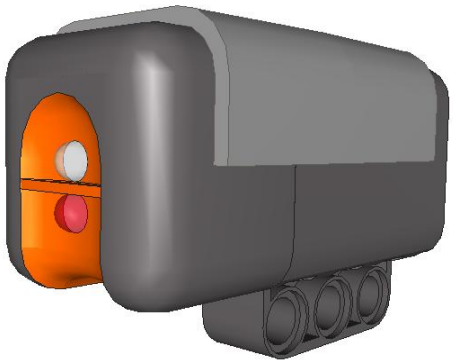


Figura 96. Sensor de luz de LEGO NXT

El sensor de luz es uno de los sensores que permiten al robot interactuar con su entorno, imitando a la visión humana, pero en mucha menor escala. El sensor de luz dota al robot de la capacidad de diferenciar entre la luz y la oscuridad siendo capaz de leer la intensidad de la luz en una habitación y medir la intensidad de la luz reflejada en una superficie. Dependiendo del nivel de reflexión es posible diferenciar ciertos rangos de color.

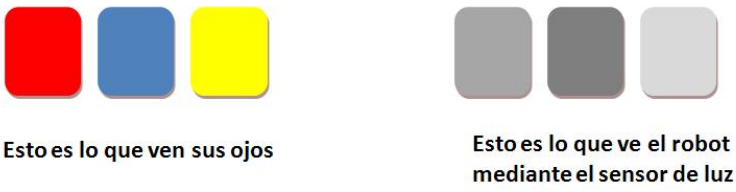


Figura 97. Comparación ojo humano – sensor de luz

La figura siguiente, muestra la sensibilidad del sensor NXT en una amplia gama de intensidades de luz, que es normalmente medido en unidades de lux (lx).

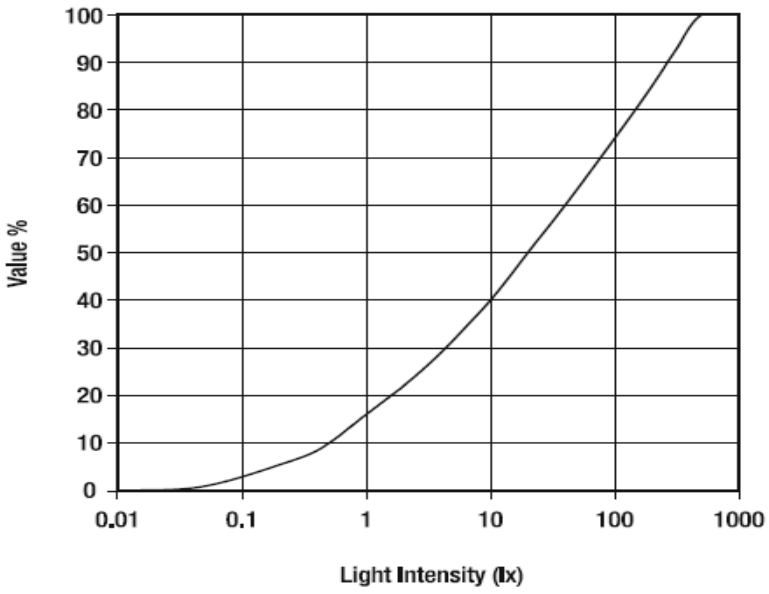


Figura 98. Sensibilidad del sensor de luz[24]

4.1.2.1. Clase LightSensor

La clase que opera el sensor de luz tiene el nombre LightSensor implementa la interfaz SensorConstants por lo que hereda todas las variables y propiedades de esta; también usa la interfaz ADSensorPort que es una extensión de BasicSensorPort y soporta sensores análogo/digital. Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\LightSensor.

Para usar el sensor de luz hay que crear un nuevo objeto, colocando en los argumentos del constructor el puerto donde se conectará el sensor, y también se indica si la luz del sensor debe estar encendida (para medir reflexión en los objetos), o apagada (para medir luz del ambiente). Los puertos pueden ser: S1, S2, S3 y S4, y la variable para indicar el encendido o apagado de la luz del sensor es una variable booleana.

---

[24] Gasperi, 2007. p. 13.

Ejemplo:

Se crea un objeto llamado “LightSensor” referenciado con el nombre “Luz”, se indica que está conectado en el puerto S1 y la luz está apagada (la variable booleana igual a false)

```
LightSensor Luz = new LightSensor(SensorPort.S1,false);
```

Nota: La variable después de la declaración del puerto, es para la luz del sensor, para un valor true la luz se enciende, para un valor false la luz se apaga, y éste solo mide la luz del ambiente.

Métodos de la clase LightSensor.

Los principales métodos utilizados por el sensor de luz son:

- .El método readValue():Este método se usa cuando se quiere obtener un valor de lectura de forma porcentual (0% a 100%), y utiliza las variables _zero y _hundred los cuales son los valores de lecturas mínimo y máximo.

  Antes de usar este método se puede calibrar el sensor para indicarle, cuales son los valores correspondientes a cada lectura aumentando así la exactitud y adaptación a las condiciones ambientales del medio.

  Para calibrar el valor mínimo de luz se llama al método calibrateLow (antes de llamarlo hay que colocar al sensor en la lectura baja):

  ```
  public void calibrateLow()
      {
          _zero = port.readRawValue();
      }
  ```

  Para el valor máximo de luz se utiliza el siguiente método:

Para calibrar el valor mínimo de luz se utiliza el método calibrateHigh (antes de llamarlo hay que colocar al sensor en la lectura alta):

```
public void calibrateHigh()
        {
                _hundred = port.readRawValue();
        }
```

- Método readNormalizedValue():Este método se usa cuando se requiere obtener un valor de lectura normalizado. Los valores correspondientes están, entre 145 (oscuridad) y 890 (luz máxima). Su valor normalizado siempre es un número entero entre 0 y 1023. Los valores más bajos corresponden a colores oscuros y los más altos a colores claros.

  Ejemplo:

```
import lejos.nxt.Button;
import lejos.nxt.LCD;
import lejos.nxt.LightSensor;
import lejos.nxt.SensorPort;
/**Programa para mostrar el funcionamiento del sensor de luz de lego
 * se imprime en pantalla el valor normalizado de luz
 * @author Charles
 */
public class PruebaLuz {
        public static void main(String[] args) throws Exception {
```

```
        LightSensor Luz = new LightSensor(SensorPort.S1,false);
        System.out.println("Presione botón"+"\n para iniciar");

        while(!Button.ESCAPE.isPressed()){

            LCD.clear();
            LCD.drawInt((int) Luz.readNormalizedValue(),0,0);
            LCD.refresh();
            Thread.sleep(500);
        }
    }
```

4.1.3. Sensor Ultrasónico

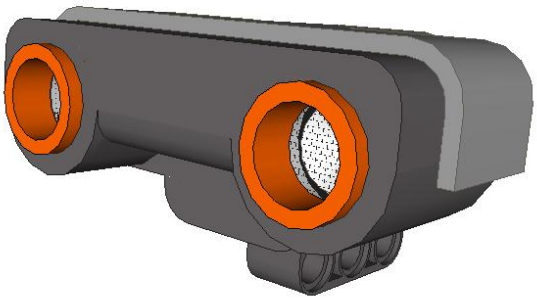


Figura 99. Sensor de ultrasonido de LEGO NXT

Este sensor le permite al robot detectar y medir obstáculos, el rango de detección entre 0 y 127 cm con una precisión de +/- 3 cm (distancias superiores a 127 cm el

sensor mostrara una salida de 255). Esto significa que si hay un obstáculo a 40 cm el sensor lo detecta desde 37 hasta 43 cm, este es su margen de error.

El sensor funciona como un sonar, emite un tren de pulsos ultrasónicos con una frecuencia de 40 KHz, luego este recibe la señal de eco y mide el tiempo entre la señal emitida y el eco recibido. Una característica importante de este sensor, es el detectar objetos grandes, como una pared plana, en este caso la medida del sensor es muy confiable. Sin embargo si el escenario se complica, con muchos objetos pequeños, su medida no es tan confiable.

4.1.3.1. Clase UltrasonicSensor.

El sensor ultrasónico de LEGO es un sensor que se comunica con el ladrillo por protocolo I2C, por lo que implementa la interfaz I2CSensor. Este sensor puede operar en dos modos, modo continuo y modo ping. En el modo continuo el sensor emite una onda de sonido de alta frecuencia y espera el eco para calcular así la distancia, en el modo ping el sensor emite una onda de sonido y recibe 8 ecos.

El modo continuo se encuentra activo por defecto, pero si se quiere cambiar a él luego de haber usado el modo ping se hace llamada al método continuous (), el cual internamente contiene:

```
public int continuous()
    {
        return setMode(MODE_CONTINUOUS);
    }
```

Modo ping.

Para activar el modo ping se llama al siguiente método, que internamente contiene:

```
public int ping()
    {
```

```
        return setMode(MODE_SINGLE);
    }
```

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\UltrasonicSensor

Métodos de la clase UltrasonicSensor.

- Método getDistance():Con este método se obtiene la medida de la distancia en cm, el método al ser llamado regresa un valor entero de 0 a 255, el rango de medida del sensor es de 0 a 127 cm, cuando el objeto se encuentra fuera del alcance del sensor el sensor entregará un valor de 255.
- Método getDistances(int dist[]):Este método regresa un arreglo de 8 valores de ecos recogidos, que puede ser almacenado en una variable, este sólo puede ser usado en el modo ping si ocurre algún error regresará un valor de -1.

  Ejemplo 1:

  Programa del funcionamiento interno del método.

```
public int getDistances(int dist[])
        {
        if (dist.length < inBuf.length || mode != MODE_SINGLE) return -1;
            wait(dataAvailableTime);
            int ret = getData(DISTANCE, inBuf, inBuf.length);
            for(int i = 0; i < inBuf.length; i++)
                dist[i] = (int)inBuf[i] & 0xff;
```

```
            return ret;
        }
```

Ejemplo 2: en la pantalla LCD se muestra la versión del sensor, ID del sensor, tipo de sensor y la distancia.

```
import lejos.nxt.*;

public class Ultrasonido {

        public static void main(String[] args) throws Exception {

          UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);

          while (!Button.ESCAPE.isPressed()) {

            LCD.clear();

            LCD.drawString(sonic.getVersion(), 0, 0);

            LCD.drawString(sonic.getProductID(), 0, 1);

            LCD.drawString(sonic.getSensorType(), 0, 2);

            LCD.drawInt(sonic.getDistance(), 0, 3);

          }

        }
```

4.1.4. Sensor de Sonido.

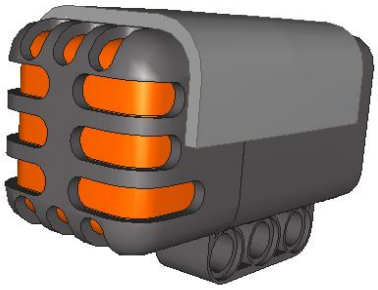


Figura 100.Sensor de sonido de LEGO NXT

Este sensor se utiliza para detectar intensidad de sonido, e incapaz detectar tonos, o modulación. Las unidades de medición de intensidad entregadas por el sensor pueden ser en (dB), y se presenta en una escala logarítmica. El sensor mide la intensidad del sonido hasta 90dB, Figura 68.

Además de la configuración para medir decibeles (dB), se puede obtener medidas en decibeles ajustados (dBA), los cuales son una escala (dB) en función del oído humano.

Debido a que los decibeles se miden en una escala logarítmica, el sensor de sonido proporciona valores de 0 a 100% para que sea más fácil su interpretación.

Ejemplo:

4-5% Una sala en silencio.

5-10% Alguien hablando lejos.

10-30% Es una conversación cerca del sensor o música en un volumen normal.

30-100% Sonidos de alto volumen.

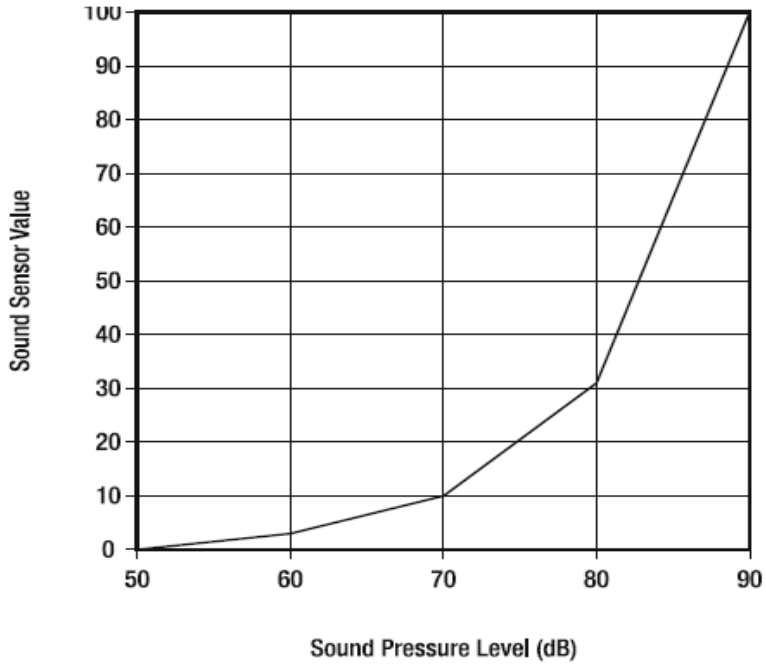


Figura 101.Valor de nivel sonoro Vs nivel de presión sonora[25]

El sensor de sonido NXT, contiene internamente un micrófono, ajustado en una pequeña esponja, un condensador y un conector, así como también un circuito impreso.

4.1.4.1. Clase SoundSensor.

Para el uso del sensor de sonido se encuentra la clase llamada SoundSensor. Esta clase implementa la interfaz SensorConstants. Se usa para operar el sensor de sonido de lego.

Para usar el sensor de sonido se debe crear un objeto

```
SoundSensor sonido= new SoundSensor(SensorPort.S1,true);
```

[25] Gasperi, 2007. p. 11.

Donde se coloca el puerto donde se conectará el sensor y se coloca el modo de operación del sensor, true regresa un valor en dBA o false regresa un valor en dB.

También se puede omitir el parámetro booleano, por lo que se coloca por defecto en modo dB.

```
SoundSensor sonido= new SoundSensor(SensorPort.S1);
```

Esta clase se encuentra en la siguiente ruta: \leJOSNXJProjects\classes\lejos\nxt\SoundSensor

Ejemplo:

```
import lejos.nxt.Button;

import lejos.nxt.LCD;

import lejos.nxt.SensorPort;

import lejos.nxt.SoundSensor;

/**un programa sencillo para comprobar el funcionamiento del sensor de sonido

 * muestra en pantalla el nivel del sonido leído

 * @author Charles

 *

 */

public class PruebaMic {

	/**

	 * @param args

	 * @throws Exception
```

```
 */
public static void main(String[] args) throws Exception {
        SoundSensor sonido= new SoundSensor(SensorPort.S1,true);
        while(!Button.ESCAPE.isPressed()){
                LCD.clear();
                LCD.drawInt((int) sonido.readValue(), 0, 0);
                LCD.refresh();
                Thread.sleep(500);
        }
}
}
```

4.1.5. Sensor de Contacto.

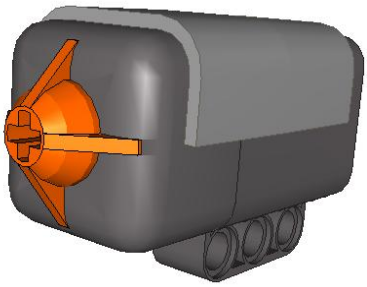


Figura 102.Sensor de contacto de LEGO NXT

Este sensor le da al robot móvil el sentido del “tacto”. Internamente en este sensor encontramos; un pulsador, un conector, una resistencia y la placa de circuito impreso.

4.1.5.1. Clase TouchSensor.

El uso del sensor de contacto se realiza por medio de la clase TOUCHSENSOR, esta clase implementa la interfaz SensorConstants. Para usarla se debe crear un objeto

```
TouchSensor touch = new TouchSensor(SensorPort.S2);
```

Para emplearla se pregunta por el método isPressed(); el cual devuelve un valor booleano, true si está presionado o false en caso contrario.

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\TouchSensor

4.1.6. Sensor de Color.

El sensor de color utiliza una luz diferente a la luz de ambiente, lo que le permite medir los diferentes valores de reflexión de cada color, en la clases de abajo, están mencionadas cuales son estos colores que detecta este sensor.

Figura 103.Sensor de color de Hitechnic para LEGO NXT

4.1.6.1. Clase ColorSensor.

Esta es la clase que contiene todos los métodos necesarios para controlar el sensor de color. Esta clase se extiende de I2CSensor, se usa para operar el sensor de color de hitechnic.

Lo primero es crear el objeto donde indicamos la referencia que usaremos para el sensor de color y el puerto donde se colocará.

```
ColorSensor cmps = new ColorSensor(SensorPort.S1);
```

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\addon\ColorSensor

Métodos de la clase ColorSensor.

Los métodos utilizados por esta clase son:

- Método getColorNumber(): Este método regresa un valor correspondiente a cada color con respecto a la siguiente lista.

0	negro
1	violeta
2	púrpura
3	azul
4	verde
5	lima
6	amarillo
7	naranja
8	rojo
9	carmesí
10	magenta
11 a 16	pasteles
17	blanco

Tabla 24. Valores correspondientes a cada color

Ejemplo:

En el siguiente ejemplo se muestra en la LCD los valores de “RGB” correspondientes, y se muestra el color propio con respecto a la tabla 1 mencionada arriba.

```
import lejos.nxt.*;
```

```
import lejos.nxt.addon.*;

/**
 * For testing the HiTechnic color sensor (see lejos.nxt.ColorSensor).
 * @author BB
 */
public class ColorDetector {
	//final static int INTERVAL = 200; // milliseconds
	public static void main(String[] args) throws InterruptedException {
		ColorSensor cmps = new ColorSensor(SensorPort.S1);
		String color = "Color";
		String r = "R";
		String g = "G";
		String b = "B";

		while(!Button.ESCAPE.isPressed()) {
			LCD.clear();
			LCD.drawString(cmps.getProductID(), 0, 0);
			LCD.drawString(cmps.getSensorType(), 0, 1);
			LCD.drawString(cmps.getVersion(), 9, 1);
			LCD.drawString(color, 0, 3);
```

```
                    LCD.drawInt((int)cmps.getColorNumber(),7,3);
                    LCD.drawString(r, 0, 5);
                    LCD.drawInt((int)cmps.getRed(),1,5);
                    LCD.drawString(g, 5, 5);
                    LCD.drawInt((int)cmps.getGreen(),6,5);
                    LCD.drawString(b, 10, 5);
                    LCD.drawInt((int)cmps.getBlue(),11,5);
                    LCD.refresh();
                    Thread.sleep(200);
                }
            }
        }
```

- Método getColorIndexNumber (): Este método funciona igual que el anterior, la diferencia es que entrega un valor con 6 Bits de resolución, para el rojo de 5-4 Bits, para el verde de 3-2 Bits y para el azul 1-0 Bits.
- Métodos getRed(), getBlue(), getGreen(): Estos métodos funcionan de la misma forma, cada uno al llamarse regresa un valor de saturación de 0 a 255, dependiendo de cual se llame. Rojo, azul, o verde.
- Métodos getNormalizedRed(), getNormalizedGreen(), getNormalizedBlue(): Estos tres métodos funcionan bajo el mismo principio, regresan un valor normalizado, dependiendo de cual se haya llamado; rojo, verde o azul.

- Métodos getRawGreen(), getRawBlue(), getRawRed(): Estos tres métodos devuelven un valor de 0 a 1023 dependiendo de cual se llame, el valor es correspondiente a cada color, verde, azul o rojo.

- Método initWhiteBalance (): Este método se usa para calibrar el nivel de blanco del sensor.

- Método initwhite(): coloca el sensor en modo de calibración, para obtener mejores resultados.

  El sensor debe ubicarse en una superficie blanca difusa, a una distancia de aproximadamente 15mm antes de llamar a este método, después de una fracción de un segundo las luces del sensor se encenderán y se realizará la calibración, cuando el sensor haya sido calibrado, este mantendrá esta información en la memoria no volátil.

  El método regresara 0 si ha sido realizada con éxito, y -1 por cualquier otra razón.

- initBlackLevel(): El método initBlackLevel hace exactamente lo mismo que iniwhite(), pero esta vez hay que colocar el sensor en una zona que no contenga obstáculos, 0 a 50 cm., para que el sensor calibre su nivel de negro.

4.1.7. Giroscopio.

El Gyro NXT contiene un sensor giroscópico de eje único, que detecta la rotación y devuelve un valor que representa el número de grados por segundo de rotación. El Gyro sensor puede medir hasta + / - 360 ° por segundo de rotación. Se conecta a un puerto del NXT mediante un cable estándar NXT y utiliza la interfaz analógica del sensor. La tasa de rotación puede leer hasta aproximadamente 300 veces por segundo.

El Gyro sensor se encuentra dentro del encapsulado estándar de Mindstorms NXT de lego para que coincida con el de otros elementos Mindstorms.

El eje de medición está en el plano vertical con la posición del sensor giroscopio que se muestra en la figura 10.

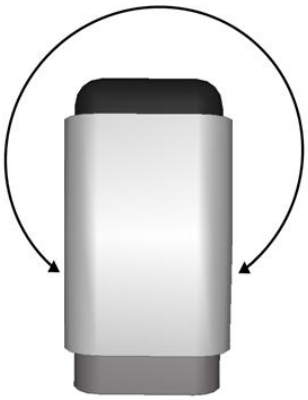


Figura 104. Sensor giroscópico de Hitechnic para LEGO NXT

4.1.7.1. Clase GyroSensor.

Esta es la clase con la que se maneja el sensor giroscópico, implementa la interfaz SensorConstants, que se encuentra en el paquete addon de lejos NXT, se usa para adquirir las lecturas del sensor giroscópico de hitechnic.

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\addon\GyroSensor

Métodos de la clase.

- Método readValue():Este método se utiliza para adquirir el valor de lectura del sensor giroscópico, el cual regresa un valor normalizado con respecto al valor del offset. El offset tiene un valor de 600 por defecto. Para modificar este valor y colocar otro se usa el método setOffset() por medio del cual se asigna un valor de offset para conseguir medidas exactas.

  Para usar el sensor giroscopio se debe crear una referencia a un objeto

  ```
  GyroSensor Gyro = new GyroSensor(SensorPort.S1);
  ```

  Ejemplo:

```
import lejos.nxt.Button;
import lejos.nxt.LCD;
import lejos.nxt.SensorPort;
import lejos.nxt.addon.GyroSensor;
/**este programa es para probar el funcionamiento del giroscopio
 *
 * @author Charles
 */
public class PruebaGyro {
	public static void main(String[] args) throws Exception {
		GyroSensor Gyro = new GyroSensor(SensorPort.S1);
			while(!Button.ESCAPE.isPressed()){
			LCD.clear();
			LCD.drawInt((int) Gyro.readValue(), 0, 0);
			LCD.refresh();
			Thread.sleep(500);
		}
	}
}
```

4.1.8. Compás.

El NXT contiene un sensor de brújula digital magnética que mide el campo magnético terrestre y calcula un ángulo de inicio. Se conecta a un puerto del NXT mediante un cable estándar NXT y utiliza el protocolo de comunicaciones de bus I2C. El ángulo de inicio se calcula con una aproximación de 1° y se actualiza 100 veces por segundo.

La brújula está ubicada en el encapsulado estándar de Mindstorms NXT de lego para que coincida con el de otros elementos Mindstorms.

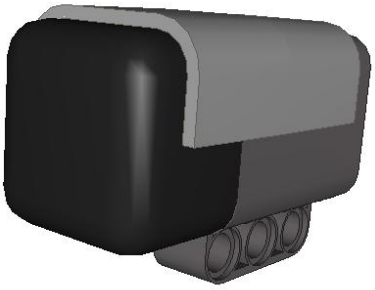


Figura 105.Compás de Hitechnic para LEGO NXT

4.1.8.1. Clase CompassSensor.

Esta clase se extiende de la clase I2CSensor por ser este un sensor que utiliza el protocolo de comunicación del bus I2C. Para trabajar con el compás hay que crear un objeto y una referencia a este.

```
CompassSensor compass = new CompassSensor(SensorPort.S1);
```

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\addon\CompassSensor

Los principales métodos a utilizar son:

Método getDegrees (): Este método regresa un valor numérico de 0 a 360 indicando como 0 el norte, el valor de lectura aumenta a medida que se gira en dirección de las manecillas del reloj. La resolución de este sensor es de 0.1.

Método resetCartesianZero (): Este método se usa para reiniciar el valor cartesiano, asignándole la posición que tiene el sensor al momento de llamar al método como norte o como valor cero.

Método getDegreesCartesian (): Este método regresa el valor cartesiano medido de 0 a 359 el valor es relativo al valor asignado como norte al momento de llamar al método resetCartesianZero ().

Ejemplo:

```
import lejos.nxt.*;

import lejos.nxt.addon.CompassSensor;

public class CompasTest {

	/**

	 * @param args

	 */

	public static void main(String[] args) throws Exception{

		CompassSensor compass = new CompassSensor(SensorPort.S1);

		while(!Button.ESCAPE.isPressed()) {

			LCD.clear();

			LCD.drawInt((int) compass.getDegrees(), 0, 0);

			LCD.refresh();
```

```
                    //Thread.sleep(500);
                }
            }
}
```

4.1.9. Acelerómetro.

El sensor de aceleración de tres ejes, es capaz de medir la aceleración en los ejes," x", "y", "z". La aceleración es medida en el rango de -2g a 2g con un periodo de 200 veces por segundo.

El sensor de aceleración también puede ser utilizado para medir la inclinación en los tres ejes. Se conecta a un puerto del NXT mediante un cable estándar NXT y utiliza el protocolo de comunicación del bus I2C. La medida de aceleración para cada eje se actualiza aproximadamente 100 veces por segundo.

El sensor de aceleración se encuentra en el l encapsulado estándar de Mindstorms NXT de LEGO para que coincida con el de otros elementos Mindstorms.

Los tres ejes de medida están etiquetados "x"," y", "z", como se muestra.

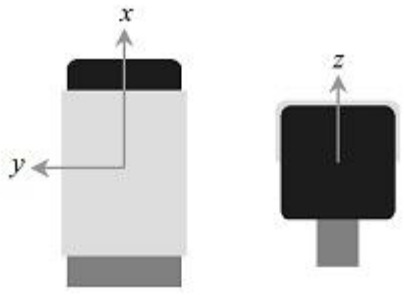


Figura 106.Acelerómetro de Hitechnic para LEGO NXT[26]

4.1.9.1. Clase TiltSensor.

Esta clase es una extensión de I2CSensor por ser este un sensor que usa el protocolo I2C.

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\addon\TiltSensor

Los métodos empleados por la clase son:

Métodos getXTilt (), getYTilt (), getZTilt (): Estos tres métodos funcionan bajo el mismo principio, regresar el grado de inclinación de cada eje al ser llamado. Por ejemplo si se llama el método getxtilt() regresa la inclinación en el eje x o regresa el valor de -1 si hubo algún error. La lectura del sensor cuando esta nivelado con el eje es 128. Los otros 2 funcionan de la misma forma.

Métodos getXAccel (), getYAccel (), getZAccel (): Estos tres métodos se usan para obtener una lectura de aceleración en cada uno de los ejes, x, y, z. Al llamar a alguno de estos métodos estos regresan un valor entero positivo o negativo en unidades mg. Ejemplo la gravedad es 9.8 m/S2 el sensor mostraría 9810.

Para usar el sensor de inclinación o acelerómetro hay que crear un nuevo objeto junto a una referencia a este.

```
TiltSensor tilt = new TiltSensor(SensorPort.S1);
```

Ejemplo:

Este programa muestra en la pantalla LCD la aceleración e inclinación en los tres ejes.

---

[26] Hitechnic, 2009.

```
import lejos.nxt.*;
import lejos.nxt.addon.*;
/**
 * Simple test of Acceleration (Tilt) sensors.
 *
 * This should work with Mindsensors and HiTechnic Acceleration
 * sensors.
 *
 * @author Lawrie Griffiths
 *
 */
public class TiltTest {
	/**
	 * @param args
	 */
	public static void main(String[] args) throws Exception{
		TiltSensor tilt = new TiltSensor(SensorPort.S1);
		while(!Button.ESCAPE.isPressed()) {
			LCD.clear();
			LCD.drawInt(tilt.getXTilt(), 6, 0, 0);
```

```
                LCD.drawInt(tilt.getYTilt(), 6, 0, 1);
                LCD.drawInt(tilt.getZTilt(), 6, 0, 2);
                LCD.drawInt(tilt.getXAccel(), 6, 0, 3);
                LCD.drawInt(tilt.getYAccel(), 6, 0, 4);
                LCD.drawInt(tilt.getZAccel(), 6, 0, 5);
                LCD.refresh();
                Thread.sleep(500);
            }
        }
}
```

5. Uso de accionamientos finales.

Los accionamientos finales son dispositivos que convierten una energía eléctrica (generalmente) en una acción sobre el entorno, sea de luz, mecánica, hidráulica, neumática, etc.

En este contexto el objetivo de los accionamientos finales es intervenir en el comportamiento del sistema robótico, siendo común en robótica móvil que estos elementos se encarguen de generar movimiento a partir de las órdenes recibidas desde el sistema de control. Los accionamientos pueden ser de tipo neumático, hidráulico o eléctrico.

5.1. Accionamientos neumáticos.

La fuente de energía de los accionamientos neumáticos es aire a presión, existen dos clases de estos accionamientos; cilindros neumáticos y motores neumáticos, los

primeros funcionan desplazando un émbolo ubicado dentro de un cilindro, dicho movimiento es producido por diferencia de presiones en los extremos del cilindro. El propósito del cilindro neumático es ubicar el émbolo en los extremos de sí mismo (Barrientos, 1997).

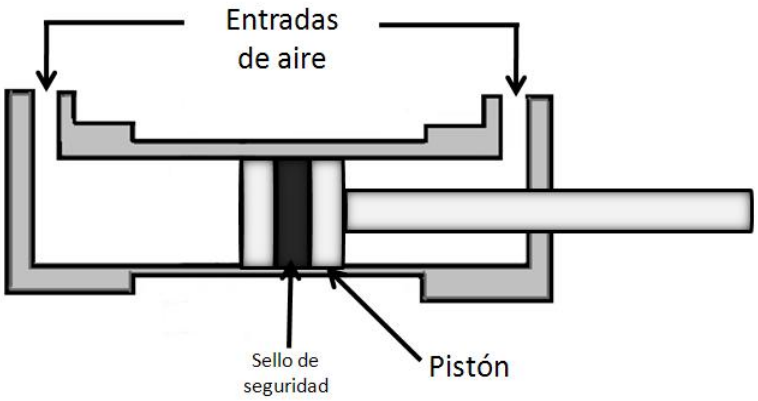


Figura 107.Cilindro neumático de doble efecto

Son los motores de aletas rotativas y los motores de pistones axiales, los motores neumáticos los más utilizados, en los cuales se consigue el movimiento de rotación por medio de aire a presión.

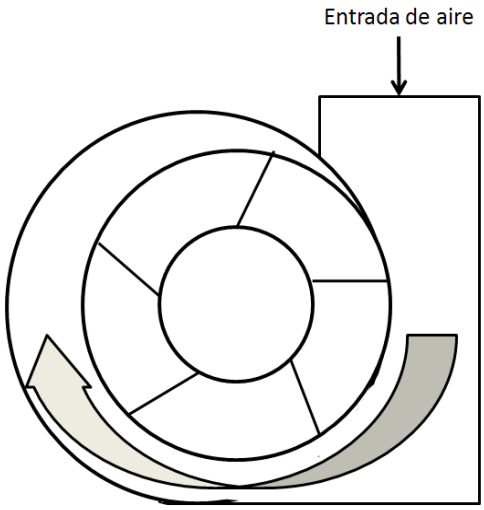


Figura 108.Funcionamiento de un motor neumático

Los accionamientos neumáticos no consiguen alcanzar precisiones muy exactas debido a las características de compresibilidad del aire, impidiendo realizar un adecuado control sobre éstos mismos. No obstante debido a su sencillez este tipo de accionamiento puede ser utilizado en robots que no requieran grandes precisiones.

Se debe tener en cuenta que para utilizar un accionamiento neumático se debe contar con una instalación de aire comprimido, la cual incluye: compresor, tuberías, filtros, secadores, entre otros.

5.2. Accionamientos Hidráulicos.

Funcionalmente son muy parecidos a los neumáticos, la diferencia radica en que en éstos utilizan fluidos, como aceites minerales, con características de compresibilidad inferior a la del aire, permitiendo alcanzar presiones más altas (Barrientos, 1997). Esta diferencia permite alcanzar precisiones mayores, razones por las cuales es más fácil realizar un control continuo sobre éstos; y se hace posible alcanzar mayores pares. Además poseen capacidad elevada de carga, alta lubricación y robustez.

Debido al uso de fluidos en el sistema hidráulico, se hace necesaria la utilización de filtrado de partículas, eliminación de aire, sistemas de refrigeración y unidades de control de distribución, haciendo la instalación más compleja que para los accionamientos eléctricos e inclusive que para los neumáticos. Por otra parte las altas presiones que se manejan propician fugas de aceite sobre toda la instalación.

5.3. Accionamientos eléctricos.

Debido al mejoramiento de las características de control, sencillez y precisión los accionamientos eléctricos han sido más utilizados en los robots, móviles, por lo que a continuación se habla en detalle de estos (Barrientos, 1997).

- Motores de corriente continua (DC): son los más utilizados en la actualidad. Están constituidos por dos devanados internos, inductor e inducido, que se alimentan de corriente continua. El inductor, está situado en el estator y crea un campo magnético de dirección fija. El inducido, situado en el rotor, hace girar al mismo debido a la fuerza que aparece como combinación de la corriente circulante por él y el campo magnético de excitación.

- Motores paso a paso: son capaces de desarrollar pares suficientes en pequeños pasos. Existen tres tipos de motores de paso a paso; de imanes permanentes, de reluctancia variable e híbridos.

- Motores de corriente alterna: No utilizados, al no poderlos controlar con precisión, la velocidad de estos motores se varía con respecto a la frecuencia de la tensión de alimentación; el control de velocidad se realiza por medio de un variador de frecuencia.

Otros tipos de motores: Existen aplicaciones en las que se requieren motores con características específicas, a continuación se describen algunos (Palacios et al., 2004).

- Motores de corriente continúa de pequeña potencia: su número de revoluciones por minuto es elevado y su par es muy pequeño, los que no les

permite ser utilizados en micro-robots a menos que se utilicen reductoras de velocidad o algún otro sistema de regulación.

- Motores de corriente continúa con reductoras: permiten aumentar el par, permitiendo mover el micro-robot con su estructura y fuente de energía que pesa mucho en proporción al tamaño del propio robot.

- Servomotores: los servomotores están construidos por un pequeño motor DC, unas ruedas dentadas que trabajan como reductoras, proporcionándole una potencia considerable, y un circuito con la electrónica necesaria para realizar giros controlados en posiciones angulares específicas por medio del envío de una señal eléctrica codificada.

  El motor que utiliza un servo es un pequeño motor DC, al que al inyectarle voltaje en sus terminales, gira en una dirección a alta velocidad pero con poca fuerza (torque); razón por la cual se hace necesario la utilización de una caja reductora (mecanismo de ruedas dentadas), la cual transforma gran parte de la velocidad de giro en torque.

  Debido a las características de alimentación del motor, voltaje DC, se requiere un circuito capaz de controlar el giro del mismo, conformado principalmente por un conversor de ancho de pulso – voltaje, un amplificador de set-point y un potenciómetro.

  El funcionamiento de un servo consiste en convertir el tren de pulsos de la señal de control en voltaje DC, por medio del conversor de ancho de pulso – voltaje; a la salida de este dispositivo se encuentra un amplificador en el que se establece un set point o punto de referencia que es el valor de posición que se requiere para el motor el cual compara este valor y con el de un potenciómetro que gira con el motor, al cual está conectado, para realizar la comparación del estado en el que se encuentra con el de la señal de control; así se determina cuanto gira el motor para ir a la posición requerida.

Estos dispositivos se utilizan en la práctica para mover palancas, pequeños ascensores y timones; como también en radiocontrol, manejo de títeres y en robots por supuesto. Este tipo de motor es de gran uso en robótica, debido a que posee las características necesarias para realizar tareas de alto torque en aplicaciones de tamaños reducidos.

Los accionamientos finales de Lego Mindstorms NXT son servomotores, los cuales, pertenecen al tipo accionamiento eléctrico.

5.4. Motor LEGO NXT.

Este motor específicamente es para el set NXT (2006). Incluye encoder de rotación, retornando al NXT la posición del eje con una resolución de un grado (1º). Debido al conector especial de este motor (no del tipo RJ-11, conector telefónico), un adaptador de cable es requerido para manejar este motor con fuente regular de 9 voltios. No se recomienda usar con el RCX, el cual, no envía la corriente alta que este motor puede consumir. La velocidad baja de rotación minimiza la necesidad de engranes externos.

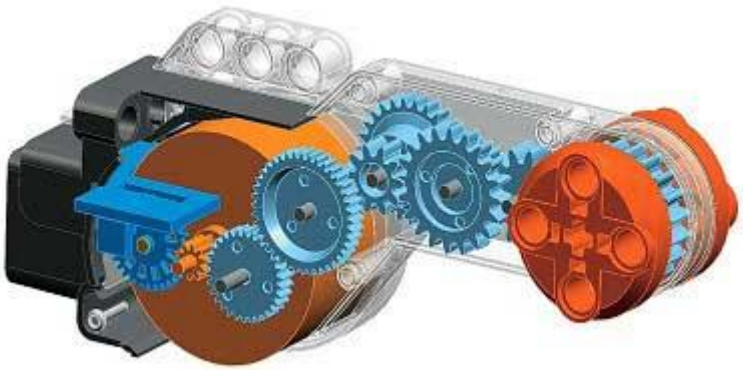


Figura 109.Vista interna del Servo-motor de LEGO NXT[27]

[27] LEGO® MINDSTORMS® (2009). *Vista Interna Servo-Motor Lego NXT.*

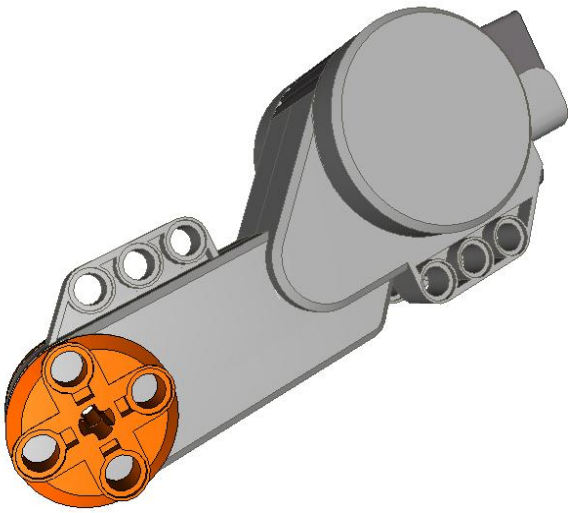


Figura 110.Servo-motor de LEGO NXT

Este motor tiene un peso de 80 gramos.

Características sin carga	
Fuente de potencia	9 voltios
Velocidad de rotación	170 rpm
Corriente sin carga	60 mA

Tabla 25. Características sin carga del servo-motor NXT

Usualmente para los motores DC, la velocidad de rotación es proporcional al voltaje que se les aplica, como se observa en la siguiente grafica. La corriente sin carga depende en poco del voltaje.

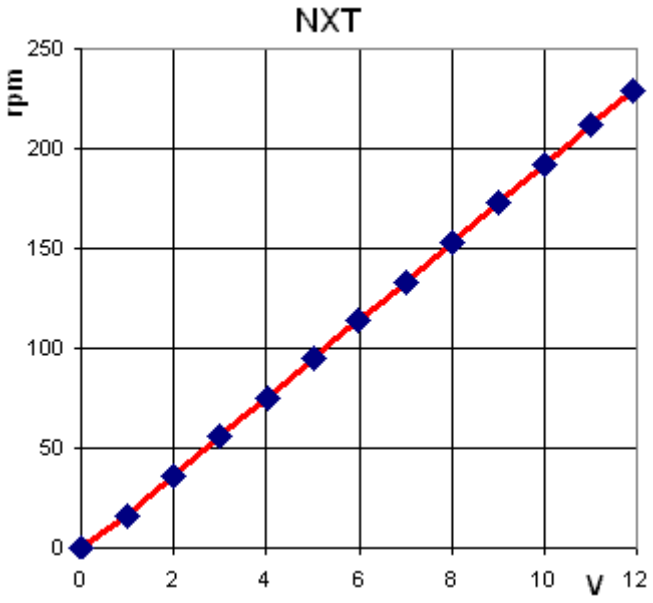


Figura 111.Grafica velocidad de rotación Vs voltaje aplicado[28]

5.4.1. Características en motor bloqueado.

El consumo de corriente de motor bloqueado es medido simplemente con el eje del motor bloqueado con la mano.

Características motor bloqueado	
Fuente de potencia	9 voltios
Torque de motor bloqueado	50 N*cm
Corriente sin carga	2A

Tabla 26. Características motor bloqueado

El motor NXT está protegido también por un termistor (Raychem RXE065 or Bourns MF-R065). Lo que quiere decir que la corriente alta de 2A (y asociada al torque) puede ser prolongada por sólo pocos segundos.

[28] LEGO® MINDSTORMS® (2009). *Gráfica Velocidad de Rotación vs Voltaje Aplicado.*

5.4.2. Características con carga.

Voltaje	Torque	Velocidad de Rotación	Corriente	Potencia Mecánica	Potencia Eléctrica	Eficiencia
4.5 V	16.7 N.cm	33 rpm	0.6 A	0.58 W	2.7 W	21.4 %
7 V	16.7 N.cm	82 rpm	0.55 A	1.44 W	3.85 W	37.3 %
9 V	16.7 N.cm	117 rpm	0.55 A	2.03 W	4.95 W	41%

Tabla 27. Características con carga

Velocidad y corriente vs. Torque

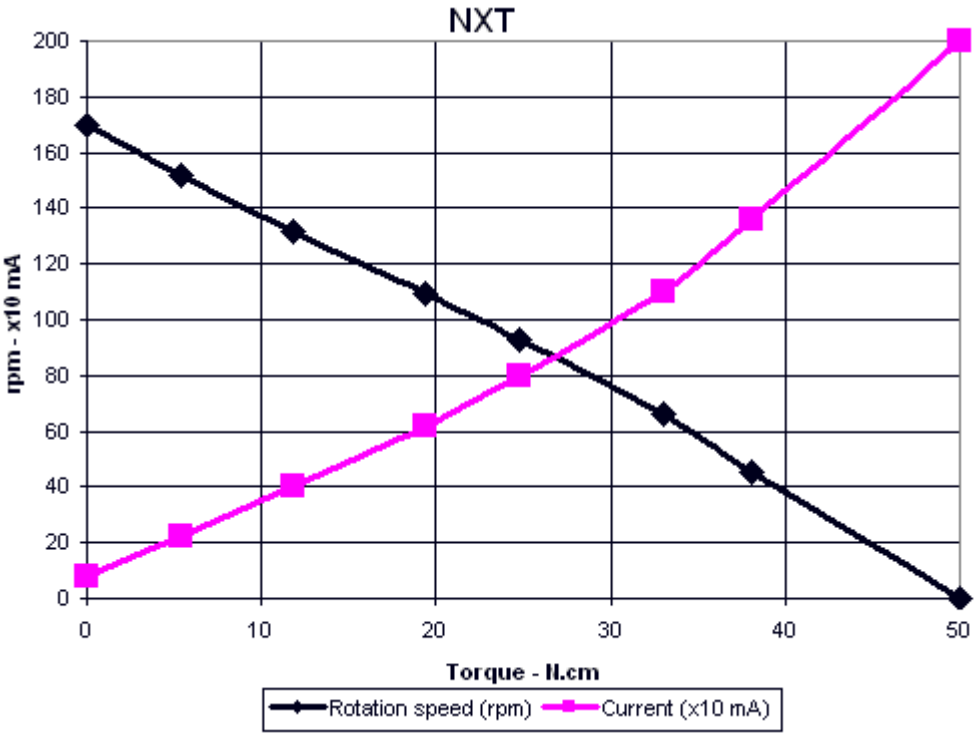


Figura 112.Gráfica de velocidad y corriente vs. Torque[29]

El motor NXT envía un alto torque gracias a su engrane de reducción de velocidad. Debido a que, también da vueltas lentamente y la eficiencia es reducida en una

[29] LEGO® MINDSTORMS® (2009). *Gráfica de Velocidad-Corriente vs Torque.*

pequeña cantidad. Este motor puede ser conectado al RCX gracias a la compatibilidad del cable, pero no es recomendado su uso en el RCX porque la alta corriente consume demasiado y es muy alta para la limitación de corriente de éste que es de 500 mA.

5.5. Clase Motor.

Con la clase motor es posible controlar los motores A, B y C debido a que están disponibles tres instancias de Motor: Motor.A, Motor.B y Motor.C. Para controlar cada motor utiliza los métodos forward, backward, reverseDirection, stop y flt. Para fijar la velocidad del motor utiliza el método setSpeed. La velocidad está en grados por segundo. Los métodos que usan el tacómetro: regulateSpeed, rotate y rotateTo, no pueden detenerse lentamente en el ángulo señalado si se llamaron cuando el motor ya se encuentra en movimiento.

Motor tiene dos modos: speedRegulation y smoothAcceleration los cuales solo trabajan si la regulación de velocidad esta también trabajando. Son inicialmente habilitadas. La velocidad es regulada comparando la cuenta del tacómetro con la velocidad de veces del tiempo transcurrido y ajustando la potencia del motor para mantener éstas estrechamente relacionadas. La aceleración suave corrige la regulación de velocidad para tener en cuenta el tiempo de aceleración. Ellas pueden ser encendidas o apagadas por los métodos speedRegulation y smoothAcceleration. La velocidad máxima actual depende del voltaje de la batería y de la carga. Sin carga, la velocidad máxima esta cerca de 100 veces el voltaje. La regulación de velocidad falla si la velocidad objetivo excede la capacidad del motor. Si se necesita mantener el motor en su posición y se sigue moviendo después de llamar a stop, se puede usar el método lock ().

Detalles del constructor:

public Motor(TachoMotorPort port)

La clase motor usa este constructor para asignar una variable de tipo motor conectado a un puerto particular.

Parámetros:

port – representa el puerto al que el motor está conectado.

Esta clase se encuentra en la siguiente ruta:

\leJOSNXJProjects\classes\lejos\nxt\Motor

5.5.1. Métodos de la clase Motor.

Detalles de los métodos utilizados por la clase Motor:

- public int getStopAngle (): Este método se utiliza para obtener el valor del ángulo en el que se detiene el motor, cuando se le ha asignado uno previamente.

  Ejemplo:

  Motor.A.getStopAngle();//Obtiene el ángulo en el que se detiene el motor.

- public void forward (): Con este método se hace rotar al motor hacia adelante.

  Ejemplo: Motor.A.forward (); // Hace girar el motor A hacia adelante.

- public void backward (): Con este método se hace rotar al motor hacia atrás.

  Ejemplo:

  Motor.A.backward(); // Hace girar el motor A hacia atrás

- public void reverseDirection (): Con este método se invierte la dirección de giro del motor, es decir, si el motor está girando hacia adelante, al aplicar el método cambia el giro hacia atrás y viceversa (el método sólo tiene efecto si el motor esta en movimiento).

- public void flt (): Este método quita la potencia a los motores. El motor perderá toda la potencia, pero esto no es lo mismo que parar. Se usa este método si se quiere que el robot no se detenga abruptamente, sino que lo haga lentamente hasta detenerse.
  public void stop (): Este método se utiliza para detener el motor instantáneamente. En otras palabras, el motor no solo se detiene, sino que también se resistirá a cualquier movimiento.

- public void lock(int power): Con este método es posible mantener el motor en la posición en la que se encuentre, aplicando una cantidad de potencia. Se usa si el método stop () no es suficiente para mantener el motor en posición frente a la carga.

  Parámetros: power – este parámetro puede obtener un valor entre 1 y 100

- public boolean isMoving(): devuelve true si el motor esta en movimiento.

- public void rotate(int angle): ocasiona que el motor gire hasta un ángulo.

  Parámetros: angle – ángulo hasta el cual rotará el motor.

- public void rotate(int angle, boolean immediateReturn): este método ocasiona la rotación del motor hasta un ángulo; si immediateReturn es verdadero, el método retorna inmediatamente y el motor se detiene por si solo si cualquier método de motor es llamado antes de que el límite sea alcanzado, la rotación es cancelada. Cuando el ángulo es alcanzado, el método isRotating () devuelve false;

  Parámetros: angle – ángulo hasta el cual el motor rotará; e immediateReturn – si es true, el método retorna inmediatamente, en consecuencia permite el monitoreo de los sensores en el hilo llamado.

- public void rotateTo(int limitAngle): produce rotación del motor a limitAngle; al terminar la rotación el getTachoCount debe estar entre más o menos 2 grados del ángulo límite cuando el método retorna.

Parámetros: limitAngle – ángulo al cual rotará el motor y se detendrá.

- public void rotateTo(int limitAngle, boolean immediateReturn): hace rotar al motor hasta limitAngle; si immediateReturn es true, el método retorna inmediatamente y el motor se detiene por sí solo y getTachoCount debe estar entre más o menos 2 grados si el ángulo límite, si cualquier método de motor es llamado antes de que el límite sea alcanzado, la rotación es cancelada. Cuando el ángulo es alcanzado, el método isRotating() retorna false.

  Parámetros: limitAngle – ángulo al cual el motor rotará, y luego se detendrá. Si immediateReturn - es true, el método retorna inmediatamente, permitiendo de esta manera el monitoreo de los sensores en un hilo llamado.

- public void shutdown (): ejecuta run () para salir.

- public void regulateSpeed(boolean yes): enciende o apaga la regulación de velocidad.

  El error de velocidad acumulativo está cerca de 1 grado después de la aceleración inicial.

  Parámetros: yes - es true para la regulación de velocidad encendida.

- public void smoothAcceleration (boolean yes): habilita una aceleración más suave. La velocidad del motor se incrementa suavemente, y no la sobrepasa cuando la regulación de velocidad es utilizada.

- public void setSpeed (int speed): establece la velocidad del motor, en grados por Segundo; arriba de 900 grados/segundos solo es posible con 8 voltios.

  Parámetros: speed – valor e grados/segundos.

- public int getSpeed (): retorna la velocidad actual del motor en grados por segundos, cuando se lea asignado una con anterioridad.
- public int getMode (): retorna el modo en el que se encuentra el motor.

  Retorna: 1 = forward, 2= backward, 3 = stop, 4 = float.
- public int getLimitAngle (): retorna en grado el ángulo en el cual se encuentra rotando el motor.
- public boolean isRotating (): retorna true cuando la tarea de rotación del motor no está aún completa en un determinado ángulo.
- public boolean isRegulating (): retorna un valor booleano si está o no regulado, es decir, para true esta regulando y para false lo contrario.
- public int getActualSpeed (): retorna la velocidad actual en grados por segundos, calculada cada 100 ms; los valores negativos significan que el motor está girando hacia atrás.
- public int getTachoCount (): retorna la cuenta del tacómetro en grados.
- public void resetTachoCount (): restablece la cuenta del tacómetro a cero.

5.6. Ejemplos de manejo del motor NXT.

Ejemplo 1:

```
import lejos.nxt.*;

public class Motor1 {

    /**

     * Este programa hace girar al motor hacia adelante y muestra en pantalla
Adelante
```

```
        * espera a que el botón ENTER sea presionado, frena el motor y muestra en pantalla
        * frenado. Espera nuevamente a que la tecla ENTER sea presionada, muestra en pantalla
        * Adelante hasta que sea presionada la tecla ENTER por última vez.
        **/
       public static void main(String[] args) throws Exception {
              LCD.drawString("Adelante", 4, 0);
              Motor.A.forward();
              Button.ENTER.waitForPressAndRelease();
              LCD.drawString("frenado", 4, 0);
              Motor.A.stop();
              Button.ENTER.waitForPressAndRelease();
              LCD.drawString("Atrás", 4, 0);
              Motor.A.backward();
              Button.ENTER.waitForPressAndRelease();
              LCD.drawString("Adelante", 4, 0);
              Motor.A.reverseDirection();
              Button.ENTER.waitForPressAndRelease();
       }
}
```

Ejemplo 2:

```
import lejos.nxt.*;

public class Motor1 {

	/**

	 * Este programa hace girar al motor 2 vueltas, mientras esta rotando muestra

	 * en pantalla la cuenta del tacómetro, la velocidad actual y el ángulo en el que termina la

	 * rotación. Luego realiza la misma acción anterior pero yendo hasta 0

	 **/

	public static void main(String[] args) throws Exception {

		LCD.drawString("Presione tecla", 2, 1);

		Motor.A.regulateSpeed(true);

		Button.waitForPress();

		Motor.A.rotate(720,true);

		while(Motor.A.isRotating()){

			LCD.drawInt(Motor.A.getTachoCount(),4, 0, 4);

			LCD.drawInt(Motor.A.getActualSpeed(), 4, 6,4);

			LCD.drawInt(Motor.A.getStopAngle(), 4, 12, 4);

		}

	Button.waitForPress();
```

```
        Motor.A.rotateTo(0, true);
        while(Motor.A.isMoving()){
                LCD.drawInt(Motor.A.getTachoCount(),4, 0, 4);
                LCD.drawInt(Motor.A.getActualSpeed(), 4, 6,4);
                LCD.drawInt(Motor.A.getStopAngle(), 4, 12, 4);
            }
        Button.waitForPress();
         }
}
```

6. Practica.

Programa 1

Hacer un programa que realice lo siguiente:

Al presionar el botón ENTER se mueva el motor A hacia adelante durante dos segundos, pasado este tiempo el motor se detenga y mostrar en pantalla en la posición “x, y” (0,5), que el motor A se detuvo.

Realizar el mismo procedimiento del paso 1 con el motor B.

Realizar el procedimiento del paso 1 con los motores A y B girando al mismo tiempo durante 10 segundos, pasado este tiempo hacer girar a los motores hacia atrás.

Programa 2

Hacer un programa que realice lo siguiente:

Al presionar el botón ENTER se mueva el motor A 90º.

Al presionar el botón ENTER se mueva el motor B 90º.

Al presionar el botón ENTER se muevan los motores A y B 90º.

Programa 3

Hacer un programa que al presionar cualquier botón mueva el "motor A" a -720º a una velocidad regulada (con el método regulateSpeed()) y muestre en la LCD la velocidad.

Sin limpiar la pantalla, realizar el paso anterior con el motor B.

Programa 4

Hacer un programa que realice lo siguiente:

Al presionar el botón ENTER el motor A gire hacia 720º

Detener el motor luego de alcanzar los 720º

Muestre en pantalla la cuenta del tacómetro.

Vuelva a colocar el tacómetro en cero.

Al presionar el botón ENTER, el motor realice los pasos del 1 al 3 a una velocidad de 500.

Programa 5

Hacer un programa que realice lo siguiente:

Hacer girar al motor 180º.

Mostrar en pantalla la lectura del tacómetro, la velocidad actual y el ángulo en el que se detiene el motor.

Hacer girar al motor al ángulo cero o ángulo de inicio.

Mostrar en pantalla la lectura del tacómetro, la velocidad actual y el ángulo en el que se detiene el motor.

Volver a realizar los pasos del 1 al 4 con una velocidad de 100.

Programa 6

Hacer un programa que muestre en pantalla la lectura del sensor ultrasónico en modo normal, del sensor de luz en modo de ambiente, del sensor de color en el modo de mostrar un número correspondiente a la tabla de colores, y del compás. Conectados de la siguiente forma.

Sensor ultrasónico = s1

Sensor de color =s2

Sensor de luz =s3

Sensor compás = s4

El valor del ultrasónico tiene que salir en la posición “x, y” (0,0)

El de color tiene que salir en la posición “x, y” (0,1)

El valor de lectura del compas tiene que salir en la posición “x, y” (0,2)

El valor de lectura del de luz tiene que salir “x, y” (0,3)

Programa 7

Hacer un programa que utilice el compás en modo de referencia, se apunta hacia un punto marcado, y que en pantalla se muestre la lectura del sensor referente a este punto.

Programa 8

Hacer un programa que muestre en pantalla la lectura del sensor de color en el modo RGB, y muestre cada uno de estos parámetros durante 2 segundos, luego se

borre la pantalla, y muestre la lectura del sensor de compas a intervalos de 200 ms, registrándolos en pantalla desde la posición (0,0) hasta la posición (15,7).

Programa 9

Hacer un programa que muestre en pantalla la lectura del sensor de aceleración, con la lectura de aceleración y de inclinación en los tres ejes, en las posiciones que el alumno considere necesario, de tal forma que se pueda observar los 6 datos.

Programa 10

Hacer un programa que muestre en pantalla la lectura del giroscopio (para observar cambios en este sensor hay que mover el sensor de forma circular, pues este mide aceleración angular), que muestre la lectura del sensor de sonido, y que muestre si el sensor de contacto ha sido o no presionado.

Programa 11

Realizar un programa que tome datos del sensor de sonido y comparar los dos diferentes modos de lectura el modo dB, y en modo dBA.

Laboratorio 4
Diseño e implementación de controladores Difuso y PID en robótica móvil.

1. Objetivos.

1.1. General.

Estudiar y comprender los sistemas de control difuso y controlador PID.

1.2. Específicos.

- Conocer los sistemas de control que son aplicables a robótica móvil.
- Comprender el funcionamiento de los controladores difuso.
- Diseñar un controlador difuso.
- Comprender el funcionamiento de los controladores PID.
- Diseñar un controlador PID.
- Diferenciar las características de un controlador difuso frente a un PID.
- Conocer las aplicaciones de los controladores difusos.
- Conocer las aplicaciones de los controladores PID.

2. Metodología.

En esta práctica los estudiantes deberán estar familiarizado con el entorno de trabajo de eclipse, tener los conocimientos fundamentales de java y saber descargar un programa al ladrillo NXT.

Antes de realizar la práctica los estudiantes deben haber leído toda la información proporcionada acerca de los diferentes tipos de control.

Durante el desarrollo de la guía se encontrarán aportes teóricos, ejercicios y ejemplos de programas, sobre control difuso y control por PID.

La práctica consta de un ejercicio de lógica difusa y la realización de un programa PID, que deberán ser resueltos y presentados la siguiente clase. En el informe entregado se debe documentar detalladamente cada programa desarrollado.

Los estudiantes deberán descargar al ladrillo los ejemplos y observar el funcionamiento del programa mostrado en la guía. Esto con el fin de afianzar sus conocimientos.

NOTA: Para el buen desarrollo de este laboratorio se recomienda leer la información contenida en el Anexo B.

3. Materiales.

- Ladrillo LEGO NXT.
- Servo motor lego NXT.
- Conectores para los sensores y accionamientos finales.
- 6 baterías AA (preferiblemente recargable).
- Sensores ultrasónicos.
- Cable USB o dispositivo bluetooth.
- PC con leJOS NXJ, Eclipse instalado y configurado[30].

[30] El computador donde se trabajará tiene que tener eclipse configurado para descargar programas al ladrillo a través de leJOS NXJ.

4. Control difuso.

4.1. Descripción del programa difuso.

Se busca controlar la distancia de un robot hacia algún objeto frente a él, con lo que se requiere controlar la velocidad para hacerlo avanzar, frenar o disminuir de una manera “suave”. Se establece como set point la distancia de 50 cm, y el comportamiento será:

- A medida que el Robot se encuentre más cerca del objeto, la velocidad será mayor y el robot retrocederá.
- A medida que el Robot se encuentre más lejos del objeto, la velocidad será mayor y el Robot avanzará.
- Cuando el Robot alcance el set point debe detenerse.

4.2. Control de velocidad y distancia aplicando lógica difusa.

La lógica difusa está basada en que todo no es totalmente cierto, sino que cada verdad tiene un poco de incertidumbre. Un ejemplo, algo no es totalmente negro, o totalmente blanco, hay infinitas tonalidades intermedios entre estos dos colores, las cuales pueden ser mezclas entre varios colores, por lo que un verde puede tener 25% de azul y 75 % de rojo, lo que la lógica difusa hace, es precisamente tener en cuenta estos valores intermedios y para el caso del ejemplo incluirlos en los conjuntos. Por ende se trabaja con pertenencias a cada conjunto y combinaciones de estos, sin hablar de pertenencias absolutas. Este tipo de control se ha convertido en el preferido para programación de robots móviles, debido a su alta respuesta a comportamientos no lineales, y teniendo una estructura lógica similar a la forma como razonan los seres humanos, facilita considerablemente la labor de diseño del controlador. El control difuso se encuentra dividió en tres partes, fusificación, implicación, defusificación, las cuales se explican.

4.2.1. Fusificación.

La primera parte del control difuso es la fusificación, consiste en convertir los valores de entrada obtenidos por el sensor ultrasónico en conjuntos difusos, en este caso, la diferencia de error, y luego convertir este valor a correspondencia en los diferentes conjuntos difusos. Entre los diferentes modelos a representar gráficamente de conjuntos difusos, se opta por los más usados, que son los triángulos y trapecios, debido a su fácil representación, su fácil calculo, y su poco consumo de recursos digitales. Sin embargo hay otras figuras para representar el conjunto difuso tales como “semi-ciclo sinodal”, “semi-círculo”, etc. Lo ideal es escoger una figura adecuada para obtener respuestas más suaves en la variable controlada, sin embargo esto implica formulas matemáticas más complejas, que requieren mayor capacidad de procesamiento, lo que conlleva a un tiempo de ejecución mayor del programa. La desventaja es significativa en los casos en donde el retardo saque al controlador de “tiempo real”, en donde se decremento considerablemente la capacidad de respuesta del sistema ante los eventos externos.

Para los valores externos en las entradas, se representan con un trapecio, lo que quiere decir que después de cierto valor, el valor de salida para ese conjunto siempre será de 1 (es decir la máxima).

En cuanto al número de variables a usar, son dos de entrada y una de salida; el error de distancia el cual es la diferencia entre el set point y el valor medido por el sensor ultrasónico, y la diferencia de error de distancia, el cual es una diferencia de un nuevo cálculo de error de distancia, con el calculado ya previamente (esto se hace para evitar en lo posible respuestas tardías con respecto a la medida), y la salida es la velocidad.

Conjuntos:

E=vmed — vset

Esto nos da el valor de error, siendo vmed el valor medido por el sensor ultrasónico, su rango de medida es de 0 a 127cm Max, cuando la distancia es mayor a 127 cm la salida es de 255

- Los conjuntos son:

  Error muy muy negativo (mMN)

  Error muy negativo (Mun)

  Error medio negativo (Mn)

  Error negativo(n)

  Error 0(E0)

  Error positivo (p)

  Error medio positivo (mP)

  Error muy positivo (mUP)

  Error muy muy positivo (mMP)

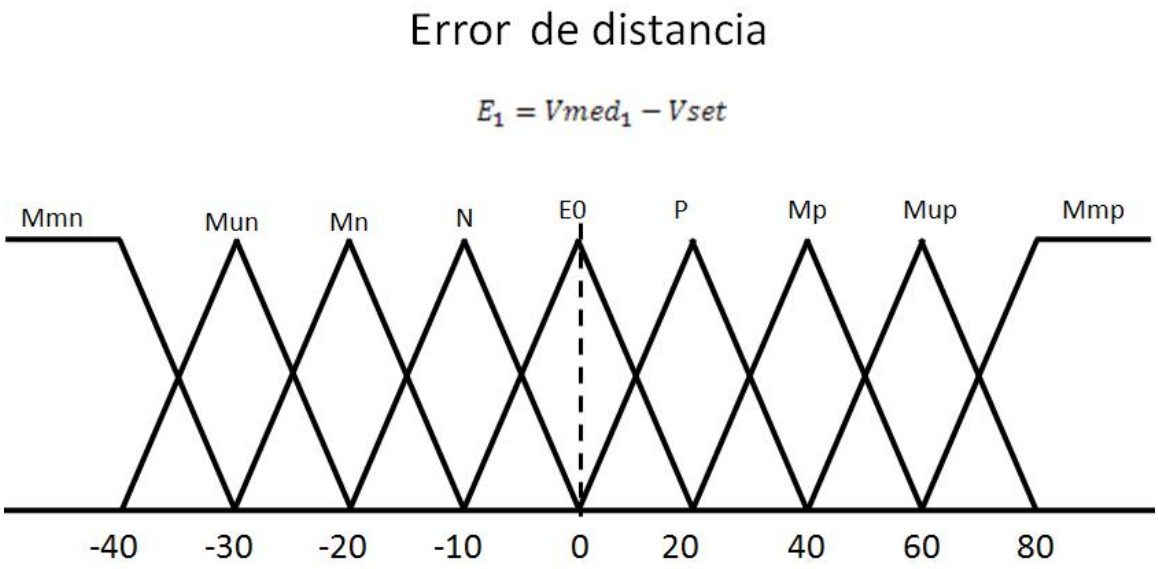


Figura 113.Error de distancia

En la figura 80 podemos ver la representación grafica de cada uno de los conjuntos difusos.

La pertenencia a cada conjunto se calcula usando la ecuación y=mx+b.

En la figura se puede observar que los rangos de cada conjunto negativo son diferentes a los rangos de los conjuntos positivos (-10, -20, -30,-40 para los negativos) (20, 40, 60, 80 para los positivos) esto es porque los valores negativos son menos que los positivos, así también como se observo que el sensor después de 127 cm marca 255, por eso el valor máximo positivo es 80.

Diferencia de error: Se realiza el mismo proceso con la diferencia de error. Y los conjuntos son:

Diferencia de error muy muy negativa.

Diferencia de error muy negativa.

Diferencia de error medio negativa.

Diferencia de error negativa.

Diferencia de error 0.

Diferencia de error positiva.

Diferencia de error medio positiva.

Diferencia de error muy positiva.

Diferencia de error muy muy positiva.

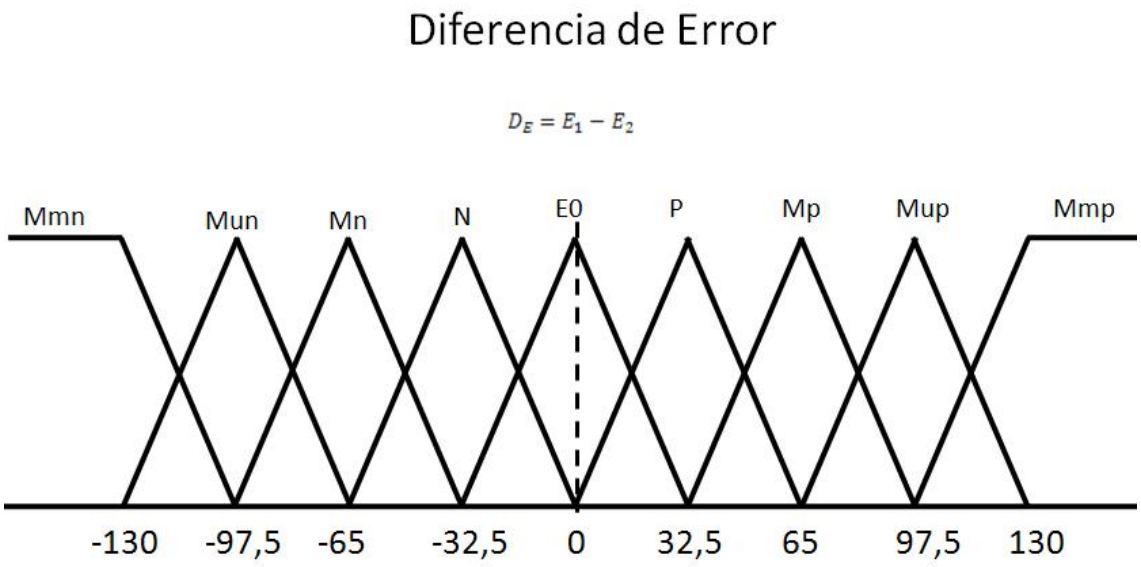


Figura 114.Diferencia de error

Los rangos de los conjuntos para la diferencia de error, se escogieron pensando las dos medidas más extremas, la diferencia de error máxima posible para el lado negativo, y para el lado positivo.

La variable de salida: La variable de salida es la velocidad. Y en la figura 82 están representados todos los conjuntos difusos que contiene esta.

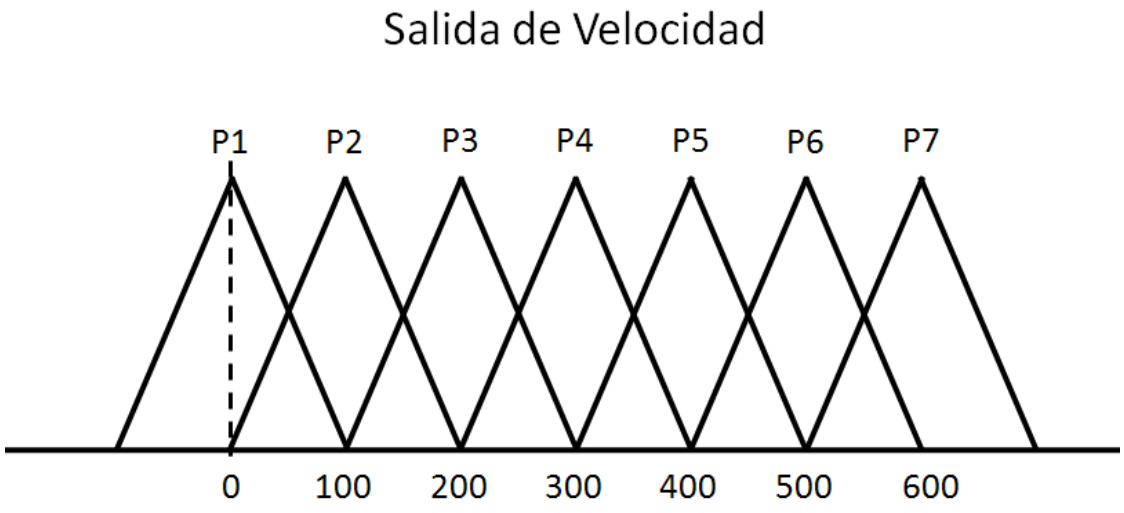


Figura 115.Variable de salida

La velocidad máxima se toma como 600 RPM, siendo un valor promedio, debido a que la velocidad de los motores solo puede ser 100 veces más que la batería, y sabiendo que la batería suele encontrarse entre 8,4 y 7,2 voltios normalmente. Se tomó un valor menor a estos.

4.2.2. Implicación.

En esta sección es donde se analizan las diferentes reglas y combinaciones establecidas.

El método a seguir es el método de Mandani. Y las reglas tienen la forma.

Si la velocidad es muy muy negativa y la diferencia es muy muy negativa la salida es p4.

Las reglas están representadas en la tabla siguiente.

	dmmn	dmun	dmn	dn	De0	dp	dmp	dmup	dmmp
Mmn	P4	P4	P5	P6	P7	P6	P5	P4	P4
Mun	P3	P3	P4	P5	P7	P5	P4	P3	P3
Mn	P2	P2	P3	P5	P6	P5	P3	P2	P2
N	P1	P2	P3	P3	P4	P3	P3	P2	P1
E0	P2	P1	P1	P1	P1	P1	P1	P1	P2
P	P1	P2	P3	P3	P4	P3	P3	P2	P1
Mp	P2	P2	P3	P5	P6	P5	P3	P2	P2
Mup	P3	P3	P4	P5	P7	P5	P4	P3	P3
Mmp	P4	P4	P5	P6	P7	P6	P5	P4	P4

Tabla 28. Aplicando reglas

Con esta combinación de reglas se obtienen los mμ y las salidas.

Los mμ son la combinación de las reglas entre cada una de la tabla de regla.

Ejemplo si el error es mmn=0.7 y la dmmn= 0.3 la salida es p4

Mu (1)= Min (0.7, 0.3) = 0.3

Salida (1)= p4

Todas estas series de combinaciones se analizan por lógica difusa, cuando la regla sea “y” se realiza el mínimo entre ambos valores, si la regla es “o” se regresa el máximo. (Ver Métodos para las reglas en el Anexo B, introducción a los tipos de control) Y el valor de salida es el correspondiente con respecto a la tabla de reglas establecidas previamente.

4.2.3. Defusificación.

Esta es la parte donde se convierten los valores difusos a valores o números correctos (resultado numérico), el método usado para hacer esto es el más común y que menos capacidad de procesamiento requiere, dando un valor equilibrado de prestaciones Vs necesidades. Se llama el centro de gravedad.

La formula resultante es:

$$Z = \frac{m\mu_1 * (salida1) + m\mu_2 * (salida2) + m\mu_3 * (salida3) + \cdots ... m\mu_n * (salida n)}{m\mu_1 + m\mu_2 + m\mu_3 + \cdots . m\mu_n}$$

En el caso de control de velocidad dependiendo de la distancia medida por el sensor ultrasónico, se analizan todas las combinaciones de las reglas y al final se obtiene según el análisis de reglas un valor de velocidad de salida.

4.3. Código de Control de velocidad y distancia aplicando lógica difusa.

Para el control de distancia se importan las clases, pilot del paquete navigation; esta clase se utiliza para controlar el robot con dos ruedas y rueda loca, esta tiene los principales comandos para la operación de un robot de este tipo; tales como ir adelante, ir atrás, girar tantos grados, avanzar tantas unidades, detenerse. (Revisar documentación para mayor información).

math del paquete java lang, el cual se utiliza en el proceso de implicación para hallar los mínimos entre dos parámetros. Y todo el paquete nxt de lejos. Con el que se utilizan ciertas clases para manejo de botones, LCD.

4.3.1. Usando las clases pilot y ultrasonicsensor.

Se comienza describiendo como son creadas e inicializados los objetos de tipo “Pilot” y “Ultrasonicsensor”, los cuales son utilizados en la implementación del algoritmo del presente laboratorio.

Para los objetos de tipo “Pilot” puedan funcionar necesitan como parámetros el tamaño de las ruedas, la distancia que las separa entre sí, a que puertos del ladrillo están conectados y cuál es el derecho y el izquierdo; los cuales son pasados como parámetros al método constructor cuando es creado el objeto. La configuración del robot con el que se trabaja es la siguiente:

- Las llantas tienen de diámetro 5.6 cm
- La distancia entre las llantas es de 10.5 cm
- Los servomotores están conectados al puerto “a” (motor Izquierdo) y al puerto “c” (motor derecho).

A continuación se muestra como es implementado en código la creación del objeto.

```
final Pilot robot = new Pilot(5.6f, 10.5f,Motor.A, Motor.C);
```

Se utiliza el sensor ultrasónico de lego, en el puerto cuatro. Esto se hace creando las referencias al nuevo objeto donde en parámetro se indica en que puerto se conecta el sensor.

```
final UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S4);
```

4.3.2. Declaración de variables.

Hay diferentes grupos de variables usadas en el diseño del algoritmo, las variables de entrada que son lectura de distancia, las cuales son 9 variables pertenecientes a cada uno de los conjuntos difusos, tienen la forma de nombres lingüísticos, tales como muy muy negativo (muyMuyN), (para más información de las variables revisar documentación de código) la variable de salida es la velocidad. Las variables

de los conjuntos difusos de salidas que van desde “p1” hasta “p7”, las variables utilizadas en la combinación de reglas difusas que son: $m\mu_n$, $salida_n$, y en la formula general de la defusificación.

Abajo se detalla el tipo de variable declarado para cada uno de los casos:

Las variables de salida desde “p1” hasta “p7” las cuales son las representaciones en el código de los “conjuntos difusos de salida”, son declaradas como final int por ser constantes y ser enteros.

Los “mu1” hasta “mu81”, las variables difusas, y las salida 1 hasta la salida81 son declarados como double, para obtener más precisión al momento de los cálculos.

Las variables lingüísticas de los conjuntos de entrada y de salida son declaradas double, por trabajar con precisión. Todas estas variables son variables de la clase fuzzydist3.

Las variables de error calculado por primera vez, diferencia de error, error calculado por segunda vez, y valor obtenido por el sensor ultrasónico, son declaradas entero. Al igual que el valor del set point.

4.3.3. Descripción de métodos.

Método inicializacionReglas.

Este método se llama antes de entrar al ciclo del programa, para ahorrar tiempo de ejecución y respuesta del robot, es la sección donde se asignan las salidas desde salida (1) hasta salida (81).

Ejemplo:

```
salida81=p4;
```

Método fusificaciónDistancia.

Este método se encarga de evaluar la pertenencia a cada conjunto difuso, del valor error de distancia. Sacando un valor de 0 a 1 de cada conjunto. Se realiza evaluando cada intervalo de las graficas representativas de este conjunto.

Lo primero es calcular el error de distancia.

$$E_1 = V_{med} - V_{set}$$

Luego se evalúan todos los intervalos.

Por ejemplo para evaluar la pertenencia de “muy muy negativo” (muyMuyN) y muy negativo (muyNegativo) en el intervalo “-40 < error < -30”.

Ejemplo:

```
if ((e1>=-40)&&(e1<=-30)){
muyMuyN=((-0.1*e1)-3);
        muyNegativo=((0.1*e1)+4);
        return;
        }
```

Donde la función es de la forma “$y = mx + b$” y la “m” y “b” han sido calculados previamente como constantes para cada intervalo.

Para la evaluación de los conjuntos representados gráficamente como trapecios simplemente se evalúan los intervalos mayores del límite del rango.

Código:

```
if (e1<-40){
        muyMuyN=1;
```

```
        return;
}
if (e1>80){
        muyMuyP=1;
        return;
        }
```

Se evalúa e para cualquier rango de error “error <-40”, el valor regresado será de “muy muy negativo” igual a 1 (muyMuN= 1).

Y lo mismo con e>80 el valor regresado será muymuyp = 1

Método fusificaciónErrorDistancia.

En este método se evalúa la diferencia de error, para su correspondencia en conjuntos difusos. Esto se hace por medio de una serie de cálculos (en el caso de esta aplicación con ecuación de la línea recta $y = mx + b$ y así hallar un valor representativo de 0 a 1 para cada conjunto. Se realiza evaluando cada intervalo de las graficas de este conjunto.

Se calcula un error medido nuevamente, para luego compararlo con la diferencia de este con el valor calculado primero; obteniendo la diferencia de error.

$$E_2 = e_1 - e_2$$

$$D_e = e_1 - e_2$$

El proceso es similar al método “fusificación Distancia” (Ítem 0), lo único que varia son los rangos de la grafica, pero el proceso es el mismo, se evalúan cada intervalos con ciclos “if”, para regresar los valores de cada conjunto.

Método implicación.

En este método se evalúan la combinación de las 81 reglas preestablecidas y se calculan los mu usando la clase math.min, la cual regresa el valor mínimo entre dos argumentos evaluados.

Ejemplo:

mu81=Math.min(muyMuyP,dMuyMuyP);

Acá "mu81" tendrá el valor mínimo entre "muy muy positivo" (muyMuyP), y "diferencia muy muy positiva" (dMuyMuyP).

Método defusificación.

En este método se regresan los valores difusos a valores numéricos del mundo. Se usa el "método de centro de gravedad" la formula general es:

$$Z = \frac{m\mu_1 * (salida1) + m\mu_2 * (salida2) + m\mu_3 * (salida3) + \cdots ... m\mu_n * (salida n)}{m\mu_1 + m\mu_2 + m\mu_3 + \cdots . m\mu_n}$$

Y la formula queda de esta forma.

Código:

```
velocidad=((mu1*salida1)+(mu2*salida2)+(mu3*salida3)+(mu4*salida4)+(mu5*s
alida5)+(mu6*salida6)+(mu7*salida7)+(mu8*salida8)+(mu9*salida9)+(mu10*sal
ida10)+(mu11*salida11)+(mu12*salida12)+(mu13*salida13)+(mu14*salida14)+(m
u15*salida15)+(mu16*salida16)+(mu17*salida17)+(mu18*salida18)+(mu19*salid
a19)+(mu20*salida20)+(mu21*salida21)+(mu22*salida22)+(mu23*salida23)+(mu2
4*salida24)+(mu25*salida25)+(mu26*salida26)+(mu27*salida27)+(mu28*salida2
8)+(mu29*salida29)+(mu30*salida30)+(mu31*salida31)+(mu32*salida32)+(mu33*
salida33)+(mu34*salida34)+(mu35*salida35)+(mu36*salida36)+(mu37*salida37)
```

```
+(mu38*salida38)+(mu39*salida39)+(mu40*salida40)+(mu41*salida41)+(mu42*sa
lida42)+(mu43*salida43)+(mu44*salida44)+(mu45*salida45)+(mu46*salida46)+(
mu47*salida47)+(mu48*salida48)+(mu49*salida49)+(mu50*salida50)+(mu51*sali
da51)+(mu52*salida52)+(mu53*salida53)+(mu54*salida54)+(mu55*salida55)+(mu
56*salida56)+(mu57*salida57)+(mu58*salida58)+(mu59*salida59)+(mu60*salida
60)+(mu61*salida61)+(mu62*salida62)+(mu63*salida63)+(mu64*salida64)+(mu65
*salida65)+(mu66*salida66)+(mu67*salida67)+(mu68*salida68)+(mu69*salida69
)+(mu70*salida70)+(mu71*salida71)+(mu72*salida72)+(mu73*salida73)+(mu74*s
alida74)+(mu75*salida75)+(mu76*salida76)+(mu77*salida77)+(mu78*salida78)+
(mu79*salida79)+(mu80*salida80)+(mu81*salida81))/

((mu1+mu2+mu3+mu4+mu5+mu6+mu7+mu8+mu9+mu10+mu11+mu12+mu13+mu14+mu15+mu16+
mu17+mu18+mu19+mu20+mu21+mu22+mu23+mu24+mu25+mu26+mu27+mu28+mu29+mu30+mu3
1+mu32+mu33+mu34+mu35+mu36+mu37+mu38+mu39+mu40+mu41+mu42+mu43+mu44+mu45+m
u46+mu47+mu48+mu49+mu50+mu51+mu52+mu53+mu54+mu55+mu56+mu57+mu58+mu59+mu60
+mu61+mu62+mu63+mu64+mu65+mu66+mu67+mu68+mu69+mu70+mu71+mu72+mu73+mu74+mu
75+mu76+mu77+mu78+mu79+mu80+mu81));
```

Método trayectoria.

Este método es el que evalúa si el robot se moverá hacia delante o hacia atrás, o se detendrá. Esto se hace por una estructura sencilla de “if else” evaluando el error medido.

Código:

```
if(e0==1) robot.stop();
            //return;*/
                  //}
```

```
if(e1<0)robot.backward();

else if (e1>0)robot.forward();

}
```

Main, método principal del programa.

En el flujo principal del programa se crea una referencia a la clase principal,

se llama al método inicializaciónreglas

Luego se entra en un bucle while que se ejecutara mientras el botón escape no se encuentre presionado.

Se llama a los métodos en el siguiente orden.

fusificacionDistancia, fusificacionErrorDistancia, implicación, defusificación,

Para asignar la velocidad al motor se usa la referencia robot creada y se hace de la siguiente manera:

Código:

```
fd.robot.setSpeed((int) fd.velocidad);
```

Luego se llama al método trayectoria.

Finalmente se muestra a manera de comprobación en la LCD las diferentes variables, distancia, error de distancia, velocidad. Entre otras.

5. Control PID.

El objetivo del programa es controlar la velocidad de los motores del robot dependiendo de la distancia medida por un sensor ultrasónico entre un objeto y el robot mismo; se ha establecido un set point determinado, el cual puede variar entre 5 y 250. Entre más lejos se encuentre un objeto del robot y se detecte un valor de medida por encima del valor de set point, los motores incrementan su

velocidad y girarán hacia adelante, proporcionalmente a la distancia. Cuando el robot se encuentre a menor distancia del set point la velocidad se irá incrementando en la medida que el objeto se encuentre más cerca haciendo girar las ruedas en reversa.

5.1. Control de distancia utilizando un controlador PID.

Un controlador PID corrige el error entre la variable de proceso medida y la deseada de set-point, y realiza una acción correctiva sobre la salida la cual será proporcional al error, con lo que se puede mantener las condiciones deseadas del proceso de manera adecuada y rápida.

El controlador PID está compuesto por tres controladores o términos; el valor proporcional determina la reacción al error actual, el valor integral determina la acción basado en la suma de los errores recientes y el valor derivativo determina la reacción basado en la tasa de los errores que han estado cambiando.

La ecuación del PID está determinada por la siguiente ecuación:

$$MV(t) = P_{out} + I_{out} + D_{out}$$

Donde MV es la variable manipulada P_{out}, I_{out} y D_{out} son las contribuciones (términos) del controlador PID a la salida.

Pero antes de realizar el algoritmo para cada término es preciso tratar el error, ya sea normalizando la variable del proceso o el error como tal. Para obtener este valor del error se debe tratar la variable controlada, en este caso la distancia, la cual depende de la medida del sensor ultrasónico que trabaja en un rango de ± 3 y 127 cm. Dado el caso que él los objetos se encuentren a más de 127 cm se obtiene un valor de 255. El error es calculado realizando la resta entre el set point y la variable del proceso.

$$Error = SP - VP$$

Luego de haber calculado el error es posible calcular los términos proporcional, integral y derivativo; en este programa sólo se emplea un sencillo controlador PD, encargado de controlar la velocidad de los motores del robot. De lo que resulta una ecuación como esta:

$$MV(t) = P_{out} + D_{out}$$

La variable manipulada MV, obtiene valores porcentuales de 0–1, valor que se multiplica por la salida que se espera, que es 500º/segundo como máxima velocidad y 0 como mínima.

Para controlar la dirección de giro de los motores se pregunta primero si la variable manipulada es mayor, igual o menor que el set point. En el caso que la MV sea mayor que el set point el robot se dirige hacia adelante, si es cero se detiene y si es menor se dirige hacia atrás.

5.2. Código del controlador PID.

5.2.1. Importar Paquetes.

Lo primero que se requiere hacer para comenzar a codificar el controlador es importar las clases de los paquetes nxt y navigation de lejos:

```
import lejos.nxt.*;

import lejos.navigation.*;
```

5.2.2. Crear Objetos.

Se utilizan las clases Pilot y UltrasonicSensor para crear un nuevo objeto de cada clase.

Para crear un objeto tipo Pilot es necesario tener un robot con sistema de locomoción tipo triciclo clásico, como también se debe conocer el diámetro de las llantas, la distancia entre las llantas y definir el motor derecho es izquierdo. A partir de esto se crea el objeto, así:

```
Pilot robot = new Pilot(5.6f, 10.5f,Motor.A, Motor.C);
```

Para crear el objeto UltrasonicSensor se necesita conocer el puerto en el que se encuentra conectado el sensor ultrasónico

```
UltrasonicSensor sonic=new UltrasonicSensor(SensorPort.S4);
```

5.2.3. Declaración de variables.

Se declaran las variables utilizadas en el programa:

```
public double Error;//Error actual

public double Error1;//Error anterior

public double ErrorN;//Error Normalizado

public double SPDist=10;//Distancia set point

public double P;//Valor Proporcional

public double D;//Valor Derivativo

public double I;//Valor Integral

public double U;//Salida del proceso PID

public double SenLec;//Lectura del Sensor

public double VSal;//Velocidad de Salida

public double Kp=1;//Constante Proporcional

public double Ki=1;//Constante Integral
```

```
public double Kd=1;//Constante Derivativa
```

5.2.4. Descripción de métodos.

Luego de realizar los tres primeros pasos es posible crear los métodos que posibilitan el control. Estos métodos se describen a continuación.

Método PID.

Este es el método primordial de todo el algoritmo, es por así decirlo, el cuerpo de éste.

Lo primero que realiza este método es leer el valor medido por el sensor y almacenarlo en una variable:

```
SenLec=sonic.getDistance ();
```

Luego se entra un ciclo de if y else anidados, donde se trata la variable leída por el sensor para obtener el valor del error:

```
if(SenLec<=SPDist){
                    if(SenLec<6){
                        Error=SPDist-5;
                    }else {
                        Error=SPDist-SenLec;
                    }
                }else if(SenLec>SPDist){
                    if(SenLec>=2*SPDist){
                        Error=SPDist-5;
                    }else if (SenLec<2*SPDist&&SenLec>((2*SPDist)-5)){
```

```
            Error=SPDist-5;
        }else{
            Error=-(SPDist-SenLec);
            }
    }
```

Se toma el valor del error y se normaliza, es decir, que a partir de esta operación el error obtiene valores entre 0 y 1.

ErrorN=Error/(SPDist-5);

Se calculan los términos proporcional, integral, derivativo, y el controlador total:

```
P=Kp*(ErrorN);
        D=Kd*(ErrorN-Error1);
    U=P+D;
```

Este valor del controlador PID varía de acuerdo de los cambios en el valor de la distancia. Para obtener el valor de la variable manipulada se debe multiplicar por un factor de 500, dando origen al valor de velocidad de salida:

```
VSal=500*U;
```

Método actualizar.

Este método actualiza el valor del error y lo almacena en una nueva variable:

```
public void Actualizar(){
        Error1=ErrorN;
        }
```

Método leading.

Este método define qué debe hacer el robot, si ir hacia adelante, ir hacia atrás o detenerse. Lo primero que hace es comparar la lectura del sensor con el set point, si la primera es mayor que la segunda se dirige hacia adelante, si es el caso contrario se dirige hacia atrás y si son iguales se detiene:

```
public void leading(){
        if (SenLec>SPDist){
            robot.forward();
        }else if(SenLec==SPDist){
            robot.stop();
        }else{
            robot.backward();
        }
    }
```

Método Main.

En este método contiene los métodos que hacen posible el control y movimiento del robot, además muestra en pantalla los valores relevantes para la operación de control de velocidad.

```
public static void main (String[] args) throws Exception{
        PID1 pD=new PID1();
        System.out.println("Presione boton para iniciar");
        Button.waitForPress();
```

```
        while(!Button.ESCAPE.isPressed()) {
            pD.PID();
            pD.robot.setSpeed((int)pD.VSal);
            pD.leading();
            pD.Actualizar();
            LCD.clear();
            LCD.drawString("Lect sen: "+pD.SenLec, 0, 0);
            LCD.drawString("EN: "+pD.Error, 0, 1);
            LCD.drawString("U: "+pD.U, 0, 4);
            LCD.drawString("P: "+pD.P, 0, 2);
            LCD.drawString("D: "+pD.D, 0, 3);
            LCD.drawString("Velo: "+pD.VSal, 0, 5);
            LCD.refresh();
            Thread.sleep(200);

        }
    }
```

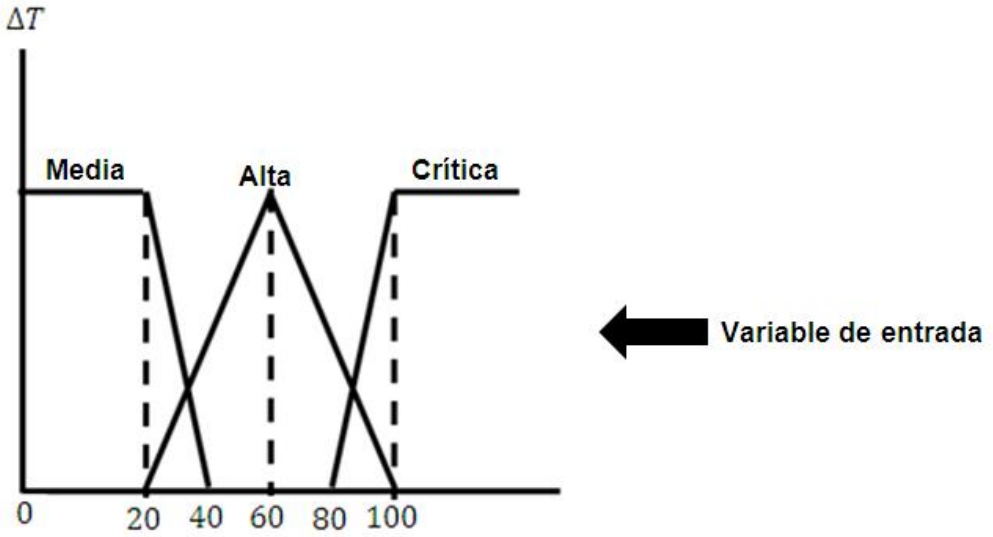
6. Practica.

1. Realizar un controlador PID que dependa de la cantidad de luz y que controle la velocidad de los motores.

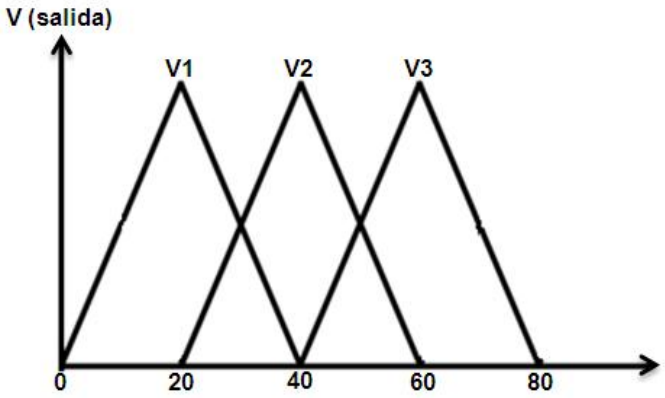
2. Realizar los cálculos para hallar la pertenencia a los conjuntos difusos en el siguiente ejercicio.

   Se requiere controlar la temperatura en un tanque, para esto se controlara el flujo del liquido refrigerante, a través del control de una bomba. La temperatura es medida por un sensor que entrega un rango de 0-100ºC.

   El Set point es de 30 º C. los tres conjuntos de entrada son: media, alta, crítica.



   Las variables de salida seria la bomba que controla el flujo del líquido refrigerante.



Para una lectura de:

a) 30

b) 50

c) 70

d) 75

e) 99

Calcule la salida del controlador, para cada uno de los valores de lectura.

Laboratorio 5
Comportamientos más comunes en robótica aplicaciones y tareas principales.

1. Objetivos.

1.1. General.

Conocer los principales comportamiento de los robots móviles, de tipo triciclo.

1.2. Específicos.

- Comprender el proceso de diseño de una tarea de robótica móvil.
- Comprobar la aplicación del control difuso en la robótica móvil.
- Construir robots móviles con el kit Lego Mindstorms NXT.
- Programar al robot para realizar una o varias tareas establecidas.
- Explicar los diferentes programas realizados para cada tarea o aplicación.
- Ampliar los conocimientos del las fases de la robótica pedagógica.

2. Metodología.

En esta práctica el grupo deberá estar familiarizado con el entorno de trabajo de eclipse, tener los conocimientos fundamentales de java y saber descargar un programa al ladrillo NXT.

Antes de realizar la práctica los estudiantes deben haber leído toda la información sobres sensores y accionamientos finales existentes en la robótica móvil, los métodos y clases necesarias que se usarán en ésta guía, los cuales están registrados en las guías anteriores a esta.

La práctica consta de un diseño base para ser armado, de cuatro aplicaciones con sus respectivas guías de armados y un programa para cada aplicación, con los cuales los grupos pueden ampliar su conocimiento en robótica móvil.

Se recomienda a los grupos estudiar, descargar al ladrillo y observar cada programa mostrado en la guía, con el fin de consolidar los conocimientos y proponer mejoras.

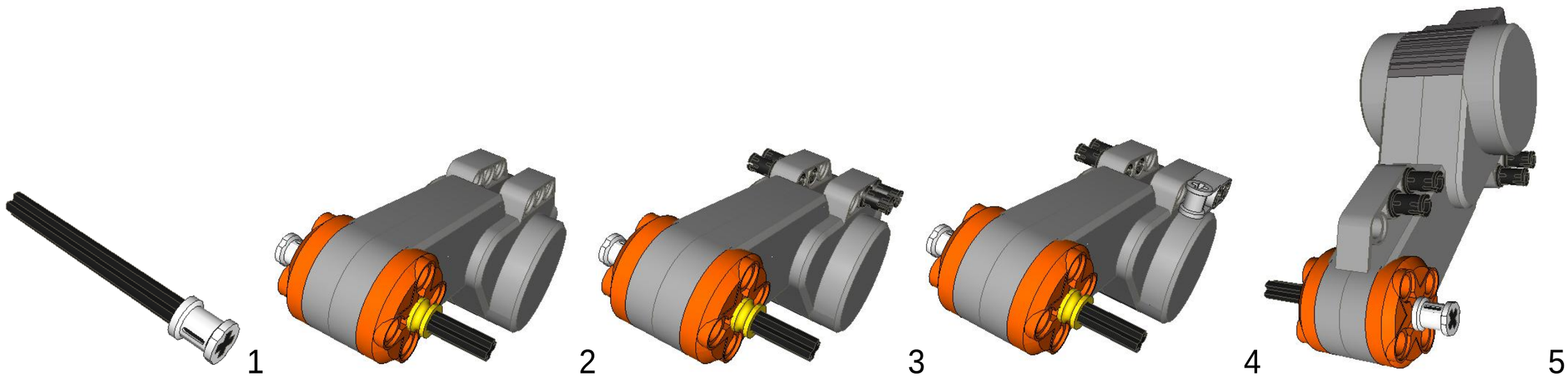
3. Materiales.

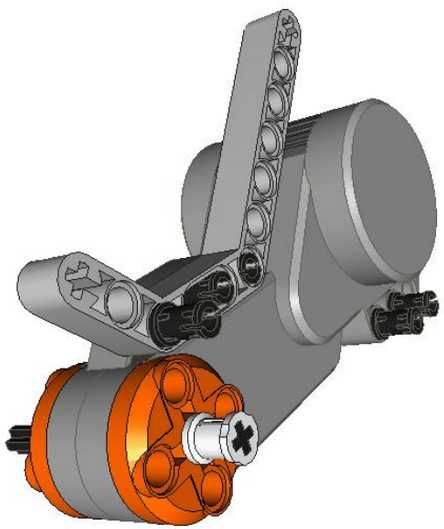
- Kit de Lego Mindstorms NXT (caja completa).
- 6 baterías AA o batería recargable.
- Cable USB o dispositivo bluetooth.
- PC con leJOS NXJ, Eclipse instalado y configurado.

4. Diseño base.

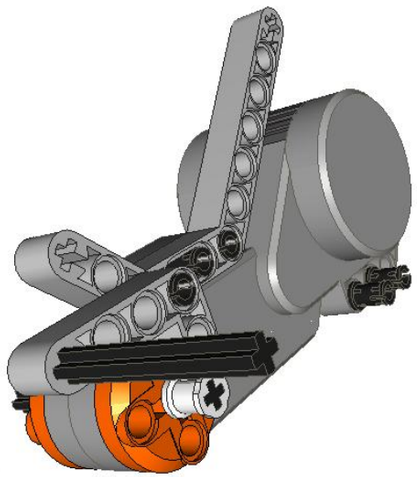
Para las diferentes tareas a realizar por el Robot pedagógico se escogió un diseño base (Estructura física fija), que puede ser modificado sin mayores complicaciones, ajustando el Robot para una tarea determinada.

4.1. Guía de armado del diseño base.


1
2
3
4
5

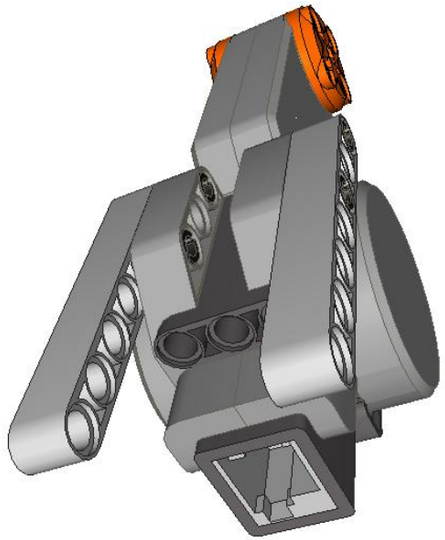


6

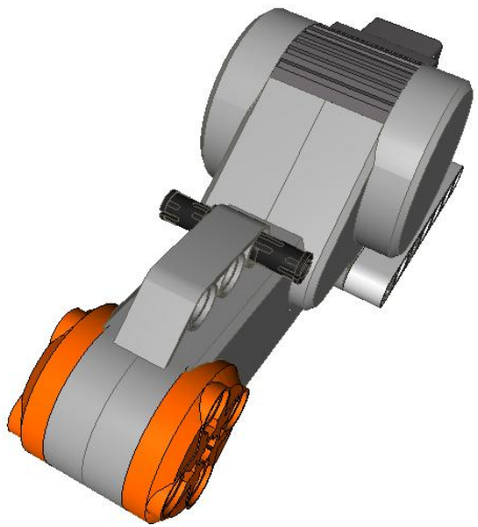


7

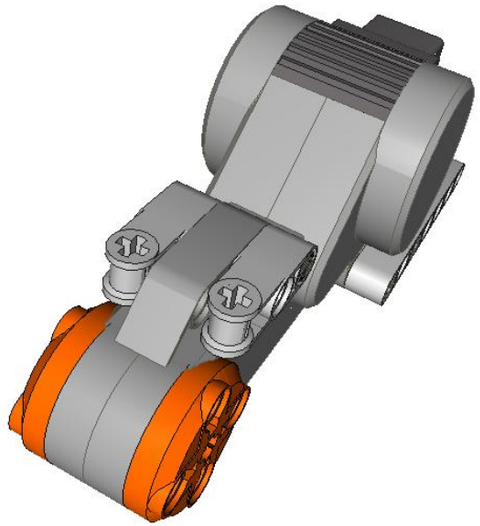
8.1



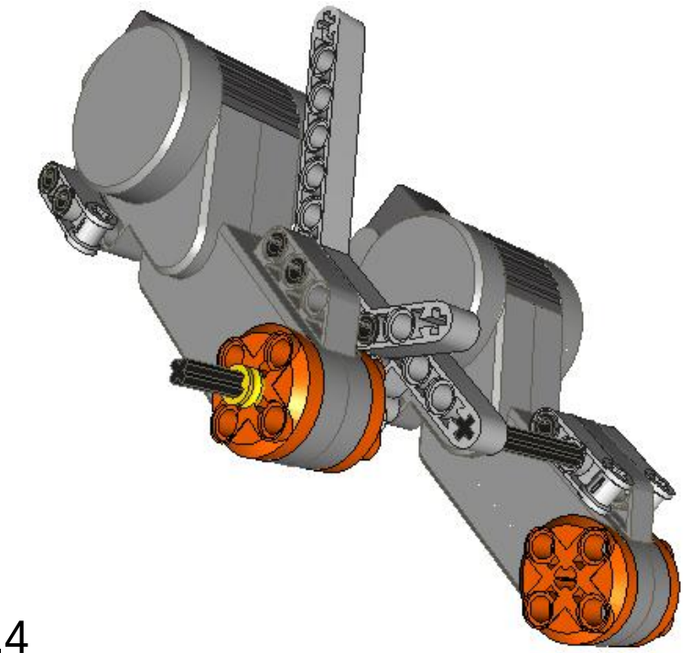
8.2



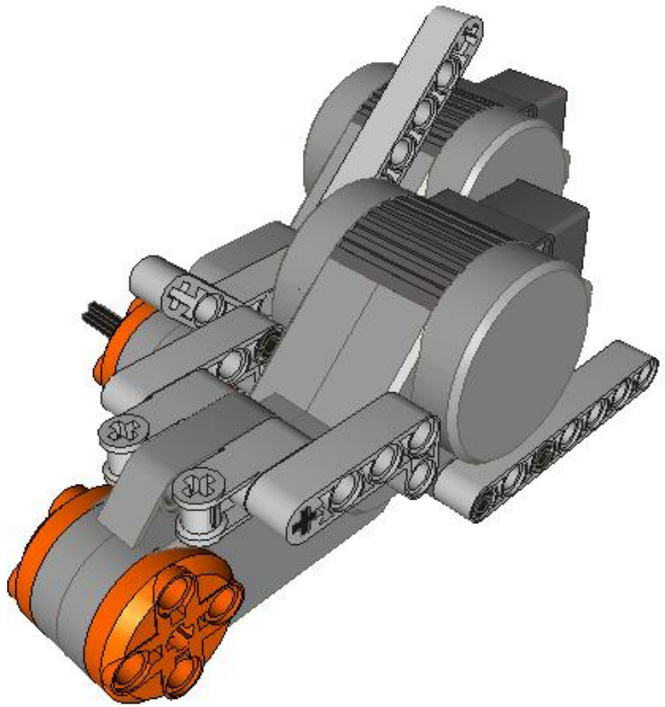
8.3



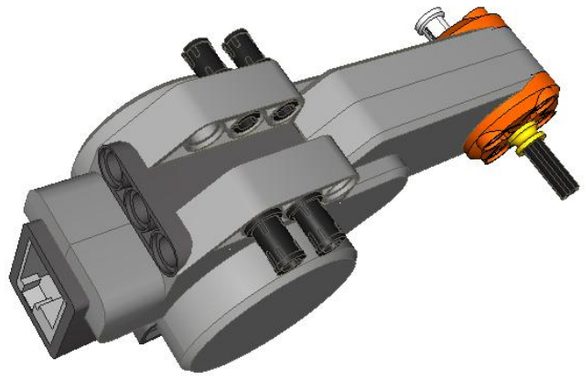
8.4



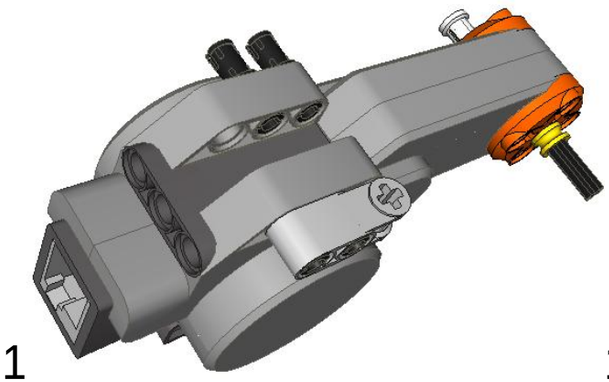
8.5



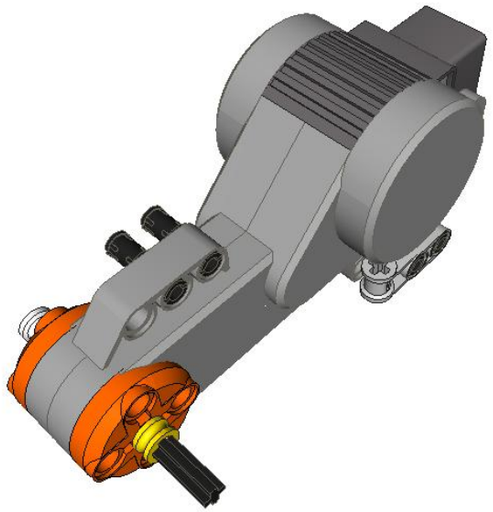
9



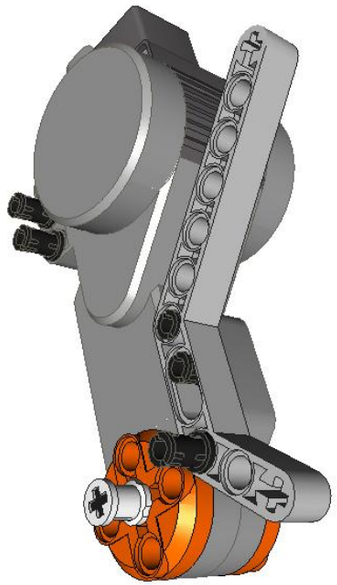
10.1



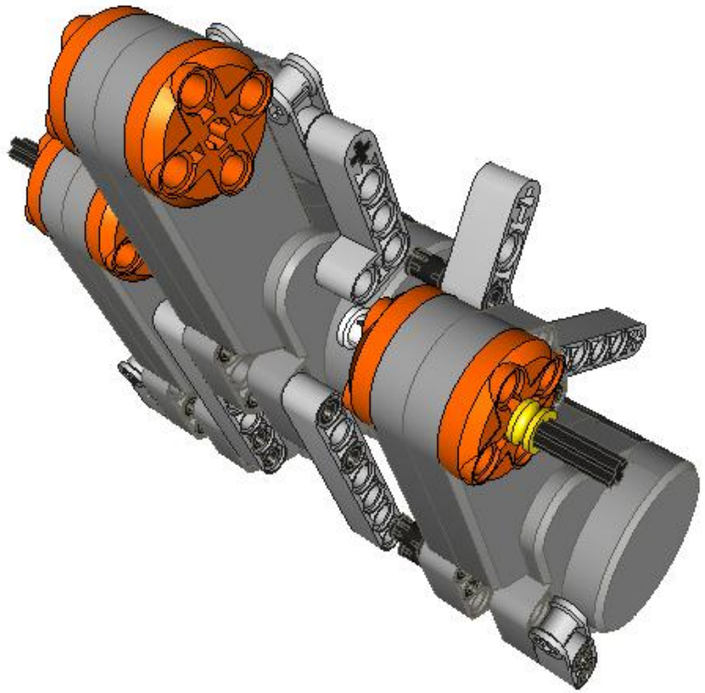
10.2



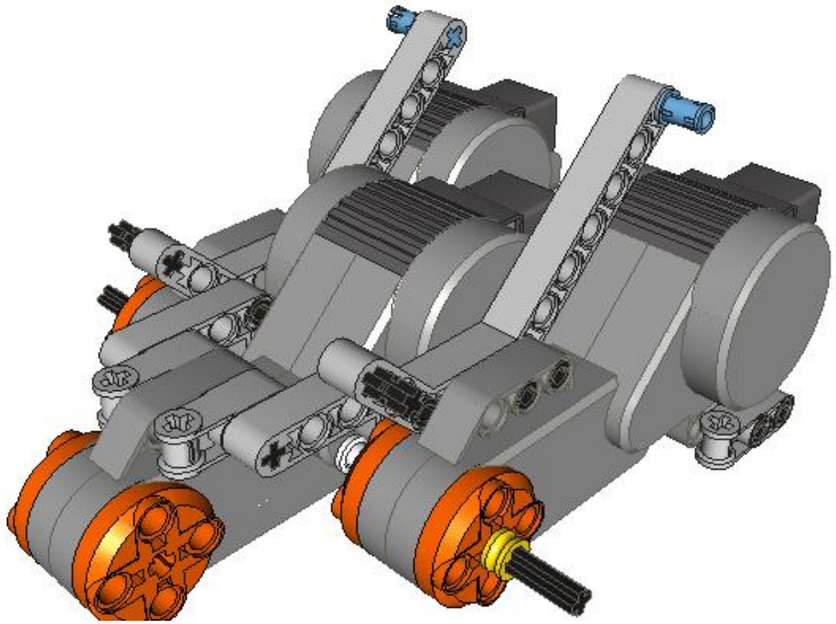
10.3



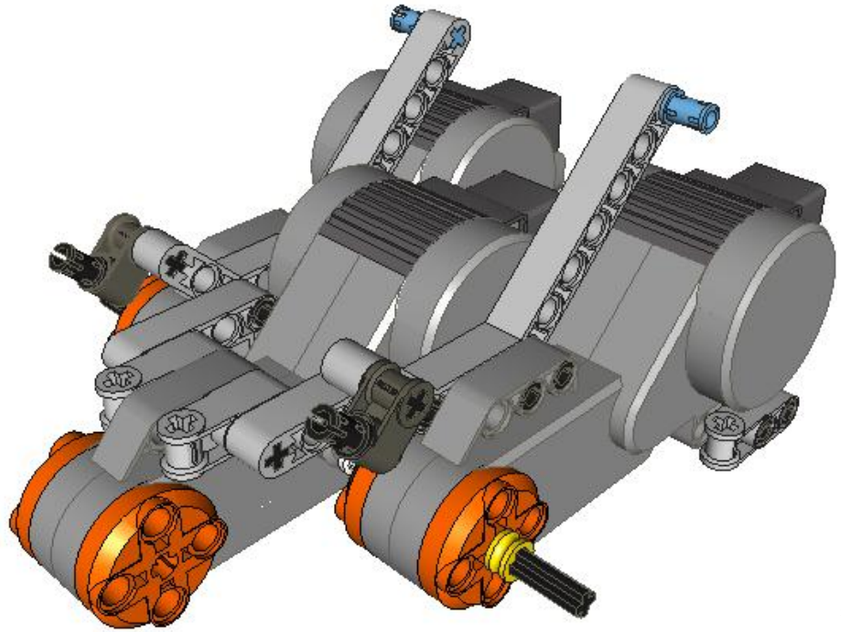
10.4



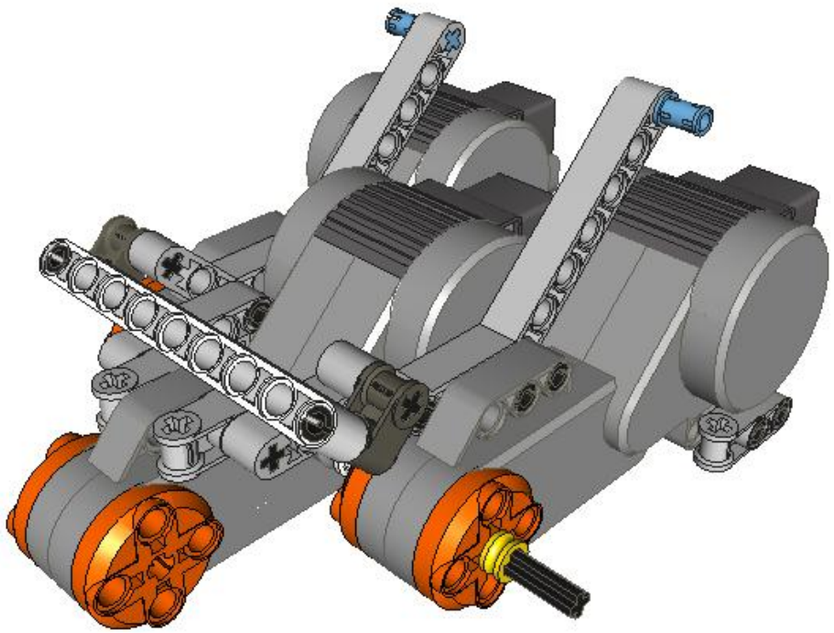
10.5



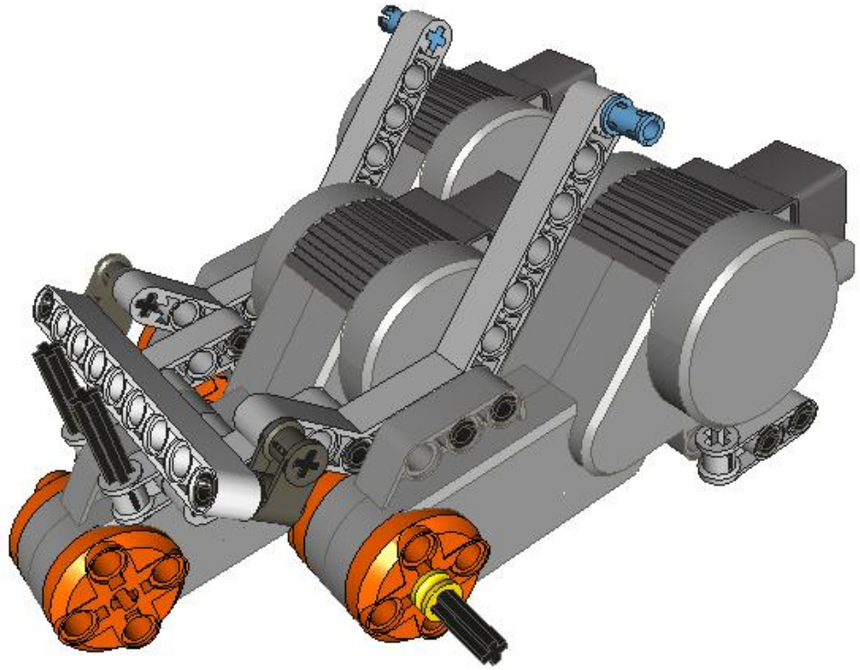
11



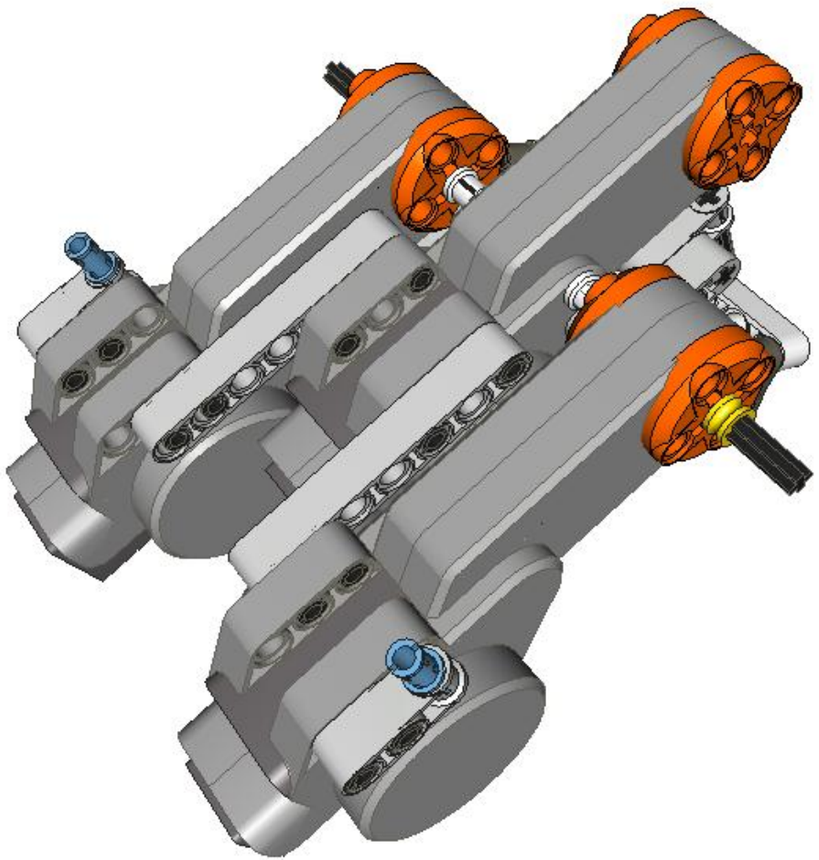
12



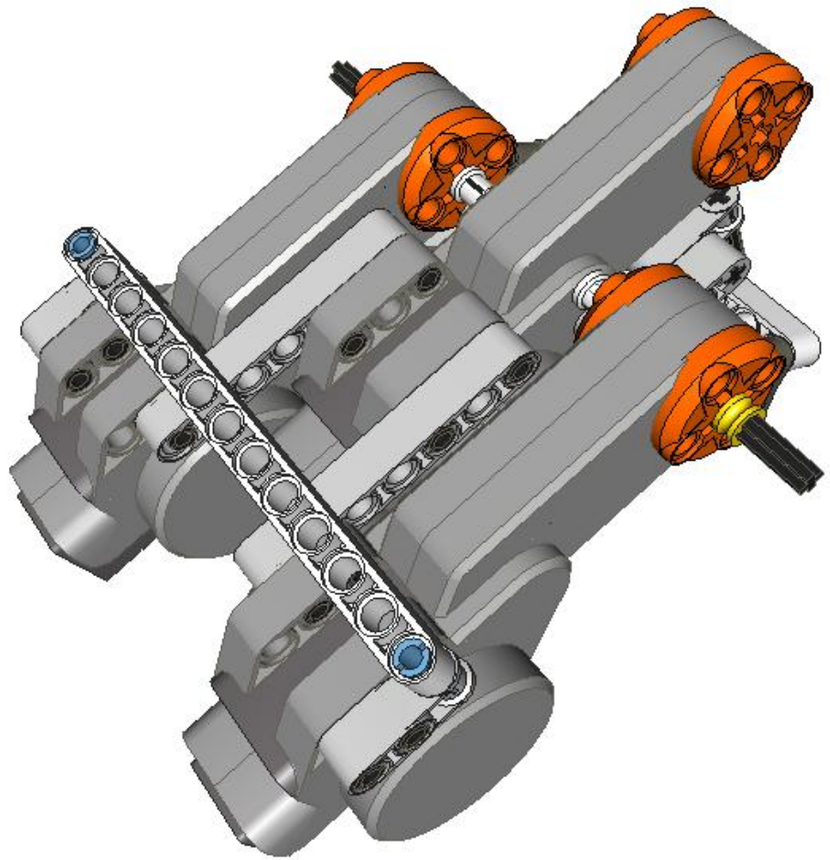
13



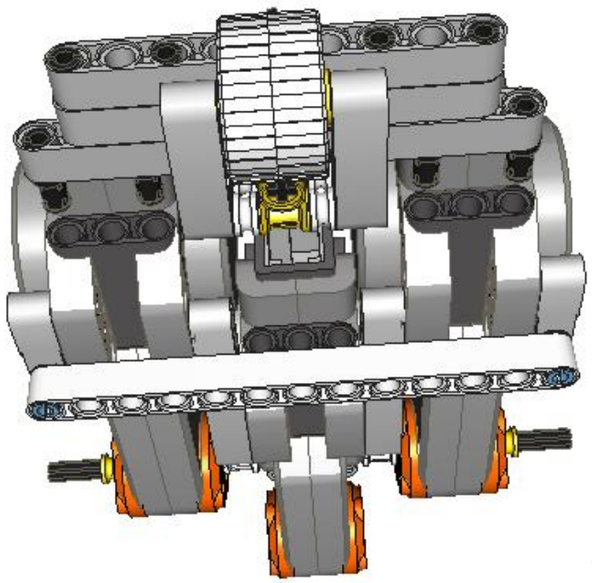
14



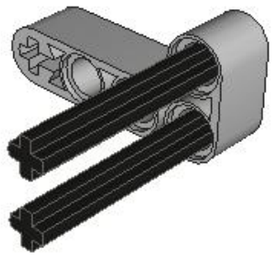
15



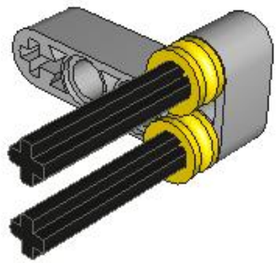
16



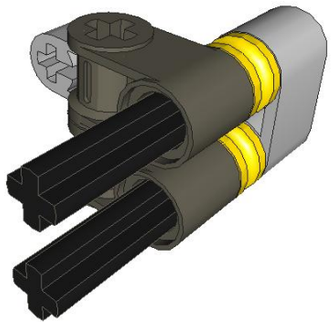
17



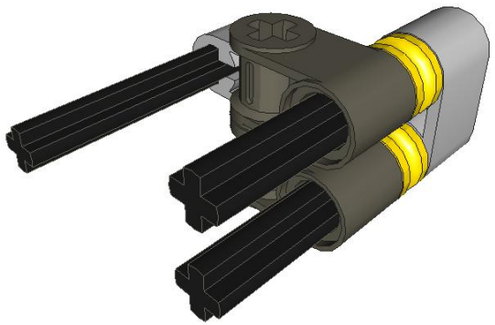
17.1



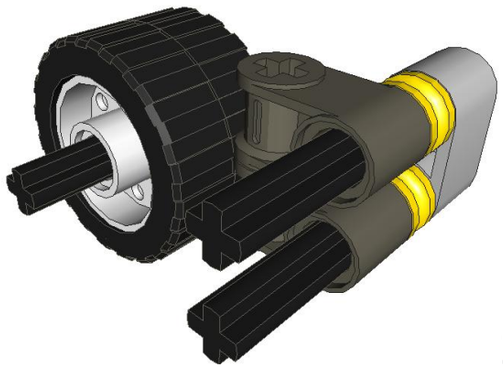
17.2



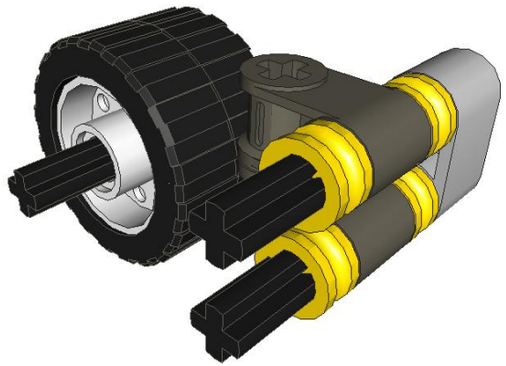
17.3



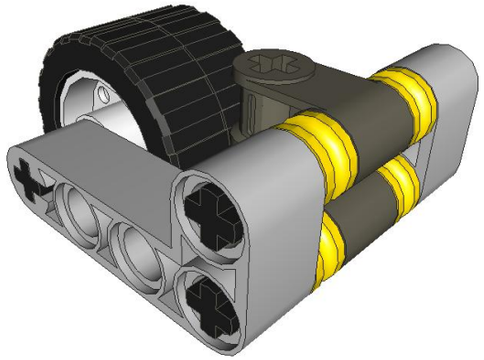
17.4



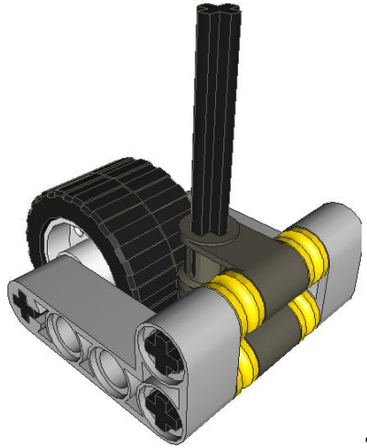
17.5



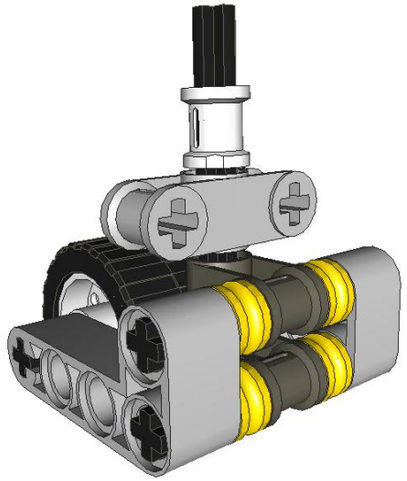
17.6



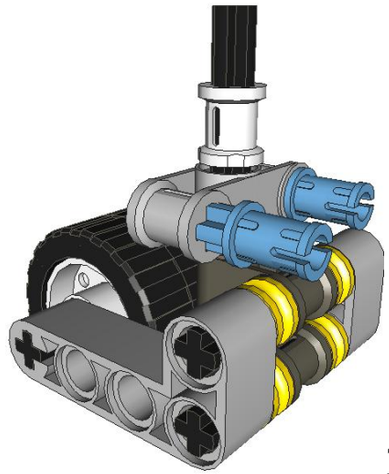
17.7



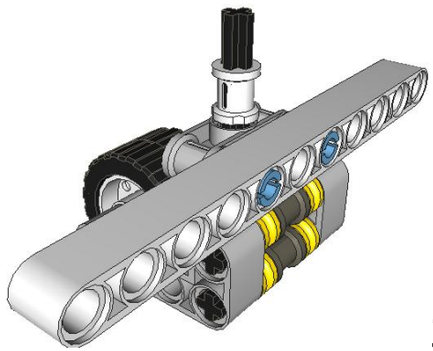
17.8



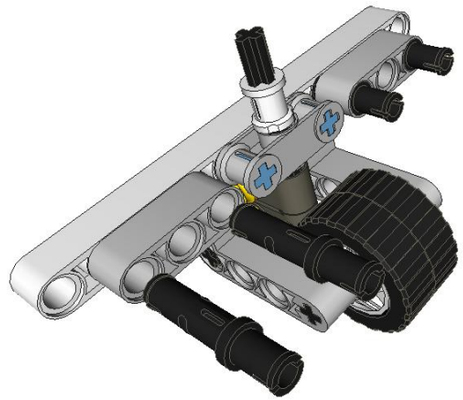
17.9



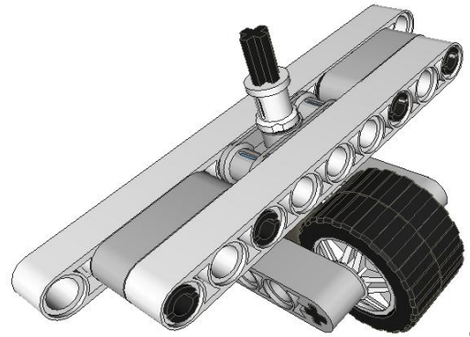
17.10



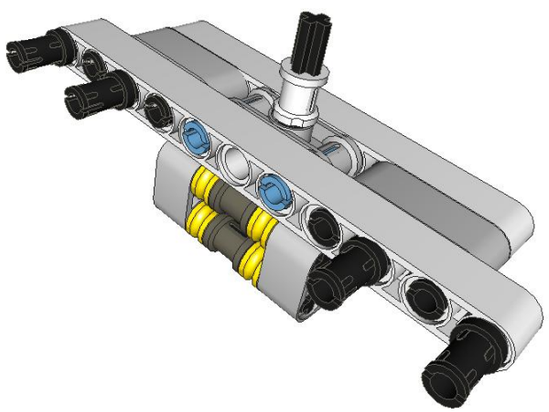
17.11



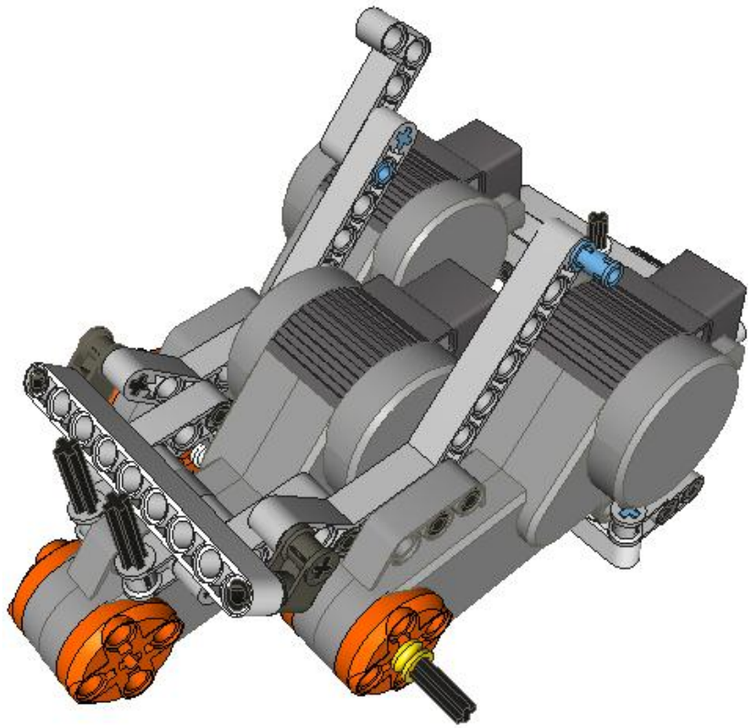
17.12



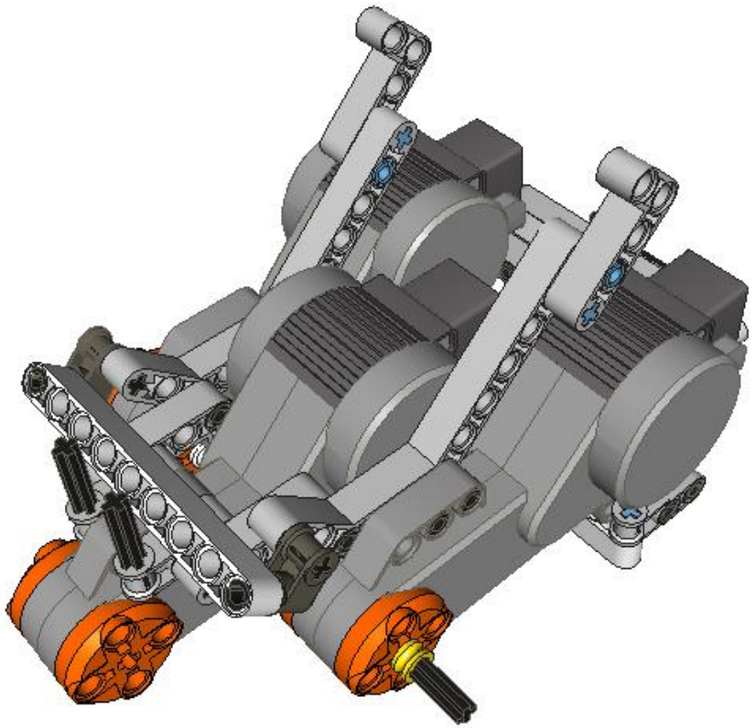
17.13



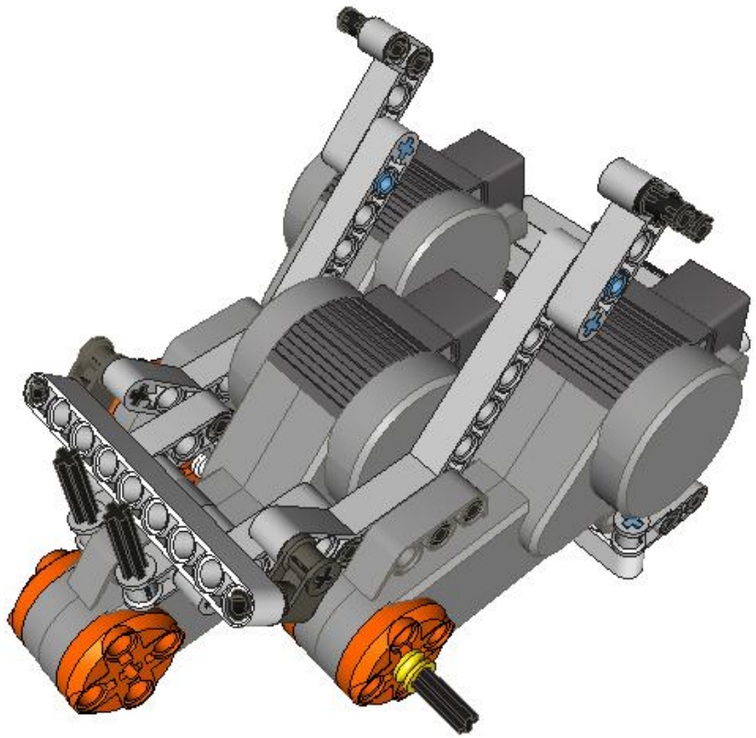
17.14



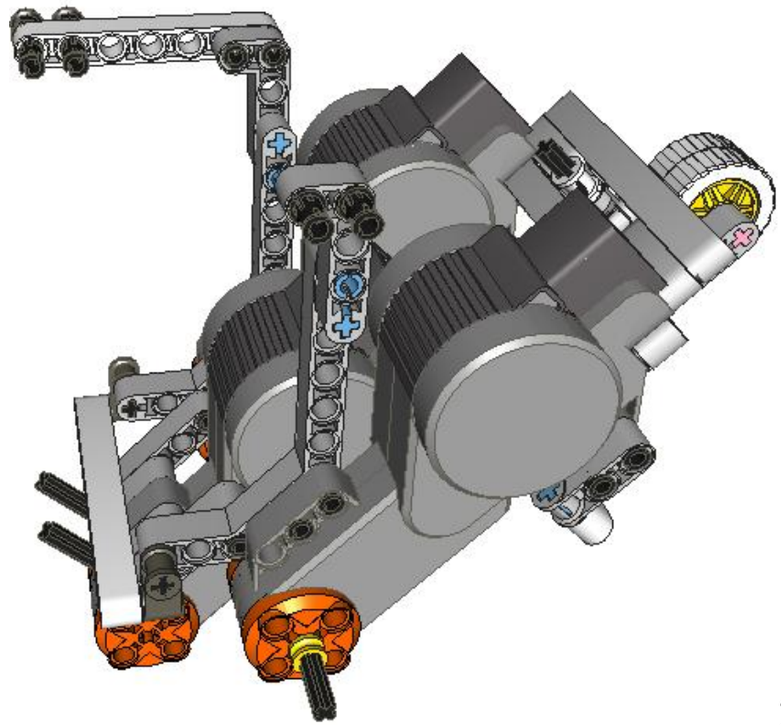
18



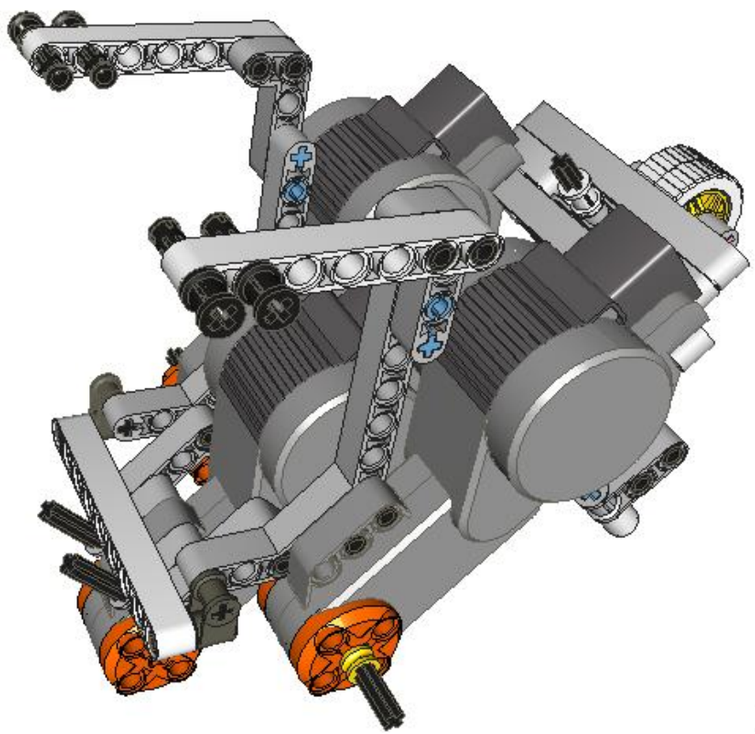
19



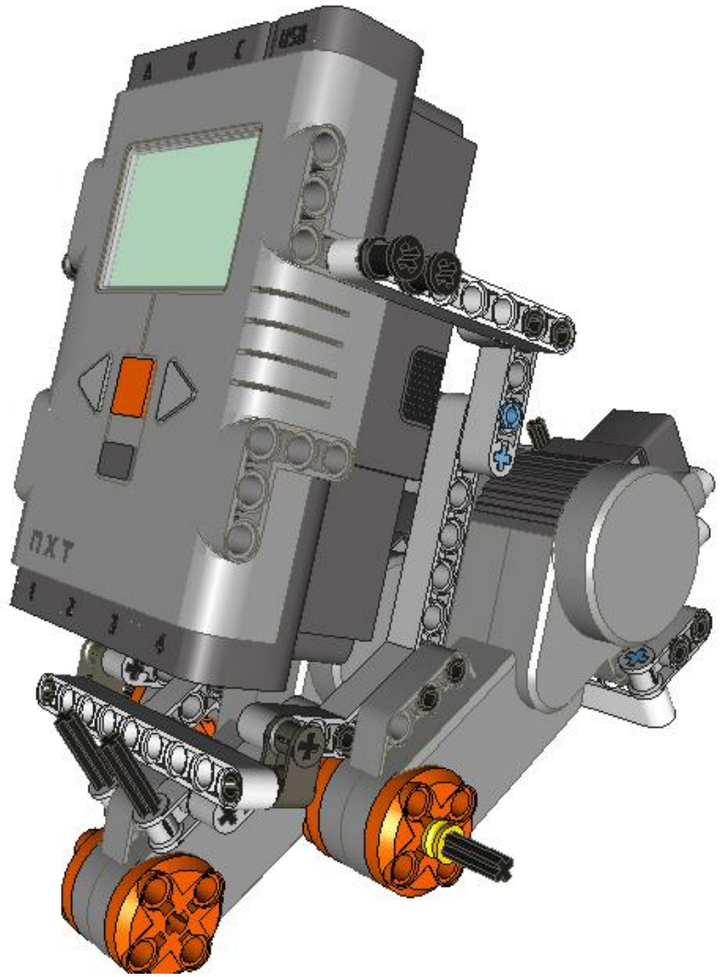
20



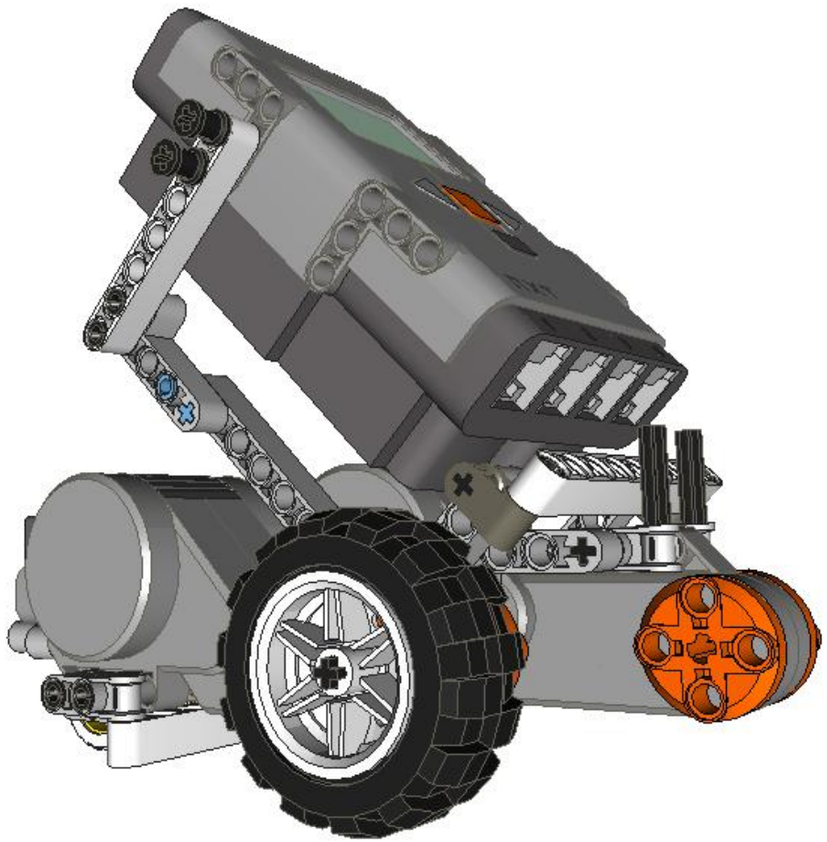
21



22



23



24

25

LISTA DE PIEZAS DEL DISEÑO BASE			
PASOS	CANTIDAD	PIEZA	DESCRIPCION
1	1	3706.DAT	Technic Axle 6
	1	3713.DAT	Technic Bush
2	1	53787.dat	Electric Mindstorms NXT Motor (Complete)
	1	32123.dat	Technic Bush 1/2 Smooth
3	4	4459.DAT	Technic Pin with Friction
4	1	42003.dat	Technic Axle Joiner Perpendicular with 2 Holes
5	2	4459.DAT	Technic Pin with Friction
6	1	32009.dat	Technic Beam 11.5 Bent 45 Double (7:3.828:3)
	1	4459.DAT	Technic Pin with Friction
7	1	32073.DAT	Technic Axle 5
	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
8	1	Submodelo_1	MOTOR_2
8.1	1	53787.dat	Electric Mindstorms NXT Motor (Complete)
	4	4459.DAT	Technic Pin with Friction
8.2	2	32524.dat	Technic Beam 7
8.3	1	6558.dat	Technic Pin Long with Friction and slot
8.4	2	42003.dat	Technic Axle Joiner Perpendicular with 2 Holes
9	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
10	1	Submodelo_2	MOTOR_3
10.1	1	53787.dat	Electric Mindstorms NXT Motor (Complete)
	1	3706.DAT	Technic Axle 6
	1	3713.DAT	Technic Bush
	1	32123.dat	Technic Bush 1/2 Smooth
	4	4459.DAT	Technic Pin with Friction
10.2	1	42003.dat	Technic Axle Joiner Perpendicular with 2 Holes
10.3	2	4459.DAT	Technic Pin with Friction
10.4	1	32009.dat	Technic Beam 11.5 Bent 45 Double (7:3.828:3)
	1	4459.DAT	Technic Pin with Friction
11	2	32062.DAT	Technic Axle 2 Notched
	2	43093.dat	Technic Axle Pin with Friction
12	2	6536.DAT	Technic Axle Joiner Perpendicular
	2	4459.DAT	Technic Pin with Friction
13	1	40490.dat	Technic Beam 9
14	2	4519.DAT	Technic Axle 3
15	2	43093.dat	Technic Axle Pin with Friction
16	1	41239.dat	Technic Beam 13
17	1	Submodelo_3	RUEDA_LOCA
17.1	2	3705.DAT	Technic Axle 4
	2	32140.dat	Technic Beam 5 Bent 90 (4:2)
17.2	2	32123.dat	Technic Bush 1/2 Smooth

17.3	2	6536.DAT	Technic Axle Joiner Perpendicular
17.4	1	3705.DAT	Technic Axle 4
17.5	1	30648.dat	Tyre 24 x 14 with Shallow Staggere
	1	55981.dat	Wheel 3 0.4 x 14 with Holes on Both Sides (Needs work)
17.6	2	32123.dat	Technic Bush 1/2 Smooth
17.7	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
17.8	1	32073.DAT	Technic Axle 6
17.9	1	32184.DAT	Technic Axle Joiner Perpendicular 3L
	1	3713.DAT	Technic Bush
17.10	2	43093.dat	Technic Axle Pin with Friction
17.11	1	32525.dat	Technic Beam 11
17.12	2	32523.dat	Technic Beam 3
	4	6558.dat	Technic Pin Long with Friction and Slot
17.13	1	40490.dat	Technic Beam 9
17.14	4	4459.DAT	Technic Pin with Friction
18	1	43093.dat	Technic Axle Pin with Friction
	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
19	1	43093.dat	Technic Axle Pin with Friction
	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
20	4	2780.DAT	Technic Pin with Friction and Slots
21	1	32524.dat	Technic Beam 7
	2	32054.DAT	Technic Pin Long with Stop Bush
22	1	32524.dat	Technic Beam 7
	2	32054.DAT	Technic Pin Long with Stop Bush
23	1	53788.dat	Electric Mindstorms NXT (Complete Shortcut)
24	1	6581.DAT	Tyre 20 x 30 Balloon Medium
	1	56145.dat	Wheel 43.2 x 22 without Pinholes, with External Ribs
25	1	6581.DAT	Tyre 20 x 30 Balloon Medium
	1	56145.dat	Wheel 43.2 x 22 without Pinholes, with External Ribs

5. Tipos de robot según su tarea.

Los robots realizados en esta guía son los siguientes:

5.1. Robot seguidor de pared.

Este Robot será capaz de seguir las paredes sin tocarlas, manteniéndose separado de la pared a una distancia de 20 cm. El entorno debe estar libre de obstáculos.

Para realizar el control de la distancia se ha propuesto un controlador difuso.

Nota: más adelante puede realizarse una versión donde por medio de otros sensores el robot pueda ser colocado en cualquier parte del entorno y este sea capaz de encontrar la pared y seguirla

5.1.1. Sensores.

Para este Robot se usaran tres sensores ultrasónicos de lego para medir distancias.

Dos estarán a los lados del robot y uno de frente, como lo muestra la figura siguiente.

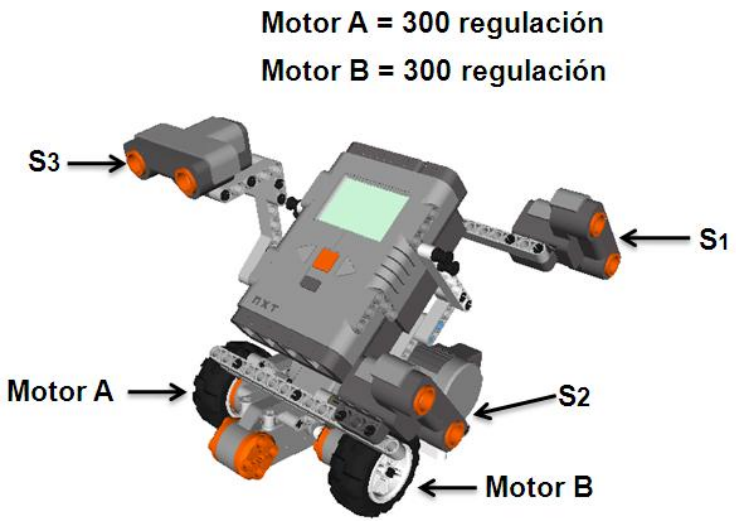


Figura 116.Robot seguidor de pared

Nota: el rango de estos sensores va desde 5-127 (Cm) se debe tener en cuenta a la hora de la programación que los rangos ingresados por el usuario no pueden exceder estos.

5.1.2. Principios de control.

Se busca controlar la distancia del Robot hacia una pared; para realizar esto se requiere controlar la velocidad de los motores independientemente. En caso de que se requiera que el robot se acerque a la pared, la velocidad del motor A será mayor que la del motor B, si se quiere que el Robot se aleje de la pared, la velocidad del motor B será mayor a la del motor A.

Para determinar la distancia y la inclinación que tiene el robot hacia la pared se utilizan dos sensores ultrasónicos colocados equidistantes con respecto al eje “x” y a diferentes distancias respecto al eje “y”. En la graficas siguientes se puede apreciar las diferentes situaciones generales a las cuales puede estar expuesto el Robot.

Posición uno: el Robot se encuentra acercándose a la pared, la distancia medida por el sensor 2 será menor a la del sensor 1.

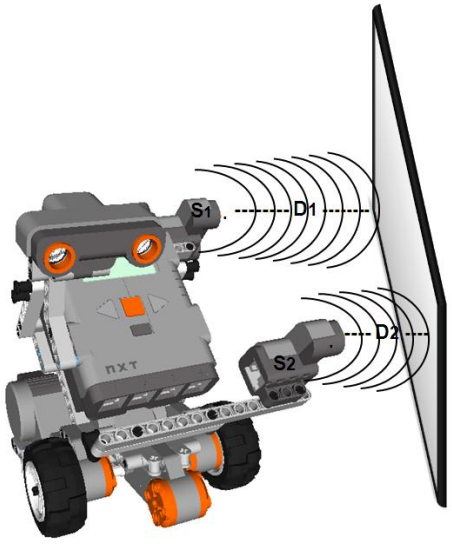


Figura 117.Posición uno

En esta situación el “motor B” debe andar más rápido que el “motor A” para que este no se choque con la pared.

Posición dos: el robot se encuentra alejándose de la pared; la distancia medida por el sensor 1 será menor a la del sensor 2.

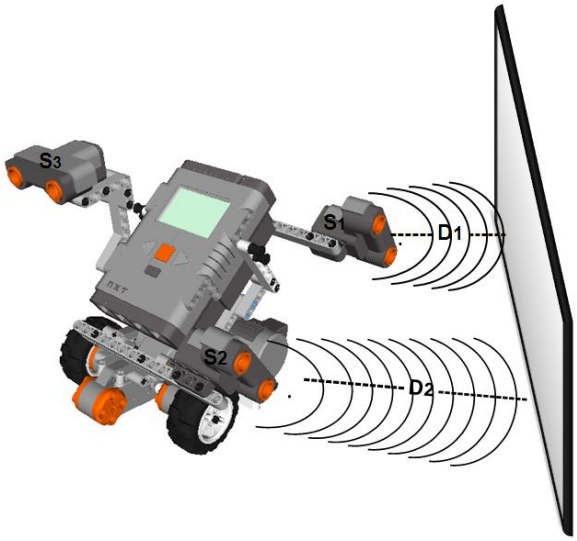


Figura 118.Posición dos

En esta situación el Robot se estará alejando de la pared, por lo que el “motor A” deberá ir más rápido que el “motor B”.

Posición tres: la distancia medida por ambos sensores será igual.

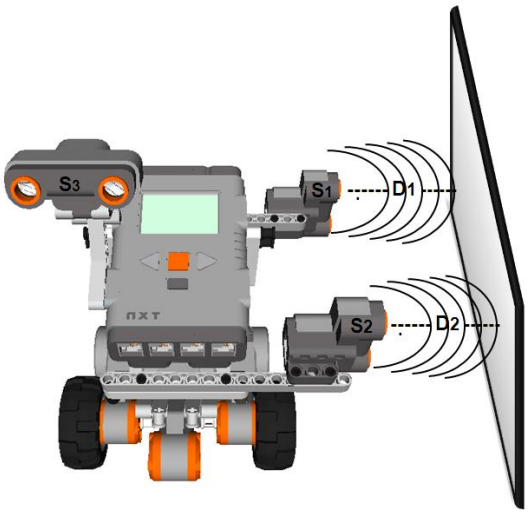


Figura 119.Posición tres

En esta situación, el Robot estará alineado y se acercara o alejara de la pared dependiendo de si está o no a la distancia set point.

5.1.3. Valores de entrada.

El robot tiene tres sensores, pero los valores de entrada al controlador difuso solo son dos (S1, S2, mirar figura 83), La medida tomada por cada uno de los dos sensores es usada para calcular el error desviación del valor medido con respecto al valor de set point establecido (en este caso es de 20cm).

Er= Vmed ─ Vset

Este error se calcula para cada uno de los dos sensores, obteniéndose dos diferentes medidas de error de distancia. Además al tener dos sensores ubicados equidistantes con respecto al eje “y”, y a diferentes posiciones del eje “x” se obtiene información de cómo es la posición del robot con respecto a la pared.

La función del tercer sensor, es evitar que choque con los obstáculos frente a él.

En cuanto al control de la velocidad la salida es la variable regulación, la cual modifica un valor constante de velocidad de cada motor, influyendo de esta forma de manera inversamente proporcional a cada motor, se ve representada la siguiente fórmula:

Motor A= 300 + regulación

Motor B= 300 ─ regulación

Donde el motor A y motor B constituyen la velocidad final que tendrá cada motor respectivamente. Se puede observar según la formulas que a medida que la variable “regulación” tienda a ser positiva, la velocidad del motor A será cada vez más grande con respecto al “motor B”, y si se esta se hace negativa “regulación”, la velocidad del “motor B” será cada vez mayor que la del “motor A”, con esto se

logra controlar la dirección del robot. Por ejemplo: Si se quiere que el motor gire a su izquierda, se hace que el “motor B” vaya más lento que el “motor A”.

Entonces tenemos tres variables. , error de distancia (medido con el sensor 1), error de distancia2 (medida del sensor dos), y regulación. De las cuales son dos variables de entrada y una variable de salida (regulación).

5.1.4. Programa experimental para obtención de valores óptimos de regulación.

El objetivo del programa es poder realizar pruebas con el Robot a diferentes distancias, velocidades y valores de regulación, con el fin de observar como era el comportamiento de este, para adquirir la base de conocimientos de la lógica difusa. La cual es la representación lógica, del comportamiento del sistema a controlar, obtenidas a través de experimentos y pruebas previas. Lo importante de esta base de conocimientos, es la facilidad con que se puede realizar modificaciones y mejoras al sistema, basándose en lo observado del comportamiento del controlador. Por ejemplo. En un programa de control difuso para regular la temperatura en un cuarto, se requiere controlar la cantidad de energía necesaria suministrada por el refrigerador para mantener el cuarto a cierta temperatura, con el fin de obtener una regulación optima de la temperatura con un bajo de consumo de energía eléctrica. Se obtuvo en pruebas preliminares que para una temperatura de 50 °C, y un flujo de 5 l/m el motor del refrigerador tiene que incrementar su velocidad a 200 RPM para suministrar la energía suficiente, y poder bajar la temperatura en el cuarto hasta la deseada. Pero al momento de implementar el programa se descubre que se estaba desperdiciando energía, porque con simplemente incrementar la velocidad a 100 RPM se consigue bajar la temperatura durante un tiempo un poco mayor, pero que para esta aplicación no pasa a ser crítico, y que resulta en un ahorro de energía eléctrica. Entonces simplemente se realiza una modificación a la estructura lógica del programa, puesto que la base de conocimiento ha sido actualizada. La facilidad con esto, es poder hacer cambios que se adapten a las variaciones de las condiciones estudiadas en un principio en el programa, y hacer programas a medida para cada aplicación.

Cabe resaltar que con el programa experimental para la obtención de los valores y con el programa del control de distancia, se puede posteriormente realizar experimentos a diferentes condiciones de proceso con el fin de obtener un programa de control más fino.

5.1.5. Funcionamiento del programa experimental.

El programa fue creado de tal forma que al ejecutarse, el usuario pueda modificar los valores de la variable “regulación” (observar en el ítem 5.1.3 la función de esta variable) con el fin de poder someter el robot a diferentes valores de regulación. Luego de introducir el valor de la variable regulación en la pantalla LCD del ladrillo, se muestra la distancia de ambos sensores, con el fin de colocar el robot en diferentes combinaciones de posiciones y distancia con respecto a la pared y observar el registro de las lecturas de los sensores.

5.1.6. Programa.

Al ejecutar el programa, se presiona cualquier botón para continuar, luego.

Con el botón izquierdo se controla si la regulación será negativa o positiva.

Si es cero la regulación es negativa, si es 1 será positiva.

Con el botón derecho se podrá incrementar el valor de regulación, que se va mostrando en pantalla en la parte derecha de esta, y el signo será mostrado en la izquierda, si es 1 quiere decir que el valor será negativo, cero si será positivo luego de haber seleccionado el valor de regulación a trabajar, se presiona el botón escape, para finalizar el ingreso de esta variable.

Ahora en pantalla sale un valor de distancia para nosotros colocar el robot a la distancia que queremos que inicie su movimiento.

Luego de haber seleccionado la distancia a la cual queremos evaluar el comportamiento del robot, presionamos el botón Enter, para indicar que finalizó la calibración de distancia.

Ahora presionamos Escape y el Robot se moverá hacia delante, mostrándonos en pantalla valores medidos de distancia con intervalos de 200 ms. Estarán ordenados en 4 columnas de 8 filas dando 32 valores de medidas tomados. Estos irán desde la columna izquierda de la pantalla LCD hasta la derecha. Luego de terminar los valores medidos el robot se detiene, permitiendo observar los resultados.

Si queremos volver a tomar otra medida con el mismo valor de regulación indicado en un principio, presionamos cualquier botón, y ahora en pantalla nos aparecerá nuevamente la opción de colocar una nueva regulación, y a otro valor de distancia, repetimos el paso donde se explica que hacer para calibración e distancia y listo.

5.1.7. Tabla de datos.

Los siguientes datos fueron obtenidos ejecutando pruebas con el Robot, a diferentes distancias y diferentes valores de regulación.

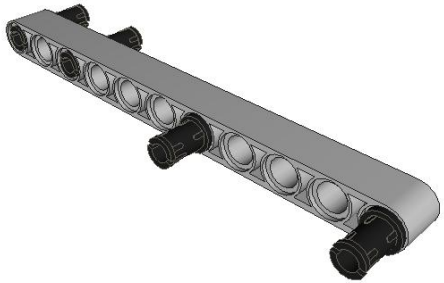
Distancia	Max	Min	Regulación
60	60	37	10
60	61	42	10
50	50	36	10
50	50	31	10
40	40	29	8
40	40	29	8
30	30	20	5
30	31	23	5
25	25	24	3
25	25	21	4
30	30	19	6
30	29	23	6
60	X	X	15
60	X	X	15

Tabla 29. Datos obtenidos mediante pruebas realizadas al Robot

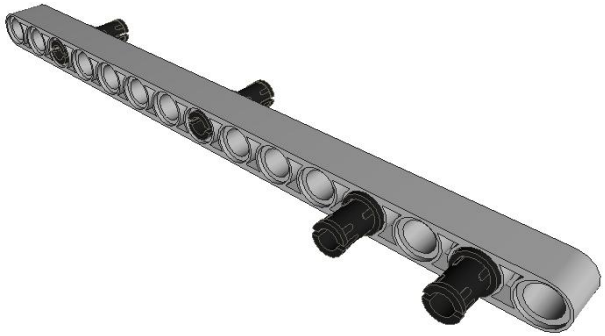
En el valor de distancia 60 a regulación de 15, no tiene datos máximos ni mínimos de distancia, debido a que el comportamiento era muy inestable, el Robot se aproximaba demasiado a la pared, saliéndose del rango de medida del sensor.

Por lo que basándose en los valores obtenidos en el programa experimental se descubrió que el rango de valores de regulación es [-10,10].

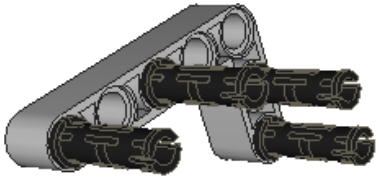
5.1.8. Guía de armado del robot seguidor de pared.



1

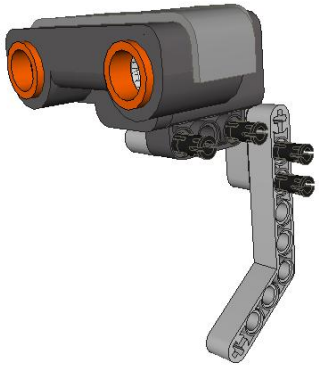


2

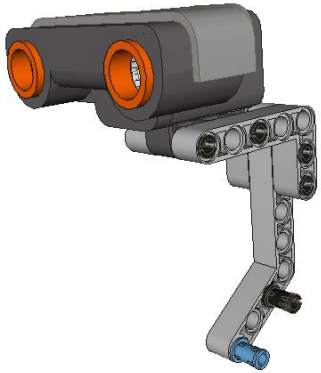


3

4

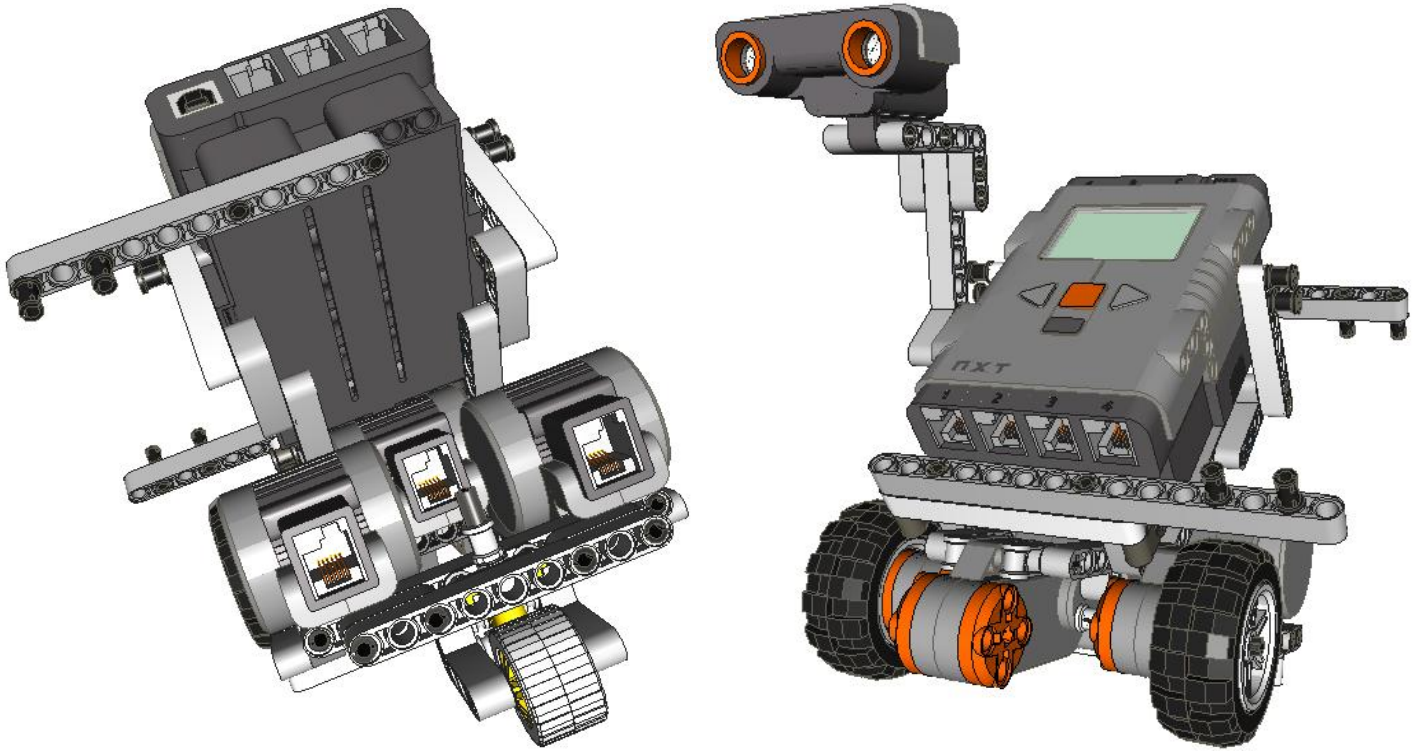


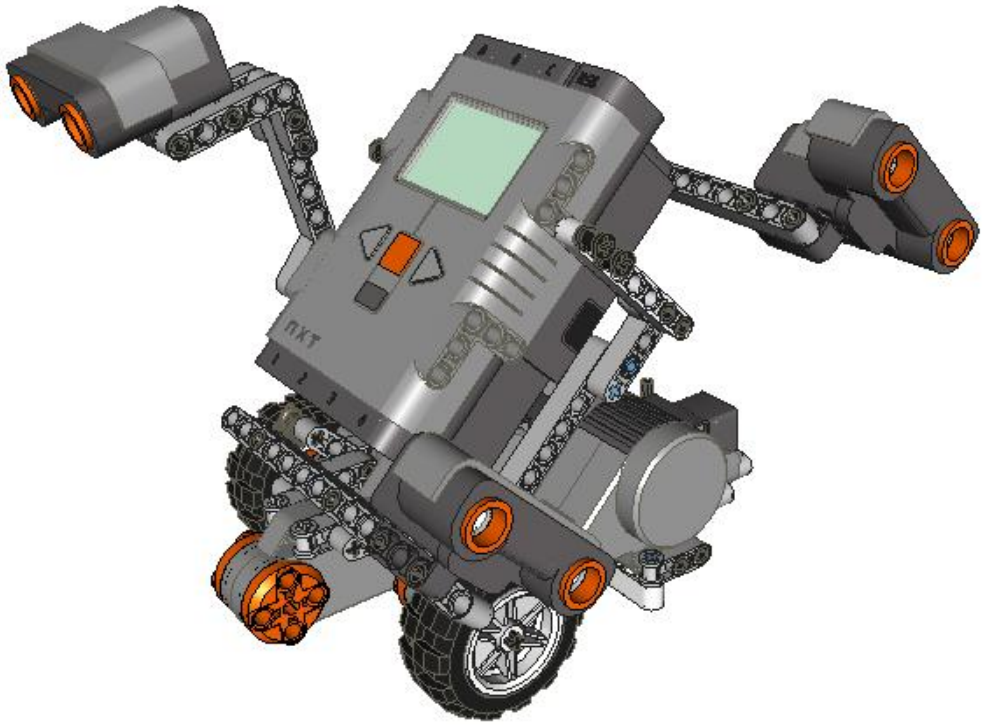
5



6

7


NXT



8

LISTA DE PIEZAS SEGUIDOR DE PARED			
PASOS	CANTIDAD	PIEZA	DESCRIPCION
1	1	32525.dat	Technic Beam 11
	4	4459.DAT	Technic Pin with Friction
2	1	32278.dat	Technic Beam 15
	4	4459.DAT	Technic Pin with Friction
3	1	32526.dat	Technic Beam 7 Bent 90 (5:3)
	4	6558.dat	Technic Pin Long with Friction and slot
4	1	6629.dat	Technic Beam 9 Bent 53.5 (6:4)
5	1	53792.dat	Electric Mindstorms NXT Ultrasonic Sensor (Complete)
6	1	43093.dat	Technic Axle Pin with Friction
	1	32526.dat	Technic Beam 7 Bent 90 (5:3)
	1	4459.DAT	Technic Pin with Friction
7	1	MODELO_DISEÑO_BASE	
8	2	53792.dat	Electric Mindstorms NXT Ultrasonic Sensor (Complete)

5.2. Robot seguidor de línea.

El robot es colocado en una superficie clara en la cual se encuentra trazado un camino por una franja negra, éste será capaz de seguir la línea negra; en el camino no debe haber ningún tipo de obstáculo. En caso que el robot se desoriente y pierde la línea debe ser orientado de vuelta al camino.

5.2.1. Sensores:

Figura 120.Robot Seguidor de línea

Se usan dos sensores de luz colocados enfrente de la estructura del robot en posición equidistante, orientados hacia abajo, de manera que detecte la superficie donde no haya línea negra; como se muestra en la figura 87.

5.2.2. Principios de control.

Se busca que el robot se mantenga siguiendo una línea negra, trazada en una superficie plana, blanca o de un color muy claro. Esto se logra colocando el robot en alguna parte del camino, según sea el caso el Robot tendrá un comportamiento específico; se pueden presentar las siguientes posibilidades:

Caso 1:

Los sensores no detectan la línea negra, el robot sigue hacia adelante con una velocidad determinada.

Caso 2:

El sensor de la derecha detecta la línea negra, el robot girará sobre su propio eje hacia la derecha, con una velocidad establecida, hasta que dicho sensor deje de detectar.

Caso 3:

El sensor de la izquierda detecta la línea negra, el robot girará sobre su propio eje hacia la izquierda, con una velocidad establecida previamente, hasta que dicho sensor deje de detectar.

Caso 4:

Los dos sensores detectan la línea negra, el robot se detendrá hasta que sea ubicado de vuelta al camino.

5.2.3. Valores de entrada.

Los valores de entrada que usa el robot son los valores de las lecturas de los dos sensores de luz, los cuales varían de acuerdo con la cantidad de luz. Los sensores se encuentran en modo reflectante de manera que cuando la luz del sensor incida sobre la superficie, ésta refleja una cantidad de luz dependiendo de la opacidad de dicha superficie; entre más opaca sea la superficie (valor de 0 para la más opaca) menor es el valor detectado por el sensor y viceversa en el caso de superficies menos opacas (valor de 100 para la más clara).

Los valores para la línea negra son menores a 33, esto se debe a la orientación y posición que tienen los sensores en el robot.

5.2.4. Funcionamiento del programa.

El programa comienza evaluando las dos entradas de los sensores de luz, luego entra en un bucle de control donde se define que acciones debe tomar el Robot, si seguir adelante, ir hacia la derecha, ir a la izquierda o detenerse; mientras esto sucede también se muestran en pantalla los valores de lectura de cada sensor.

5.2.5. Programa.

Se ubica el robot en el camino y se procede a correr el programa y se realizan las siguientes acciones:

24. Hacer la lectura de los dos sensores mediante el método read () y se almacena cada una en una variable:

```
public void read(){
        S1=luz1.readValue ();
        S2=luz2.readValue ();
}
```

25. Entra en el bucle de control:

```
if (S1>SPL && S2>SPL){
    pilot.setSpeed(200);
    pilot.forward ();
}else if (S1<SPL && S2>SPL){
    pilot.setSpeed (300);
    pilot.steer (110);
```

```
}else if (S1>SPL && S2<SPL){
        pilot.setSpeed (300);
        pilot.steer(-110);
        }else if (S1<SPL && S2<SPL){
        pilot.stop ();
}
```

Se ha establecido un valor set point previamente para el valor de la línea oscura, que es de 33, los valores menores a este set point representan el color oscuro o negro y los mayores el color claro.

En esta parte del código define cuál será la acción que debe tomar el robot de acuerdo a cuatro condiciones:

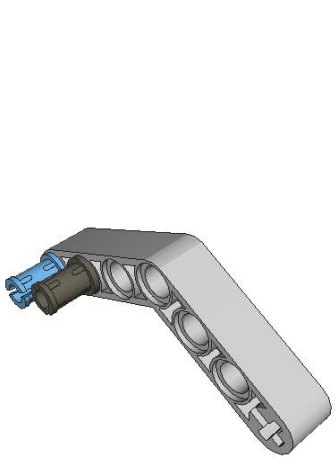
- Si los sensores no detectan línea negra el robot debe ir hacia adelante.
- Si el sensor derecho detecta línea negra el robot debe ir hacia la derecha, hasta que deje de detectar.
- Si el sensor izquierdo detecta línea negra el robot debe ir hacia la izquierda, hasta que deje de detectar.
- Y por último si los dos sensores detectan línea el robot debe detenerse.

26. Se muestran los valores de lectura en pantalla:

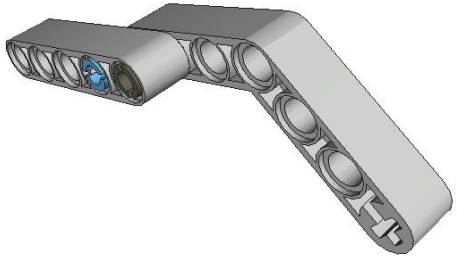
```
LCD.drawInt(sL.S1, 0, 0);
LCD.drawInt(sL.S2, 0, 1);
```

Por medio de estos comandos se muestra en la pantalla LCD los valores obtenidos de la lectura de los sensores.

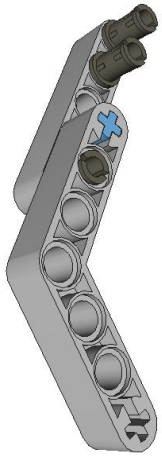
5.2.6. Guía de armado del robot seguidor de línea.



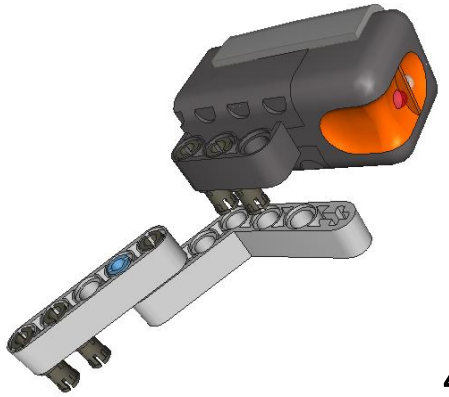
1



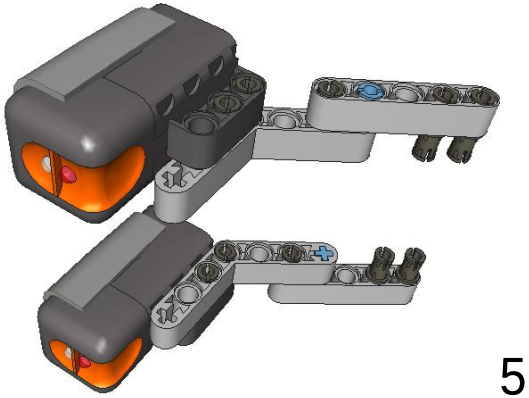
2



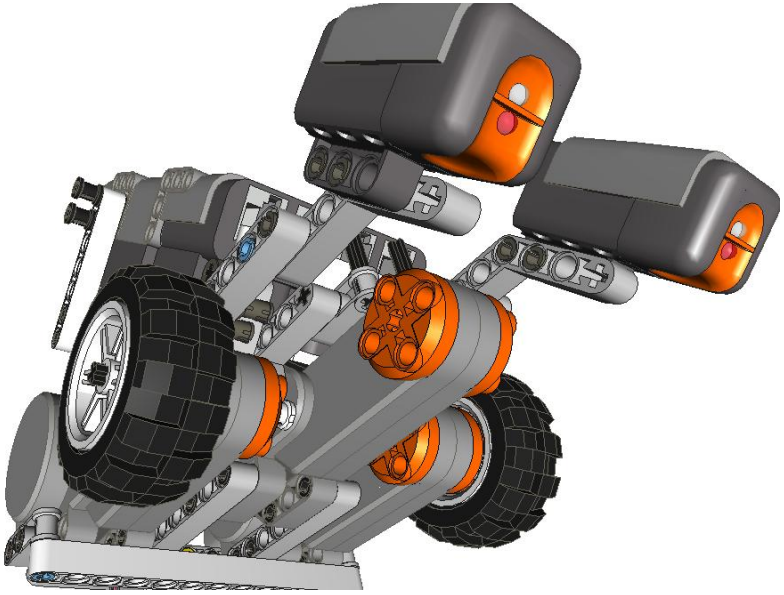
3



4



5



6

LISTA DE PIEZAS SEGUIDOR DE LINEA			
PASOS	CANTIDAD	PIEZA	DESCRIPCION
1	1	43093.dat	Technic Axle Pin with Friction
	1	32348.dat	Technic Beam 7 Bent 53.5 (4:4)
	1	4459.DAT	Technic Pin with Friction
2	1	32316.dat	Technic Beam 5
3	2	4459.DAT	Technic Pin with Friction
4	1	55969.dat	Electric Mindstorms NXT Light Sensor (Complete)
	2	4459.DAT	Technic Pin with Friction
5	1	43093.dat	Technic Axle Pin with Friction
	1	32348.dat	Technic Beam 7 Bent 53.5 (4:4)
	1	4459.DAT	Technic Pin with Friction
	1	32316.dat	Technic Beam 5
	2	4459.DAT	Technic Pin with Friction
	1	55969.dat	Electric Mindstorms NXT Light Sensor (Complete)
	2	4459.DAT	Technic Pin with Friction
6	1	MODELO_DISEÑO_BASE	

5.3. Robot seguidor de luz.

El robot es colocado en un espacio oscuro, se realiza una toma de lectura de la luz que éste va a seguir. El robot está programado para que cuando no haya fuente de luz de vueltas en un sentido predefinido, cuando encuentre una fuente de luz se dirigirá hacia ella.

5.3.1. Sensores.

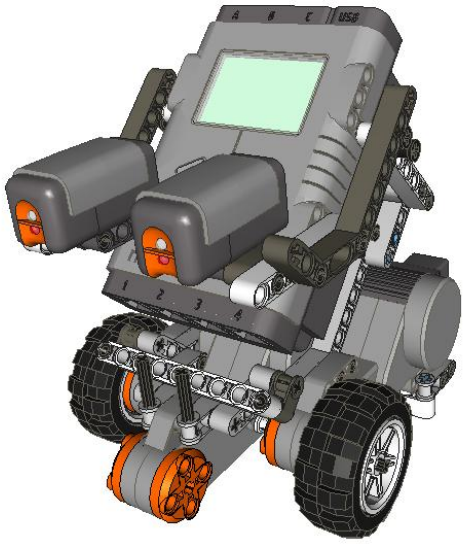


Figura 121.Robot Seguidor de luz

Se usan dos sensores de luz colocados enfrente de la estructura del robot en posición equidistante, orientados hacia enfrente, de manera que detecte la luz; como se muestra en la figura 88.

5.3.2. Principios de control.

Se busca que el robot sea capaz de detectar la luz que se ponga enfrente de él y la siga. Esto es posible colocando al robot en un ambiente con muy poca luz, permitiéndole detectar cambios en su entorno; así si se coloca una fuente de luz en sus cercanías va a ser capaz de detectarla y posteriormente seguirla. El robot tiene

3 comportamientos principales que ocurren dependiendo de las condiciones del ambiente:

Caso 1:

Los sensores no detectan una fuente de luz lo suficientemente fuerte para seguirla. Cuando esto sucede el robot da vueltas en un sentido predeterminado en la programación.

Caso 2:

Si el sensor de la derecha detecta más luz que el izquierdo, el robot girará a la derecha; si por el contrario el sensor de la izquierda detecta más luz que el de la derecha hacia la izquierda. En general, el robot girará hacía donde detecte más luz.

Caso 3:

Si tanto el sensor derecho como el sensor izquierdo detectan la misma cantidad de luz, el robot se dirigirá hacia adelante.

5.3.3. Valores de entrada.

Los valores de entrada que usa el robot son los valores de las lecturas de los dos sensores de luz, los cuales varían de acuerdo con la cantidad de luz. Los sensores se encuentran en modo no reflectante, de manera que detecte únicamente la luz externa.

Los valores de lectura de luz dependen en gran medida de la luz medida al principio del programa, la cual, se convierte en el set point de luz.

5.3.4. Funcionamiento del programa.

El programa comienza midiendo las dos entradas de los sensores de luz, luego entra en un bucle de control donde se define que acciones debe tomar el robot, si seguir adelante, ir hacia la derecha o ir a la izquierda; mientras esto sucede también se muestran en pantalla los valores de lectura de cada sensor.

5.3.5. Programa.

Se ubica al robot en un ambiente con poca luz, se procede a correr el programa y se realizan las siguientes acciones:

1. Se ejecuta el método seleccion(), donde se realiza la lectura de luz y se determinan los set points, alto y bajo, así:

```
public void seleccion(){
    while (!Button.RIGHT.isPressed()){
        LCD.drawString("Presione RIGHT", 0, 0);
        LCD.drawString (" para hacer ", 0, 1);
        LCD.drawString ("lectura de luz", 0, 2);
        SPLL1=luz1.readValue ();
        SPLL2=luz2.readValue ();

        if (SPLL1>SPLL2){
            SPLL=SPLL1;
        }else
            SPLL=SPLL2;
    }
    SPLH=SPLL*2.5;//El valor de 2.5 fue determinado por ensayo y error
    LCD.clearDisplay ();
    }
```

2. Entra en el bucle de control:

```
public void seguir(){
        leer();
        if ((S1>SPLL && S1<SPLH) && S1>S2){
            irDerecha();
        }else if ((S2>SPLL && S2<SPLH) && S2>S1){
            irIzquierda();
        }else if (S1>SPLH || S2>SPLH){
            irAdelante();
        }else
            buscar();
    }
```

Primero se realiza la lectura de los sensores por medio del sencillo método leer():

```
public void leer(){
            S1=luz1.readValue ();
            S2=luz2.readValue ();
    }
```

Luego de leer se entra en un bucle de if y else if anidados, donde se encuentran tres opciones:

- La primera es si la luz está entre el set point bajo y el set point alto, el robot debe ir hacia donde sea mayor el valor de luz.
- La segunda es que si la luz es la misma para los dos sensores y supera el valor alto de luz, el robot debe ir hacia adelante.
- Y la última es cuando ninguno de los dos sensores detecten luz en el rango establecido por SPLL y SPLH, el robot debe girar en un sentido predeterminado hasta encontrar luz nuevamente.

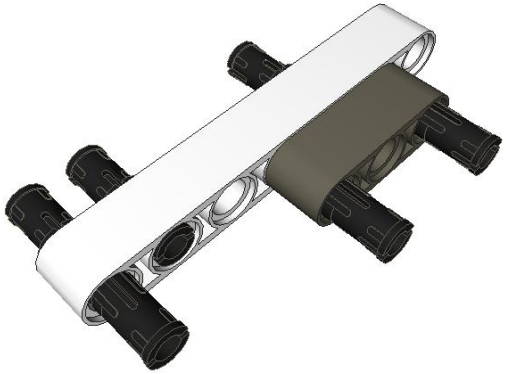
3. Se muestran los valores de lectura en pantalla:

```
LCD.drawString("SPLL "+sL2.SPLL, 0, 0);

    LCD.drawString ("SPLH "+sL2.SPLH, 0, 1);

    LCD.drawString ("L1 "+sL2.S1, 0, 3);

        LCD.drawString("L2 "+sL2.S2, 0, 4);

        LCD.refresh();
```

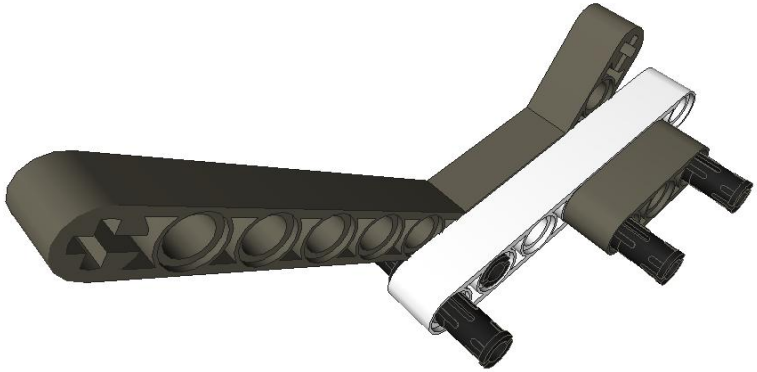
Por medio de estos comandos se muestra en la pantalla LCD los valores obtenidos de la lectura de los sensores.

5.3.6. Guía de armado del robot seguidor de línea.

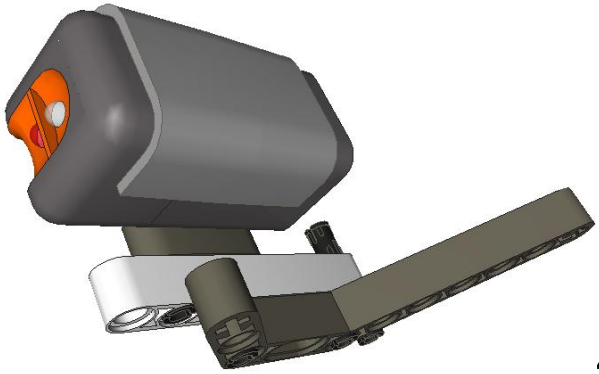
1



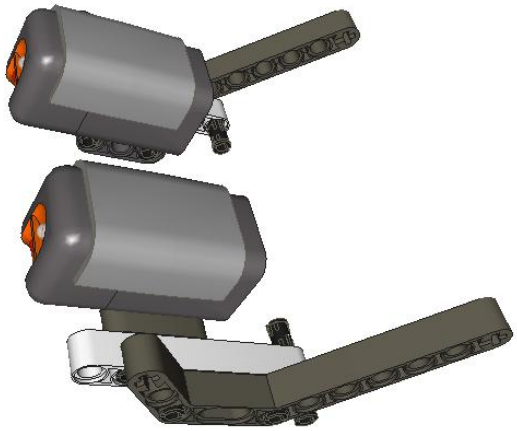
2



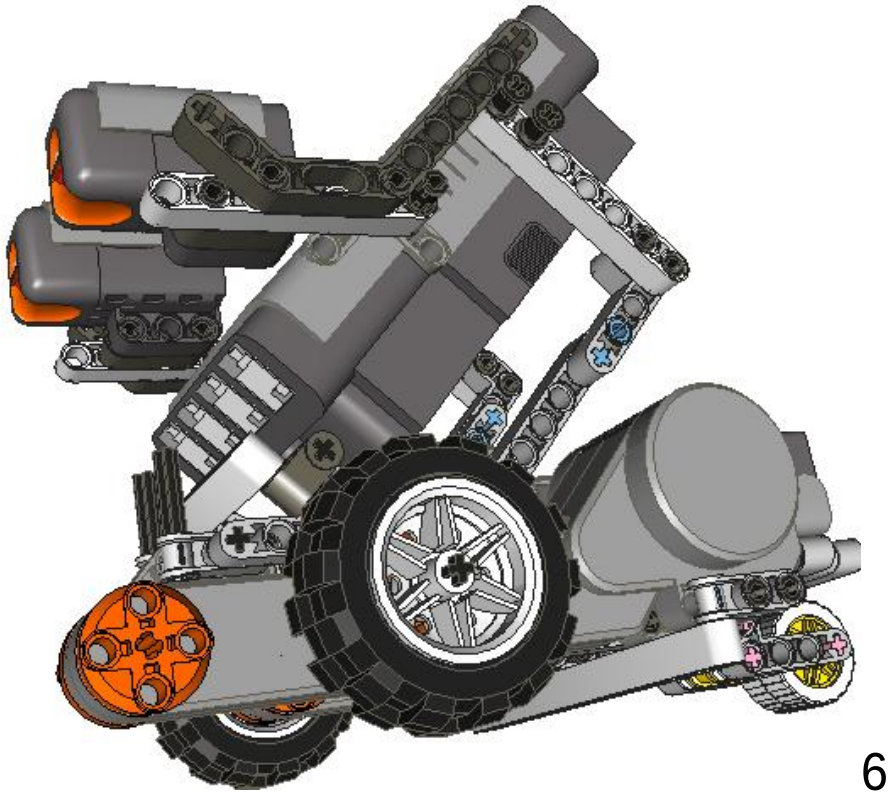
3



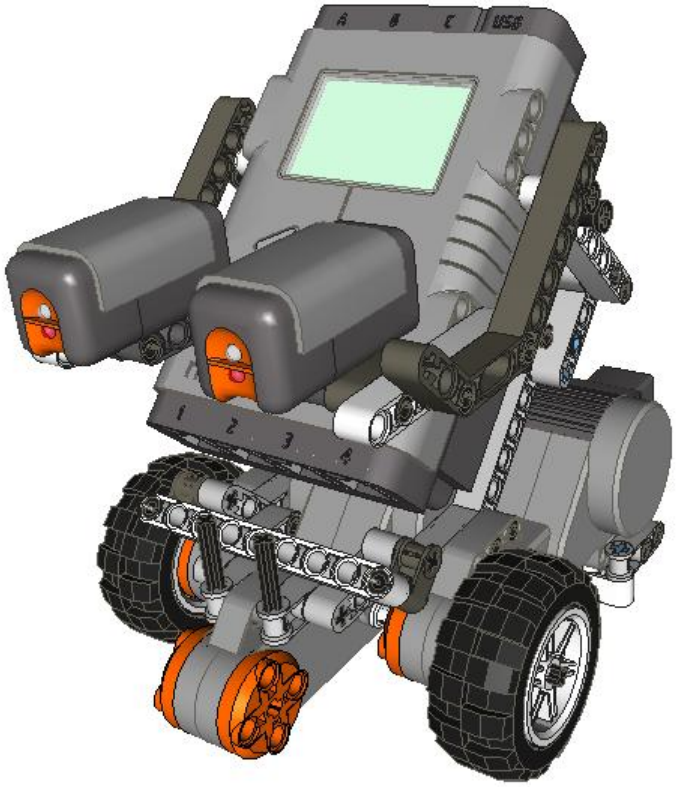
4



5



6



LISTA DE PIEZAS SEGUIDOR DE LUZ			
PASOS	CANTIDAD	PIEZA	DESCRIPCION
1	1	32524.dat	Technic Beam 7
	3	6558.dat	Technic Pin Long with Friction and slot
	2	2780.DAT	Technic Pin with Friction and Slots
2	1	32523.dat	Technic Beam 3
3	1	32009.dat	Technic Beam 11.5 Bent 45 Double (7:3.828:3)
4	1	55969.dat	Electric Mindstorms NXT Light Sensor (Complete)
5	1	55969.dat	Electric Mindstorms NXT Light Sensor (Complete)
	1	32009.dat	Technic Beam 11.5 Bent 45 Double (7:3.828:3)
	1	32523.dat	Technic Beam 7
	1	32524.dat	Technic Beam 3
	3	6558.dat	Technic Pin Long with Friction and slot
	2	2780.DAT	Technic Pin with Friction and Slots
6	1	MODELO_2.ldr MODELO_2	

5.4. Robot con pinza sujetadora.

Para el presente diseño el estudiante debe:

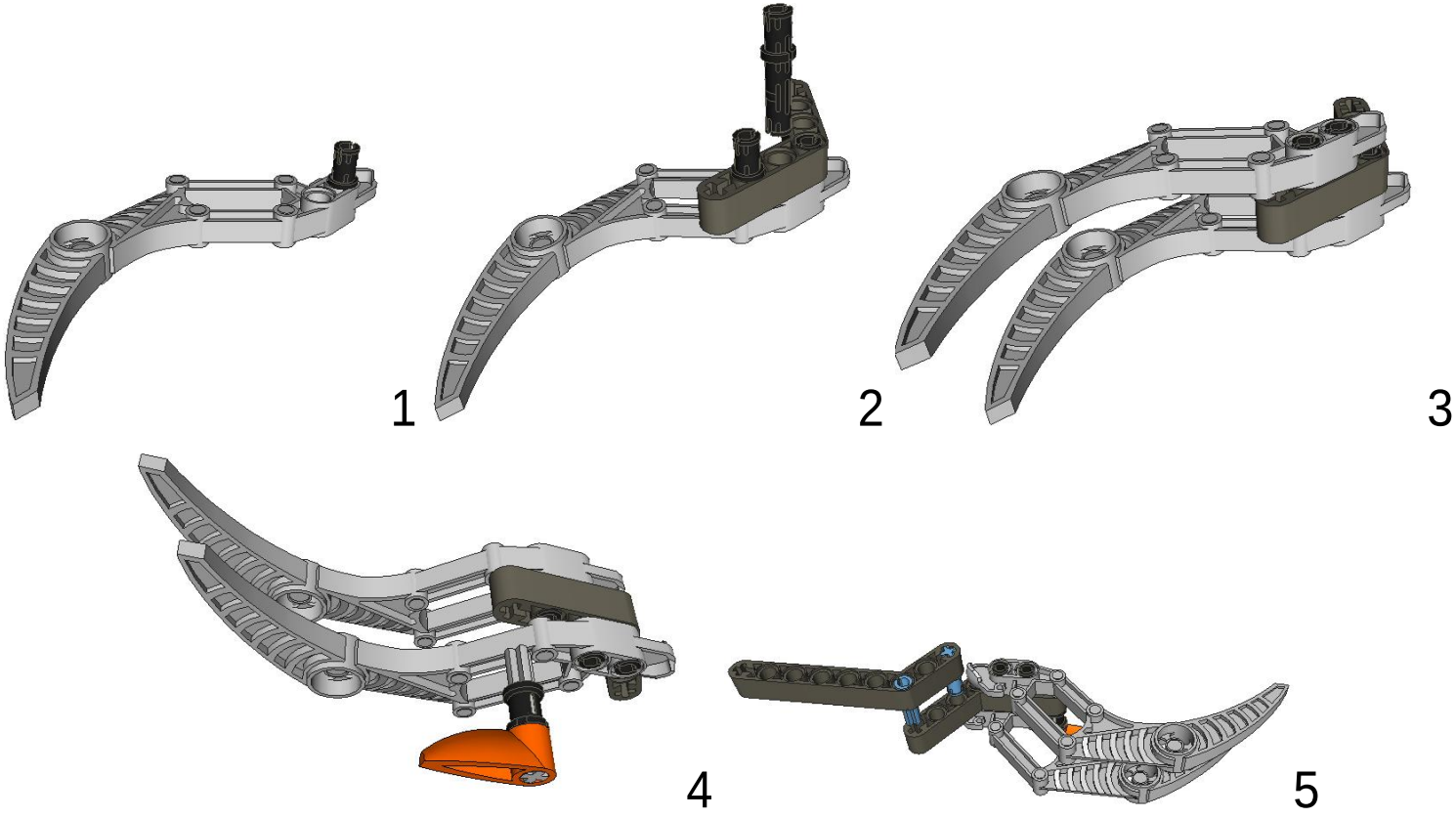
Observar el comportamiento del robot con el programa ejemplo, revisar el código del ejemplo, documentar el programa ejemplo y realizar un nuevo programa donde emplee los sensores de luz y ultrasonido. Para este nuevo programa documentarlo de tal manera que se explique la lógica del programa y el algoritmo.

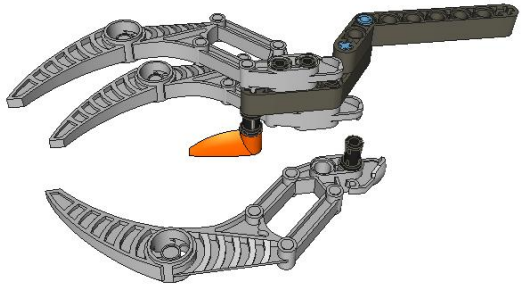
Programa ejemplo:

```
import lejos.nxt.Button;

import lejos.nxt.LCD;

import lejos.nxt.Motor;

import lejos.navigation.*;

/**

 * Programa que se utilizó para conocer la distancia en grados que se

 * requiere para que el robot agarre la bola con la pinza.

 * @author Charles

 * @version 0.1

 *

 */

public class PruebaAgarrarB {

        static TachoNavigator tachoNav;
```

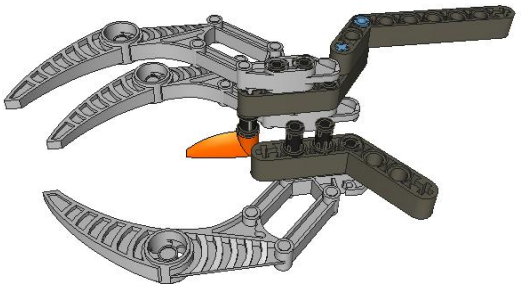
```
        public static void main(String[] args) throws Exception {
                tachoNav = new TachoNavigator(5.6f, 10.5f, Motor.A,
Motor.C);
                LCD.drawString("instrucciones \n"+"presiones boton", 0, 0);
                Thread.sleep(1000);
                LCD.drawString("Motor pinza pto B\n"+"presione boton", 0, 3);
                Motor.B.resetTachoCount();
                Motor.B.setSpeed(300);
                Thread.sleep(1000);
                        Motor.B.rotateTo (-117,true);
                        while (Motor.B.isMoving())Thread.sleep(500);
                        Motor.B.lock(100);
                        Thread.sleep(1000);
               tachoNav.setSpeed (500);
               tachoNav.setPosition (0, 0, 0);// heading along y axis
               tachoNav.goTo (-10,0);
               tachoNav.goTo (-10,25);
                        Motor.B.rotateTo (0);
              }
}
```

5.4.1. Guía de armado del robot con pinza sujetadora.

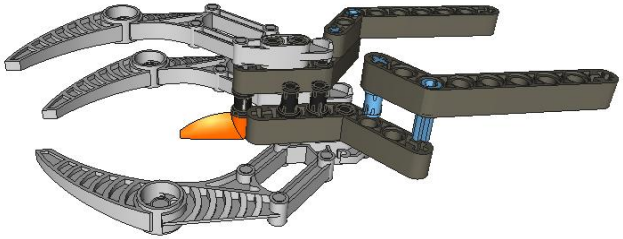




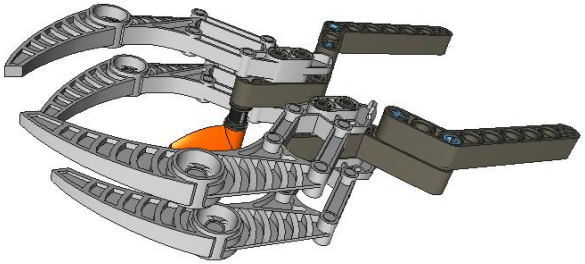
6



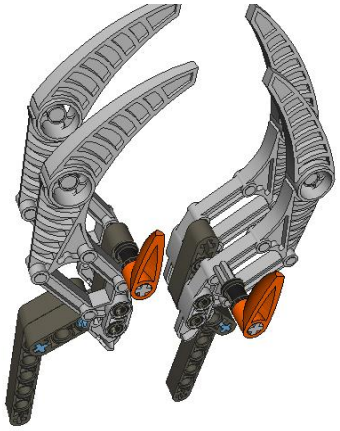
7



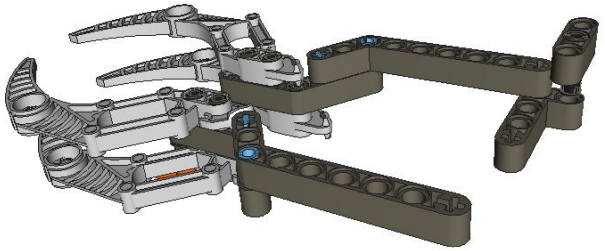
8



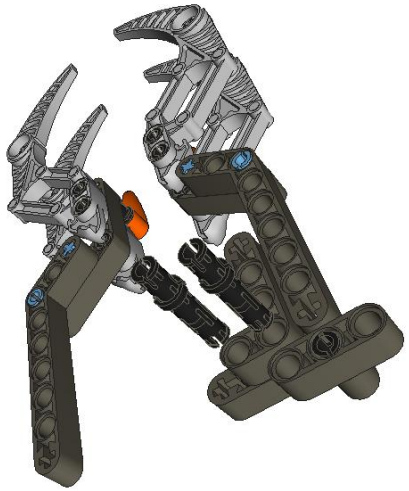
9



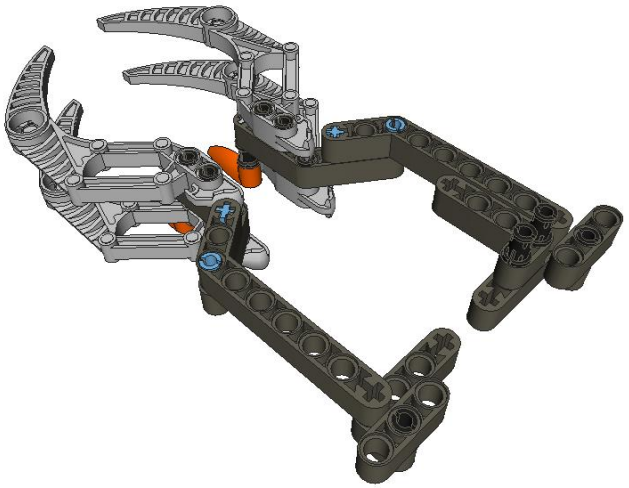
10



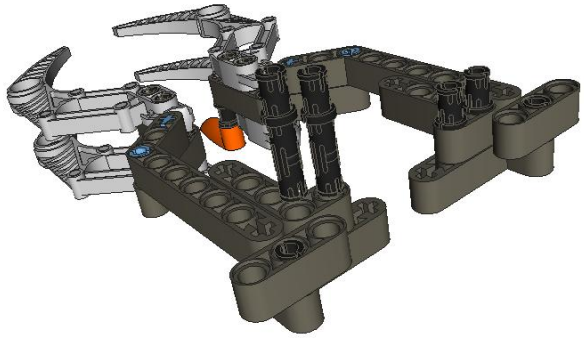
11



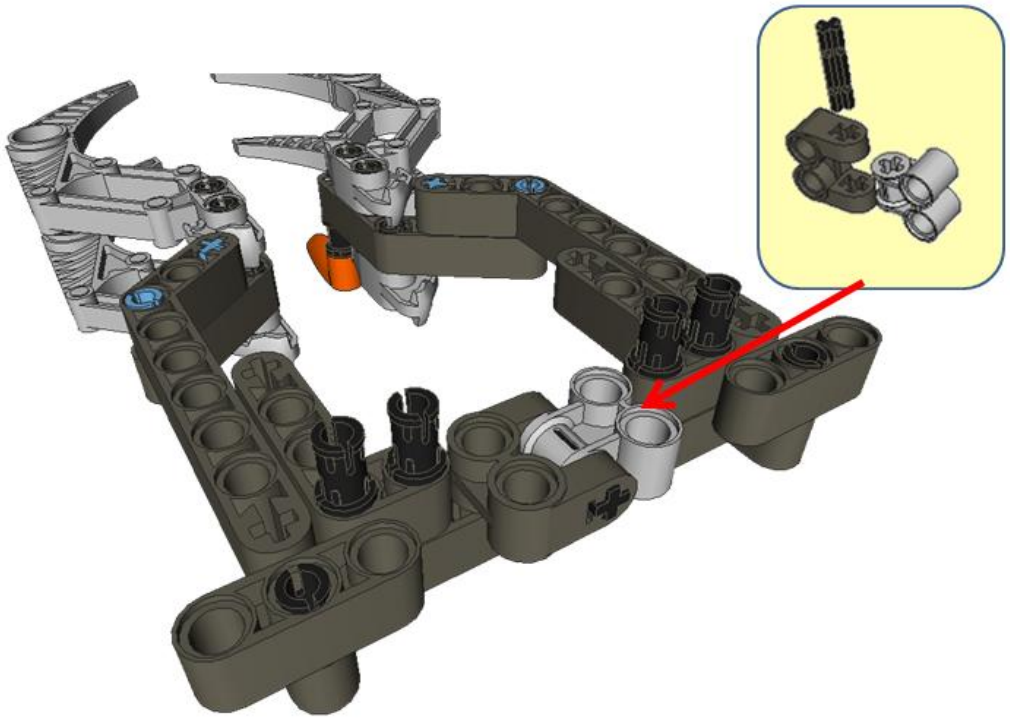
12



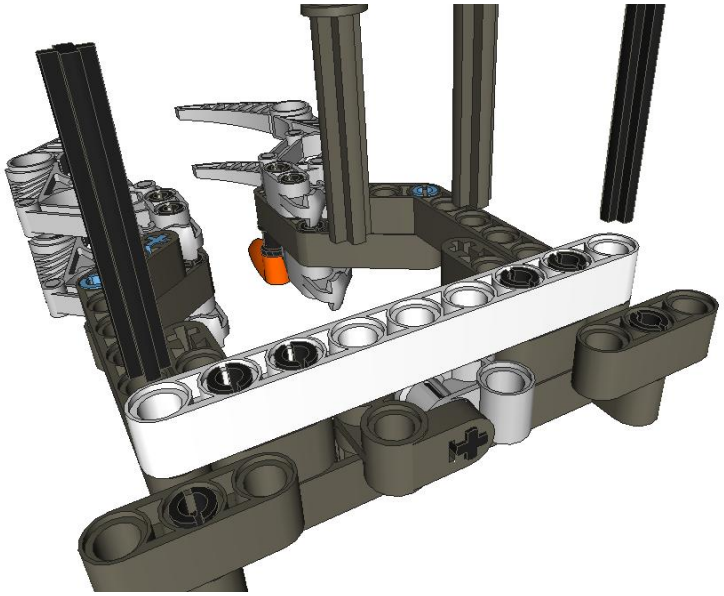
13



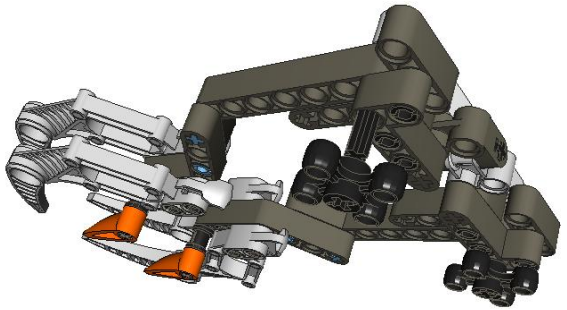
14



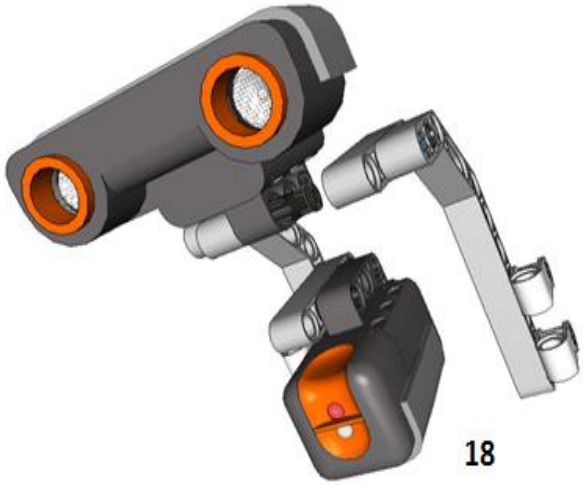
15



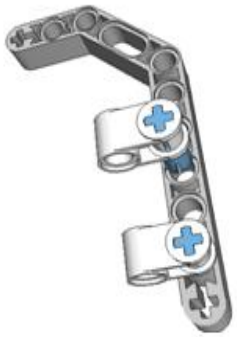
16



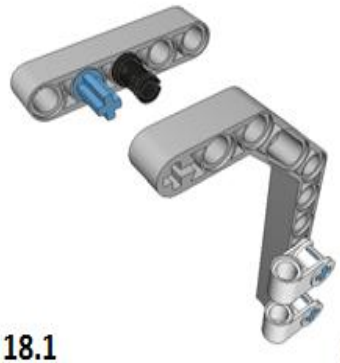
17



18

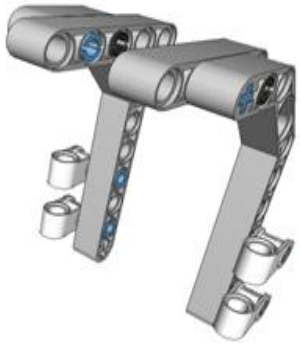


18.1

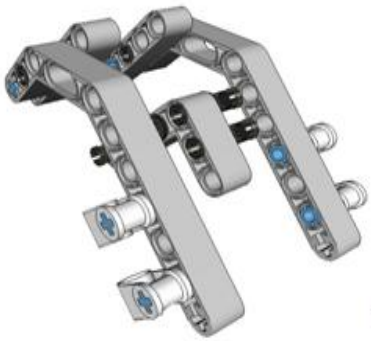


18.2

18.3



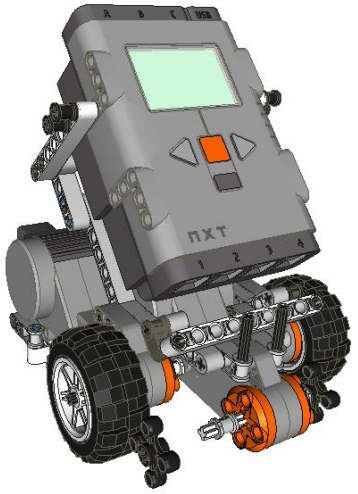
18.4



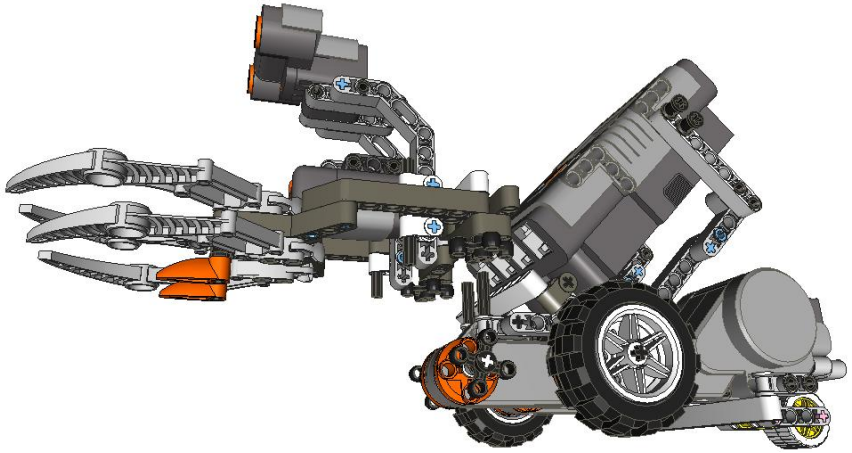
18.5

18.6

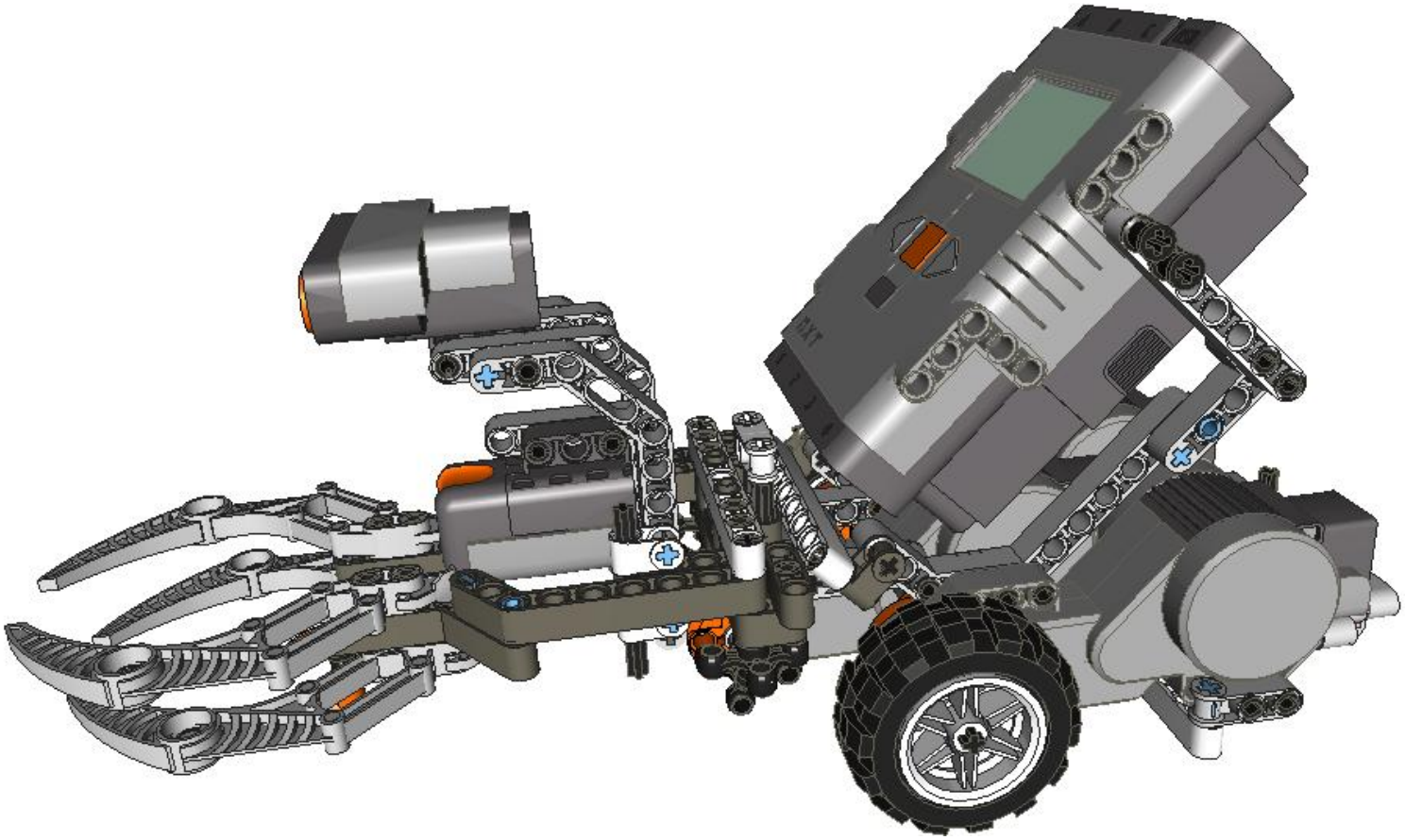
19



20



21



22

LISTA DE PIEZAS PINZA			
PASOS	CANTIDAD	PIEZA	DESCRIPCION
1	1	50914.dat	Technic Bionicle Weapon Pincer Suukorak
	1	4459.DAT	Technic Pin with Friction
2	1	32348.dat	Technic Beam 7 Bent 53.5 (4:4)
	1	6558.dat	Technic Pin Long with Friction and slot
	1	4459.DAT	Technic Pin with Friction
3	1	50914.dat	Technic Bionicle Weapon Pincer Suukorak
4	1	4519.DAT	Technic Axle 3
	1	41669.dat	Technic Bionicle 1 x 3 Tooth with Axlehole
	1	3713.DAT	Technic Bush
5	2	43093.dat	Technic Axle Pin with Friction
	1	32271.dat	Technic Beam 9 Bent 53.5 (7:3)
6	1	50914.dat	Technic Bionicle Weapon Pincer Suukorak
	1	4459.DAT	Technic Pin with Friction
7	1	32348.dat	Technic Beam 7 Bent 53.5 (4:4)
	1	6558.dat	Technic Pin Long with Friction and slot
	1	4459.DAT	Technic Pin with Friction
8	2	43093.dat	Technic Axle Pin with Friction
	1	32271.dat	Technic Beam 9 Bent 53.5 (7:3)
9	1	50914.dat	Technic Bionicle Weapon Pincer Suukorak
10	1	4519.DAT	Technic Axle 3
	1	41669.dat	Technic Bionicle 1 x 3 Tooth with Axlehole
	1	3713.DAT	Technic Bush
11	1	32523.dat	Technic Beam 3
	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
	1	4459.DAT	Technic Pin with Friction
12	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
	2	6558.dat	Technic Pin Long with Friction and slot
13	1	32523.dat	Technic Beam 3
	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
	1	4459.DAT	Technic Pin with Friction
14	1	32140.dat	Technic Beam 5 Bent 90 (4:2)
	2	6558.dat	Technic Pin Long with Friction and slot
15	1	32062.DAT	Technic Axle 2 Notched
	1	32291.DAT	Technic Axle Joiner Perpendicular Double
	1	41678.DAT	Technic Axle Joiner Perpendicular Double Split
16	2	6587.DAT	Technic Axle 3 with Stud
	2	3705.DAT	Technic Axle 4

	1	40490.dat	Technic Beam 9
17	2	32072.dat	Technic Knob Wheel
18	1	SUBMODELO 1	
18.1	2	6536.DAT	Technic Axle Joiner Perpendicular
	2	3749.DAT	Technic Axle Pin
	1	32009.dat	Technic Beam 11.5 Bent 45 Double (7:3.828:3)
18.2	1	3749.DAT	Technic Axle Pin
	1	32316.dat	Technic Beam 5
	1	4459.DAT	Technic Pin with Friction
18.3	2	6536.DAT	Technic Axle Joiner Perpendicular
	2	3749.DAT	Technic Axle Pin
	1	32009.dat	Technic Beam 11.5 Bent 45 Double (7:3.828:3)
18.4	1	43093.dat	Technic Axle Pin with Friction
	1	32316.dat	Technic Beam 5
	1	32526.dat	Technic Beam 7 Bent 90 (5:3)
18.5	5	4459.DAT	Technic Pin with Friction
	1	55969.dat	Electric Mindstorms NXT Light Sensor (Complete)
18.6	1	53792.dat	Electric Mindstorms NXT Ultrasonic Sensor (Complete)
	2	6558.dat	Technic Pin Long with Friction and slot
19	2	32073.DAT	Technic Axle 5
20	1	DISEÑO BASE	
	1	44294.DAT	Technic Axle 7
	2	3713.DAT	Technic Bush
	2	32072.dat	Technic Knob Wheel
21	1	32184.DAT	Technic Axle Joiner Perpendicular 3L

ANEXO B

Introducción a los tipos de control.

(Diseño de un control difuso con Fuzzy Logic Toolbox).

Cada día los seres humanos realizan miles de procesos de control, una simple tarea que a simple vista puede parecer sencilla, al desglosarla puede tener incluido un arduo procesos de razonamiento y control. Por ejemplo: montar bicicleta. Se requiere controlar el equilibrio, dar pedales, controlar la dirección, y controlar la velocidad. Todas estas tareas de control muchas veces son realizadas por el cerebro, de manera subconsciente.

Así que en general un proceso de control como su nombre lo indica, es supervisar un proceso que se requiere mantener bajo unas condiciones deseadas. Para realizar esto existen muchos métodos con los que se busca mantener estas condiciones. El control puede ser realizado con o sin la supervisión del hombre. En el primer caso, el hombre es el que se encarga de analizar si el proceso se encuentra en las condiciones deseadas, y realiza su intervención (cuando sea necesario) actuando en el proceso según su razonamiento y experiencia le indique, para mantener las condiciones. Cuando el control es de tipo automático, el proceso puede realizarse sin o con poca intervención del hombre, la función de supervisión en este caso es realizada por los sensores, y las decisiones son tomadas por el controlador quien es el cerebro de la maquina (CPU). Existen varias formas de implementar el control. Siendo las básicas, en lazo abierto y lazo cerrado; de estos dos sistemas de control se discutiera más adelante.

Para hablar de control automático se deben definir ciertos términos propios de éste, como son:

Realimentación: Retorno de parte de la salida de un circuito o sistema a su propia entrada.

Variable manipulada: es el elemento al cual se le modifica su magnitud, para lograr el resultado deseado.

Salida del sistema, variable controlada o variable del proceso: es el valor obtenido por el sistema, el cual se encuentra relacionado con el valor de entrada.

Set-point: es el valor que se desea alcanzar.

Offset: Es el error de estado estacionario, error acumulado o el error que aún permanece en el tiempo (Ogata, 1998).

1. Control de lazo abierto.

Los sistemas de control de lazo abiertos son aquellos en los que la acción de control no depende de la salida del sistema. Lo que quiere decir, que en un sistema de este tipo no hay realimentación y no se realiza comparación entre los valores de entrada y salida del mismo. Lo que implica que el sistema no será capaz de responder sin la intervención del hombre, a variaciones externas del proceso (Ogata, 1998).

2. Control de lazo cerrado.

En los sistemas de lazo cerrado o sistemas de control realimentados utilizan la medida de la salida del proceso (variable del proceso – VP) para compararla y hacerla coincidir con un valor deseado o también llamado set-point (SP). Para realizar esta comparación se realiza una resta entre el set-point y la variable de proceso $(SP - VP)$, el resultado de la operación es llamado Error (e). El principal objetivo de este sistema de control es reducir el error y llevar la salida a un valor conveniente.

2.1. Tipos de control de lazo cerrado

Entre los diferentes tipo de control de lazo cerrado se encuentran: el control ON-OFF, control PID y control difuso.

2.1.1. Control ON-OFF.

Este controlador solo tiene dos estados ON-OFF (Encendido/Apagado, Derecha/Izquierda, etc.). Como todo controlador de lazo cerrado, realiza una

comparación entre el set-point y la variable del proceso y se toma una de las dos opciones ON u OFF dependiendo del signo del error; el rango en el que debe moverse la señal de error antes de que se conmute entre ON y OFF se denomina brecha diferencial o GAP, esta brecha es se utiliza para evitar la una operación muy frecuente del mecanismo de encendido-apagado (ON-OFF).

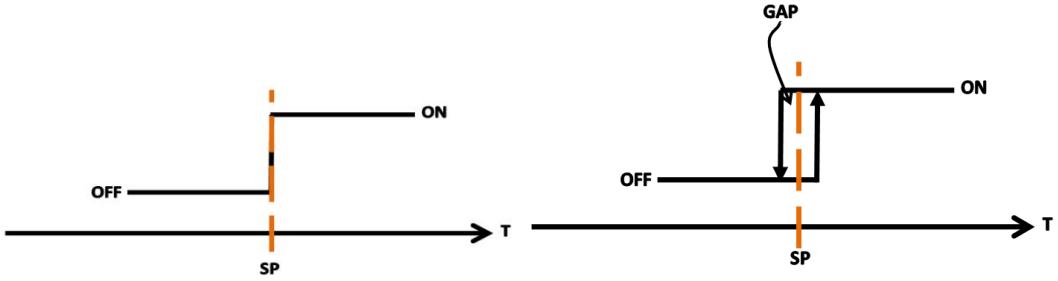


Figura 122.Control ON-OFF

2.1.2. Controlador PID.

Un controlador Proporcional-Integral-Derivativo (controlador PID) es un mecanismo genérico de control de lazo realimentado ampliamente usado en los sistemas de control industrial. Un controlador PID corrige el error entre la variable de proceso medida y la deseada de set-point, y realiza una acción correctiva sobre la salida, la cual será proporcional al error, con lo que se puede mantener las condiciones deseadas del proceso.

El controlador PID está compuesto por tres controladores o términos; el valor proporcional determina la reacción al error actual, el valor integral determina la acción basado en la suma de los errores recientes y el valor derivativo determina la reacción basado en la tasa de los errores que han estado cambiando.

La ecuación del PID está determinada por la siguiente ecuación:

$$MV(t) = P_{out} + I_{out} + D_{out}$$

Donde MV es la variable manipulada, P_{out}, I_{out} y D_{out} son las contribuciones (términos) del controlador PID a la salida. El controlador puede proporcionar la acción de control diseñada para los requerimientos de un proceso específico con la sintonización de las tres variables mencionadas. La respuesta del controlador puede ser descrita en términos de la respuesta del control a un error, el grado en el cual el controlador sobrepasa el set-point y el grado de oscilación del sistema.

Los controladores tipo PID pueden ser de diferentes combinaciones de estas variables manipuladas. PI, PD, P o I.

Término Proporcional: El término proporcional hace un cambio en la salida del controlador la cual es proporcional al valor del error actual. La respuesta proporcional puede ser ajustada multiplicando el error por una constante proporcional Kp, llamada ganancia proporcional.

El término proporcional está dado por la ecuación:

$$P_{out} = K_p * e(t)$$

Donde:

P_{out}: Término proporcional de salida.

K_p: La ganancia proporcional.

e : $Error = SP - VP$ (Setpoint – Valor del Proceso).

t: Tiempo o instante de tiempo.

Una alta ganancia proporcional resulta en un gran cambio en la salida para un dado cambio en el error. Si la ganancia proporcional es muy alta, el sistema puede volverse inestable. Por el contrario, una ganancia pequeña resulta en una respuesta pequeña de salida para un gran error de entrada, y por lo tanto un controlador menos sensible. Si la ganancia proporcional es muy pequeña, la acción de control puede ser muy pequeña cuando responda a alteraciones del sistema.

Término Integral: La contribución del término integral es proporcional tanto a la magnitud del error como a la duración del error. Integrando el error sobre el tiempo se produce el offset acumulado que debió haber sido corregido previamente. El error acumulado es multiplicado por la ganancia integral y adicionado al controlador de salida. La magnitud del término integral sobre la acción de control es determinada por la ganancia integral.

El término integral está dado por:

$$I_{out} = Ki \int_0^t e(T)dT$$

Donde:

I_{out}: Término integral de salida.

Ki: Ganancia integral.

Ki: Error = SP – VP

t: Tiempo.

T: Variable de integración.

El termino integral (adicionado al término proporcional) acelera el movimiento del proceso hacia el set-point y elimina el error residual del estado estacionario (el error que permanece) que ocurre con un sólo controlador proporcional. Sin embargo, ya que el término integral responde a los errores acumulados del pasado, puede hacer que el valor actual supere el valor de set-point ocasionando un sobrepaso de la medida de control, aumentando o disminuyendo desproporcionadamente el valor de salida de la variable manipulada.

Término Derivativo:

$$D_{out} = Kd * \frac{de}{dt}(t)$$

El término derivativo disminuye la tasa o cantidad de cambio de la salida del controlador, esto es, debido a que la acción de control derivativa responde al cambio del error instantáneamente y produce una corrección significativa antes de que la magnitud del error se vuelva demasiado grande; este efecto es más notable cerca del set-point del controlador (Ogata, 1998). Este controlador es capaz de prever el error e iniciar una acción correctiva a tiempo con lo que tiende a aumentar la estabilidad del sistema.

El término derivativo solo actúa cuando hay un cambio en el error, es por esta razón que siempre se utiliza con los términos proporcional y/o integral.

La importancia de los controladores PID radica en que se aplica de manera casi general – es decir, aplica el mismo algoritmo – a la mayoría de sistemas de control. Tiene como ventaja que es bien conocido entre los sistemas para control de procesos y presenta como desventaja que cuando existen muchas variables a ser controladas dentro de un mismo proceso no es óptimo y se requiere mucho tiempo en el procesamiento de los datos, como por ejemplo en robótica móvil.

El controlador P se utiliza en sistemas que dependan de la magnitud de su error, y no de la velocidad de cambio de éste. El controlador PI se utiliza en aplicaciones

con aplicaciones que pueden tolerar cantidades específicas de sobrepaso y se necesita un restablecimiento del valor del error, como en el control de velocidad de un móvil, como en el llenado de un tanque por ejemplo. Y El controlador PD se utiliza en sistemas de respuesta rápida.

2.1.3. Lógica Difusa.

La lógica difusa o también llamada lógica borrosa apareció alrededor de 1965 por Lotfi A. Zadech de la universidad de california en Berkeley, y desde que este científico planteó la teoría de conjuntos difusos se han realizado múltiples aplicaciones, sobre todo en el campo de control automático, y en la robótica móvil. Sin embargo antes de comenzar a hablar de lógica difusa hay que recordar de qué trata la teoría clásica de la lógica y de los conjuntos. La teoría de conjuntos clásica parte de que una persona u objeto hace parte de cierto grupo de objetos llamado conjunto. Los objetos pertenecientes a estos conjuntos tienen ciertas cualidades y propiedades en común, las cuales tienen que cumplirse totalmente para que un objeto pertenezca a este. La pregunta para evaluar la pertenencia a un conjunto clásico tiene la forma ¿este objeto pertenece a este conjunto? , y la respuesta solo puede ser, si o no, pero no puede ser una respuesta ambigua. En cambio en estadística y probabilidad, el enfoque es diferente, la pregunta tiene la forma, ¿Cuál es la probabilidad de que esto pase? Y la respuesta es un valor porcentual de 0 a 100%. De esa forma se representan los conjuntos difusos. Se basan en representaciones porcentuales de cada objeto referente a cada conjunto. Por ejemplo si se busca saber cuáles son los objetos pertenecientes a “x” conjunto, el enfoque en lógica difusa seria hallar la pertenencia de cada objeto evaluado respecto al conjunto “x”, mientras que en lógica clásica se agrupan los elementos que solo pertenecen totalmente a este conjunto, descartando muchos que pertenecen parcialmente, debido que no se permite que un elemento pertenezca y a la vez no pertenezca al mismo conjunto, por ende hay muchos problemas del mundo real que no pueden ser representados por conjuntos clásicos, porque existen muchos problemas donde las respuestas no siempre tienen que ser totalmente ciertas, entonces es necesario incluir un grado de incertidumbre; como

decir un vaso está por la mitad. El vaso puede estar medio lleno, pero también medio vacío, y estas consideraciones son permitidas en la lógica difusa.

Conjuntos clásicos:

$$Fx = \{0,1\}$$

Ejemplo:

Donde se dice que en fx hay dos elementos 0 y 1.

Conjuntos difusos:

$$Fx = [0,1]$$

Los conjuntos difusos trabajan con la pertenencia absoluta a un conjunto u otro, se busca el valor porcentual de la pertenencia del objeto a cada conjunto. Los conjuntos difusos suelen simbolizarse por variables lingüísticas, variables que el hombre suele usar para calificar las cosas, como adjetivos que se le coloca a ciertos compartimientos, como por ejemplo, la velocidad de un carro , puede dividirse en tres grupos: rápido, lento, normal. Estas variables lingüísticas asignadas a cada conjunto, suelen representarse gráficamente.

Representaciones graficas de los conjuntos: Los conjuntos suelen representarse por medio de graficas correspondientes a una función, lo ideal es buscar la función y grafica que mas aplique a la necesidad del problema a abordar. Triángulos, trapecios, sinusoidales, puede ser usada cualquier forma que pueda ser representada por medio de una función grafica, donde las más comunes son las mencionadas. Y de todas estas las más usadas son los triángulos y trapecios, por ser las que menos consumo de procesamiento requieren a la hora de los cálculos. Esto se ve claramente al momento de realizar una simple multiplicación y suma como en el caso de los triángulos y $y = mx + b$, en cambio hallar una función

trigonométrica $y = sin\theta$, implicara un proceso más largo, lo que finalmente determinara mayor consumo de recursos.

Control difuso: El control difuso se diferencia del control convencional en que trabaja con valores borrosos o difusos, en vez de trabajar con valores absolutos, su proceso se encuentra dividido en tres etapas. Lo primero es convertir los valores medidos a conjuntos difusos, proceso que lleva el nombre de fusificación; realizar las combinaciones de reglas del sistema, conocido como implicación; y la conversión de los valores difusos a valores numéricos se conoce como, defusificación. Este tipo de control tiene una gran aplicación en la robótica móvil y cuando se requiere modelar procesos con comportamientos no lineales.

- Fusificación: En esta etapa se convierte el valor medido al lenguaje difuso, y con su valor de pertenencia al mismo. Se puede trabajar con múltiples variables, así como también se puede obtener múltiples valores de pertenencia a diferentes conjuntos, todo depende del sistema.

  Ejemplo:

  Conjuntos: rápido, lento, normal.

  Estos son los rangos de los conjuntos.

  Lento [0,20]

  Normal [20,40]

  Rápido [40, 60]

  En lógica clásica la representación grafica sería la siguiente.

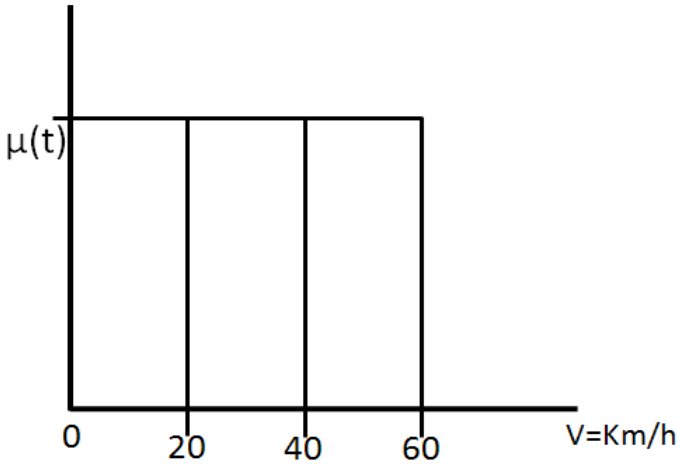


Figura 123. Representación grafica en lógica clásica

En el enfoque de lógica difusa, la grafica seria así:

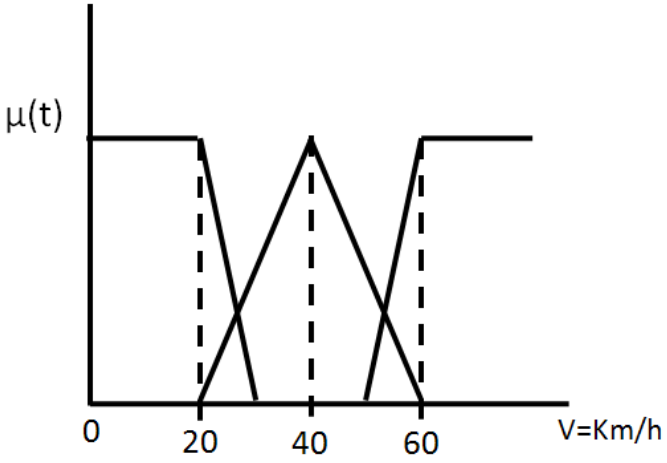


Figura 124. Representación grafica en lógica difusa

El cálculo de esta pertenencia se hace por medio de la ecuación que describa a la representación del conjunto, ejemplo en el caso de triángulos será $y = mx + b$.

- Implicación: Todo proceso a controlar trabaja bajo una base de conocimiento, el cual se obtiene del estudio previo del proceso que se quiere

modelar y controlar. Esta base del conocimiento genera ciertas reglas o respuestas a posibles cambios, en el caso de la lógica difusa estas reglas tienen una estructura lingüística muy similar al razonamiento de los seres humanos. Ejemplo: si la velocidad es baja y la distancia es cerca la acción de control es alta, en este ejemplo luego de tener una velocidad medida baja, se realiza una acción de control alta, para tratar de llevar la velocidad al nivel deseado. El proceso donde se combinan todas estas reglas se llama implicación.

Métodos para las reglas:

[31]Lista de reglas básicas para la defusificación por centro de máximos.

$$\mu_A(a \wedge b) = \mu_A(a) \wedge \mu_A(b) \min\{\mu_A(a), \mu_A(b)\};$$

$$\mu_A(a \vee b) = \mu_A(a) \vee \mu_A(b) \max\{\mu_A(a), \mu_A(b)\};$$

$$\mu_A(\bar{a}) = 1 - \mu_A(a);$$

$$\mu_A(a \to b) = \mu_A(a) \to \mu_A(b) = \min\{1, 1 + \mu_A(b) - \mu_A(a)$$

$$\mu_A(a \leftrightarrow b) = \mu_A(a) \leftrightarrow A(b) = 1 - |\mu_A(a) - \mu_A(b)|$$

Ejemplos:

Si es un y la salida se le aplica el método del mínimo.

U(a y b) = u(a) y u(b) = min (u(a),u(b));

Si es un or la salida se aplica el max

[31] Goebel, 2003. p. 68.

U(a o b) = u(a) o u(b) = max (u(a),u(b))

- Defusificación: En este proceso se convierten los valores difusos obtenidos en los procesos anteriores, a valores numéricos. Puesto que luego del proceso de implicación y combinación de reglas se obtiene una nueva forma del conjunto difuso de salida, la cual se encuentra en forma de pertenencias a conjuntos difusos. Para poder culminar el proceso se convierten estos valores porcentuales de cada conjunto a datos numéricos. Esto se puede realizar por diferentes métodos, dentro de los más conocidos están:

  Centro de área

  Centro de máximos

  Primero de máximos

  El más usado es el de centro de área debido a la sencillez de sus cálculos, y poco consumo de recursos de los sistemas de cómputo. Pero todo depende del proceso a controlar, y de las herramientas que se usen. Lo que se busca es llegar a un equilibrio entre capacidad de procesamiento, y exactitud deseada.

  La fórmula para el centro de área es la siguiente:

$$Z = \frac{m\mu_1 * (salida1) + m\mu_2 * (salida2) + m\mu_3 * (salida3) + \cdots ... m\mu_n * (salidan)}{m\mu_1 + m\mu_2 + m\mu_3 + \cdots . m\mu_n}$$

  Siendo “z” el valor de salida entregado al proceso de salida.

La lógica difusa ha tenido un auge y aceptación en diferentes áreas de la ciencia, debido a sus múltiples aplicaciones, de los cuales se destacan, control de procesos industriales y procesos donde se use control adaptativo. Generalmente suele usarse en sistemas no lineales, difíciles de modelar, y de presentar con una ecuación, debido a las condiciones cambiantes del proceso; por ende suelen ser muy usados

en la robótica. Principalmente en la robótica móvil, debido a su facilidad de adaptación en entornos poco estructurados.

La forma de evaluar las reglas es simplemente obtener los valores de salida y los valores de μ_n.

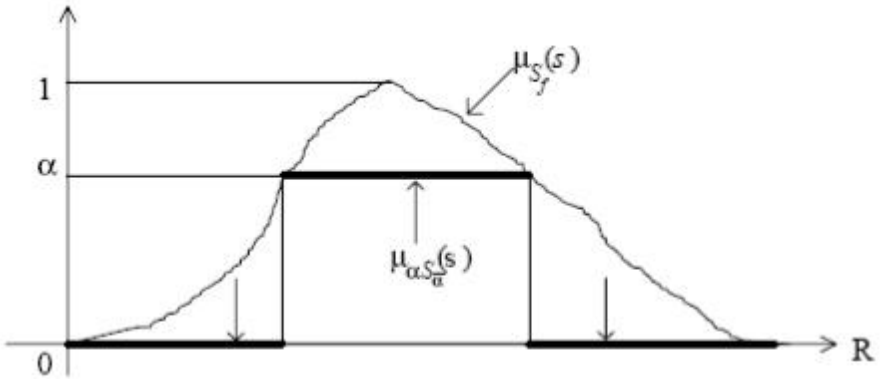


Figura 125. Valores de salida y valores para

Ejemplo:

Se buscará controlar la distancia de un robot móvil, y la velocidad. Las variables de entrada serán, error de distancia, y velocidad, la variable de salida es la velocidad.

Variable de entrada error de distancia.

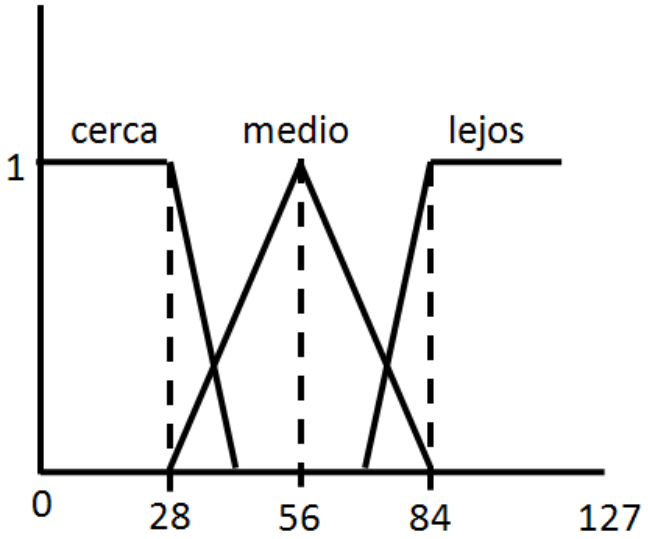


Figura 126. Variable de entrada de error de distancia

Velocidad.

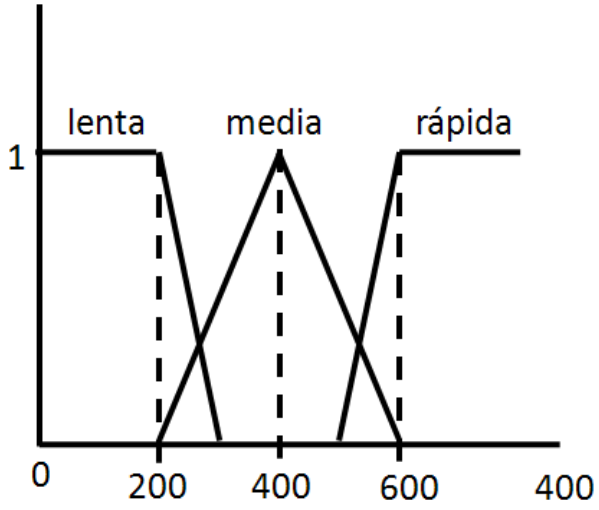


Figura 127. velocidad

Variable de salida.

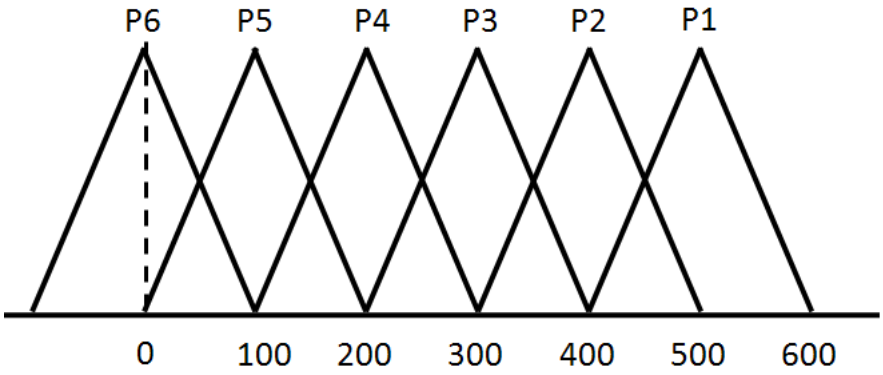


Figura 128. Variable de salida

Conjunto de reglas:

	Lento	Media	Rápida
Cerca	P1	P2	P5
Medio	P4	P5	P5
Lejos	P6	P5	P0

Tabla 30. Conjunto de regla

Reglas:

Si la distancia es cerca y la velocidad es lejos entonces la salida es p1.

Si la distancia es cerca y la velocidad es media entonces la salida es p2.

Si la distancia es cerca y la velocidad es rápida entonces la salida es p5.

Si la distancia es media y la velocidad es lenta entonces la salida es p4.

Si la distancia es media y la velocidad es media entonces la salida es p5.

Si la distancia es media y la velocidad es rápida entonces la salida es p5.

Si la distancia es lejos y la velocidad es lenta entonces la salida es p6.

Si la distancia es lejos y la velocidad es media entonces la salida es p5.

Si la distancia es lejos y la velocidad es rápida entonces la salida es p0.

La medida es 0.3 cerca, y 0.7 media.

Velocidad 0.4 media 0.6 rápida.

Revisando las 4 combinaciones posibles con los datos obtenidos las reglas entonces son:

Cada salida se evalúa según la regla correspondiente a los métodos descritos arriba (revisar 0).

Salida 2= p2

$$\mu_{2=}\min(0.3,0.4) = 0.3$$

Salida 3=p5

$$\mu_3 = \min(0.3,0.6) = 0.3$$

Salida 5= p5

$$\mu_5 = \min(0.7,0.4) = 0.4$$

Salida 6= p5

$$\mu_6 = \min(0.7,0.6) = 0.6$$

Los valores de salida 1, 4, 7, 8 y 9, al igual que cada μ correspondiente, será cero, por ser la pertenencia a estos conjuntos cero, según los datos del ejemplo.

En la etapa de la defusificación según la fórmula descrita en (0)

$$Z = \frac{0.3 * (400) + 0.3 * (100) + 0.4 * (100) + 0.6 * (100)}{0.3 + 0.3 + 0.4 + 0.6} = 156.25$$

Ese sería el valor de salida según los valores de entrada del ejemplo.

3. Construyendo Sistemas difusos con el software Fuzzy Logic Toolbox[32].

Herramientas de interfaz grafica de usuario Fuzzy Logic Toolbox.

En esta sección, se usan las herramientas de la interfaz gráfica de usuario Fuzzy Logic Toolbox (GUI) para construir un sistema de Inferencia Fuzzy (FIS por Fuzzy Inference System) para el ejemplo de control de velocidad – descrito posteriormente. Aunque se puede construir un FIS trabajando estrictamente desde un comando de línea, esto es mucho más fácil que construir un sistema grafico.

El GUI de Fuzzy Logic Toolbox no soporta la construcción de FIS usando datos. Si se desea dar datos para la construcción de un FIS, se utilizan las siguientes técnicas:

- Los comandos genfis1, genfis2, o genfis3 para generar un FIS tipo Sugeno. Se puede seleccionar File > Import en el editor FIS para importar el FIS y realizar la inferencia fuzzy, como se describe en el editor de FIS (FIS Editor).
- Técnicas de aprendizaje Neuro-Adaptativas para modelar el FIS, como se describe en anfis y en el editor GUI ANFIS.

Se pueden utilizar cinco herramientas primarias de GUI para construir, editar, y observar el sistema de inferencia difuso en el toolbox:

- FIS Editor (Editor FIS – Fuzzy Inference System, en castellano Sistema de Inferencia Difuso).

- Membership Function Editor (Editor de Función Miembro).

[32] MathWorks. (2009). Fuzzy Logic Toolbox.

- Rule Editor (Editor de Regla).
- Rule Viewer (Visor de Regla).
- Surface Viewer (Visor de Superficie).

Estos GUI's son conectados dinámicamente, en donde los cambios que se hagan al FIS usando uno de ellos, puede afectar lo que se ve en otro GUI abierto. Se puede tener cualquiera o todos ellos abiertos para cualquier sistema dado.

El FIS Editor maneja los asuntos de alto nivel para el sistema: ¿Cuántas variables de entrada y salida? ¿Cuáles son sus nombres? El software Fuzzy Logic Toolbox no limita el número de entradas. Sin embargo, el número de entradas debe ser delimitado por la memoria disponible en la máquina. Si el número de entradas es muy extenso, o el número de funciones miembro es muy grande, entonces será difícil para analizar el FIS usando otras herramientas GUI.

El Membership Function Editor es usado para definir las formas de toda la función miembro asociadas a cada variable.

El Rule Editor es para editar la lista de reglas que define el comportamiento del sistema.

El Rule Viewer y el Surface Viewer son usados para observar, más no para editar, el FIS. Son herramientas de sólo lectura estrictamente. El RuleViewer es un entorno computacional técnico de MATLAB basado en la visualización del diagrama de inferencia difuso. Usado como un diagnóstico, el puede mostrar (por ejemplo) cuáles reglas están activas, o cómo las formas individuales de la función miembro influencian los resultados. El Surface Viewer es usado para mostrar la dependencia de una de las salidas ya sea una o dos de las entradas por lo que, se genera y dibuja una superficie de salida del sistema completo.

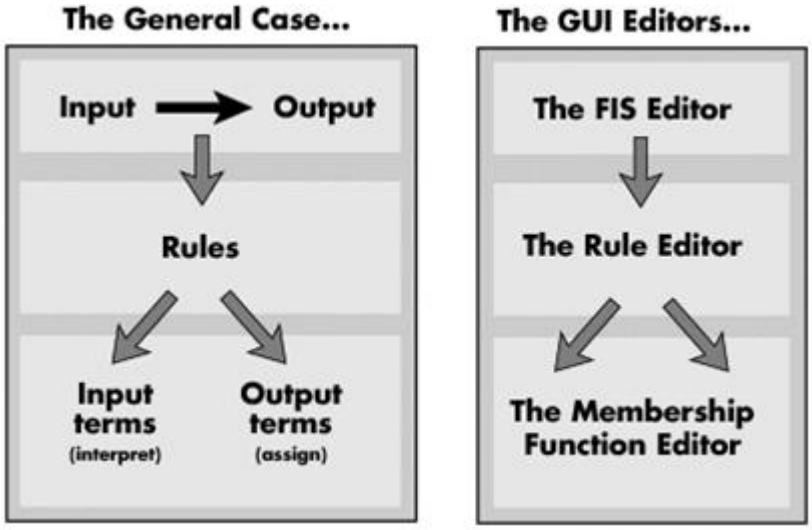


Figura 129.Surface Viewer

Las cinco GUI's pueden todas interactuar e intercambiar información. Cualquiera de ellas puede leer y escribir en el workspace (espacio de trabajo) y en un archivo (las vistas de sólo lectura pueden aún intercambiar gráficos con el workspace y guardarlos en un archivo). Para cualquier FIS, cualquiera o todos estos cinco GUI's pueden ser abiertos. Si más de uno de estos editores es abierto para un sistema individual, las diferentes ventanas GUI's son advertidas de la existencia de las otras, y, si es necesario, actualiza las ventanas relacionadas. Así, si las funciones miembro son cambiadas usando el editor de función miembro, esos cambios son reflejados en las reglas mostradas en el editor de regla. Los editores para cualquier número de diferentes FIS pueden ser abiertos simultáneamente. El FIS Editor, el Membership Function Editor, y el Rule Editor pueden leer y modificar los datos FIS, pero el Rule Viewer y el Sourface Viewer no modifican los datos FIS en ninguna manera.

3.1. Cómo empezar.

Se comenzará con una descripción básica de dos entradas, una salida de velocidad.

El control de velocidad: Dependiendo de la posición del robot, el cual puede estar cerca o lejos de un objeto, y la velocidad actual de este mismo, ¿cuál debe ser la regulación de velocidad?

El punto de inicio es escribir las reglas para el control de velocidad, basadas en la experiencia.

1. Si la distancia es grande y la velocidad actual rápida, la regulación de velocidad es la mayor.

2. Si la distancia es pequeña y la velocidad actual es lenta, la regulación de velocidad es la mínima.

De esta forma cuando se obtengan valores intermedios no se podrá realizar una acción de acorde a esto.

Teniendo conocimiento a cerca de las reglas y teniendo una idea de cómo debe ser la salida se pueden utilizar las herramientas GUI para la construcción de un Sistema de Inferencia Fuzzy, con el objetivo de determinar un control más completo y exacto.

3.1.1. El FIS Editor.

El editor FIS muestra la información a cerca de un sistema de inferencia fuzzy. Para abrir el Editor FIS, se debe escribir el comando fuzzy y dar click a la tecla enter; luego se encuentra la siguiente ventana:

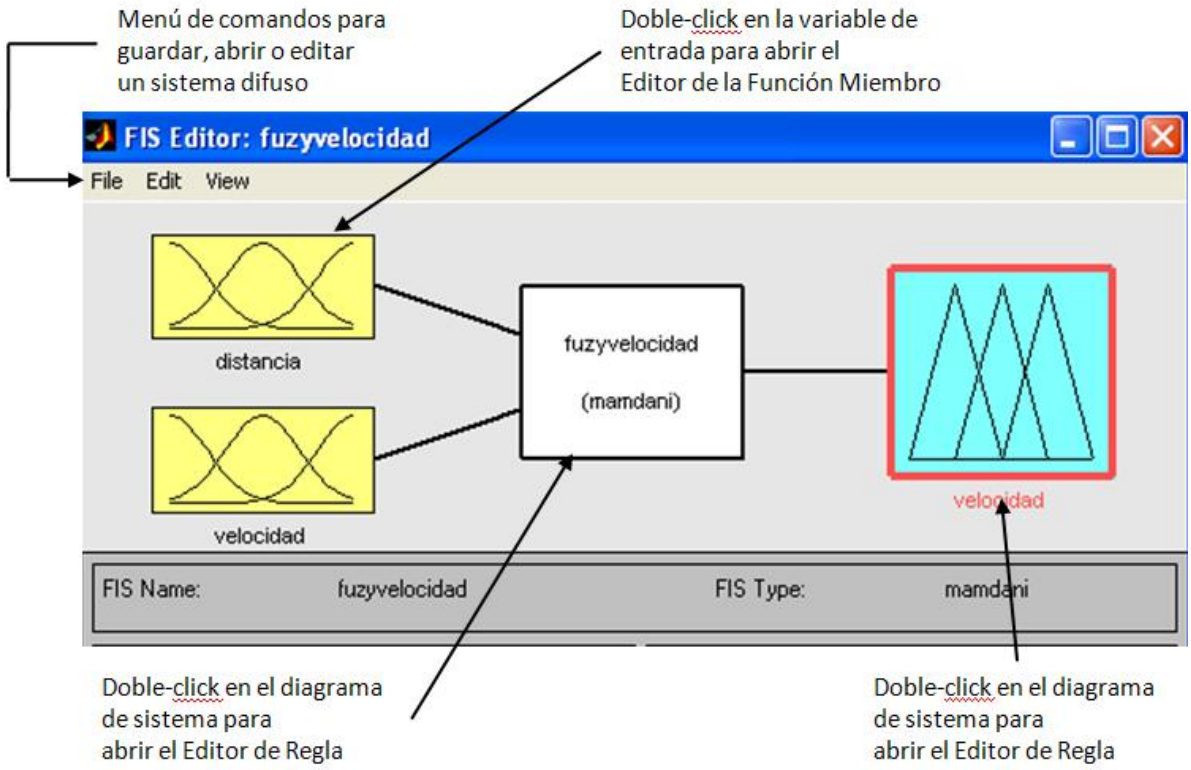


Figura 130.Editor FIS

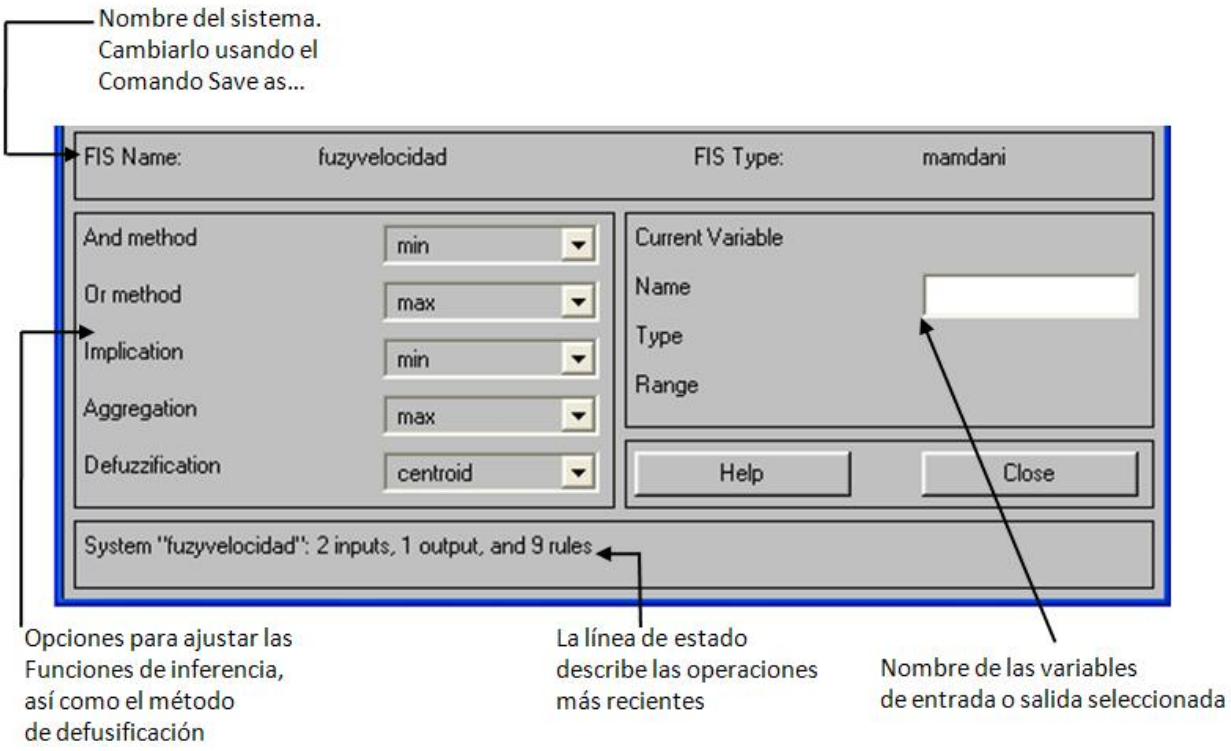


Figura 131.Variables a estudiar

El diagrama de arriba muestra los nombres de cada variable de entrada en la izquierda, y de cada una de aquellas variables de salida en la derecha. La misma función miembro mostrada en la ventana son sólo íconos y no representan las formas actuales de la función miembro:

- En la parte de abajo del diagrama está el nombre del sistema y del tipo de inferencia utilizado. El predeterminado tipo de inferencia Mandani, que se utiliza en este ejemplo.

- En la parte de abajo del diagrama el nombre del sistema de inferencia difuso, en el lado izquierdo de la figura, son los menús emergentes que permiten modificar las diferentes piezas del proceso de inferencia.

- En la derecha en la parte superior de la figura se es el área que muestra el nombre de ya sea una variable de entrada o salida, está asociada al tipo de función miembro, y su rango.

- Los dos campos siguientes son especificados solo después de la función miembro.

- Debajo de esta región están los botones de Help (ayuda en línea) y Close (cerrar la ventana). En la parte superior esta el estado de la línea que transmite la información acerca del sistema.

El Editor FIS abre un documento genérico “sin”, con una entrada llamada input1, y una salida llamada output1.

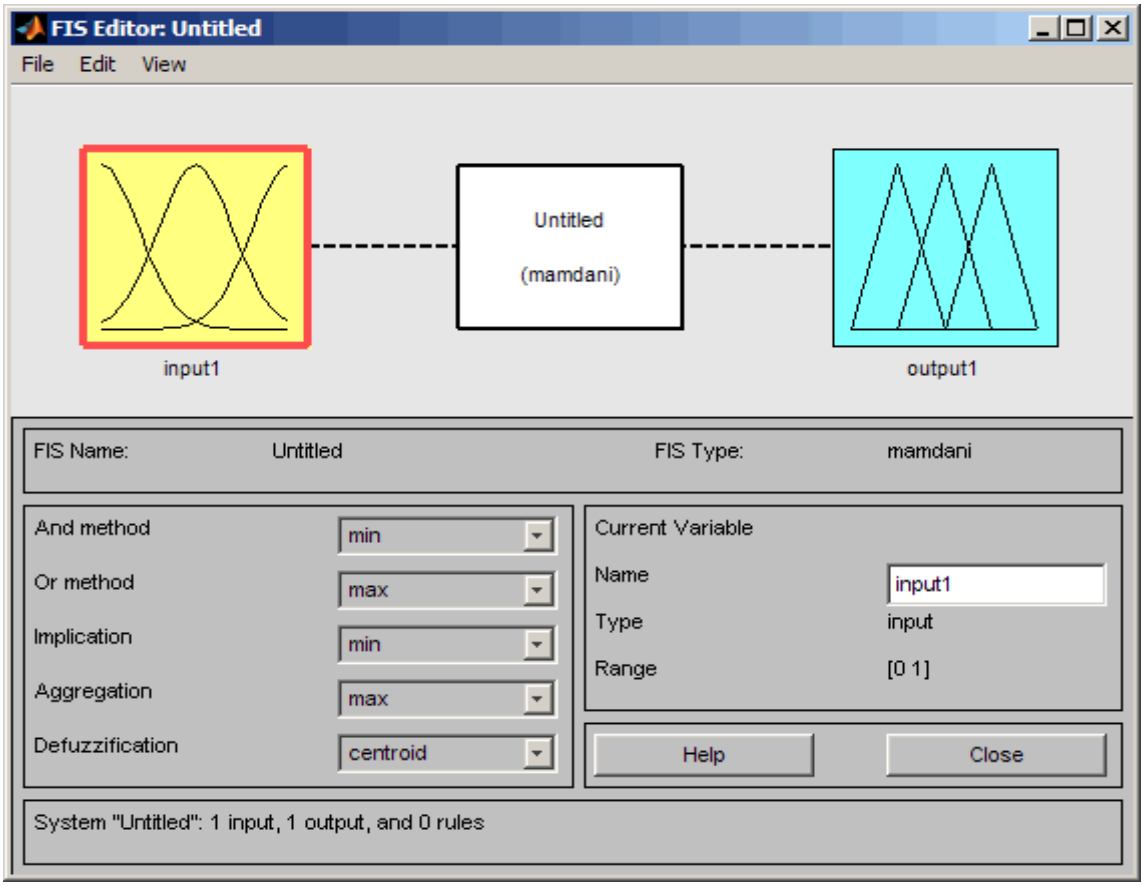


Figura 132.Ejemplo de sistema de 2E/1S

En este ejemplo, se construye un sistema de dos entradas y una salida. Las dos entradas son distancia y velocidad. La salida es velocidad.

Para adicionar una segunda variable y cargar los nombres de las variables a fin de reflejar estas designaciones:

1. Seleccionar Edit > Add variable > Input.

Un segundo cuadro amarillo aparece llamado input2.

2. Click el cuadro amarillo input1. Este cuadro está resaltado con un contorno rojo.

3. Editar el campo del nombre de input1 a distancia, y presionar Enter.

4. Dar click en el cuadro amarillo input2. Este cuadro es resaltado con un contorno rojo.

5. Editar el campo Name de input2 a velocidad, y presionar Enter.

6. Dar click en el cuadro azul output1.

7. Editar el campo Name de output1 a velocidad, y presionar Enter.

8. Seleccionar File > Export > To Workspace.

9. Introduzca el nombre de la variable Workspace en este caso Velfuzzy, y dar click en OK.

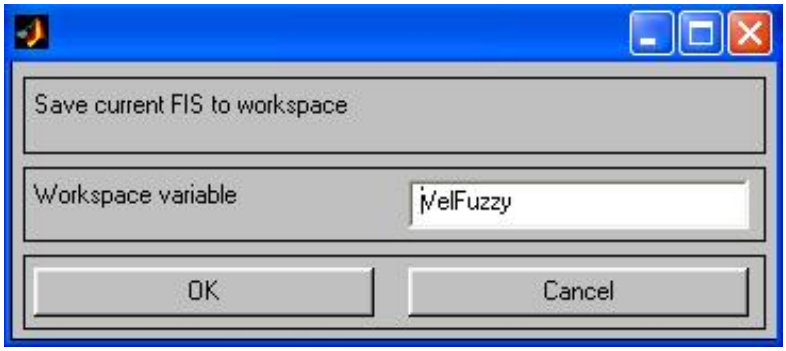


Figura 133.Paso 9

El diagrama es actualizado para reflejar los nuevos nombres de las variables de entrada y salida. Hay ahora una nueva variable en el workspace llamada Velfuzzy que contiene toda la información de este sistema. Guardándola con el nuevo nombre en el workspace, también se puede renombrar el sistema entero. La ventana luce como el siguiente diagrama.

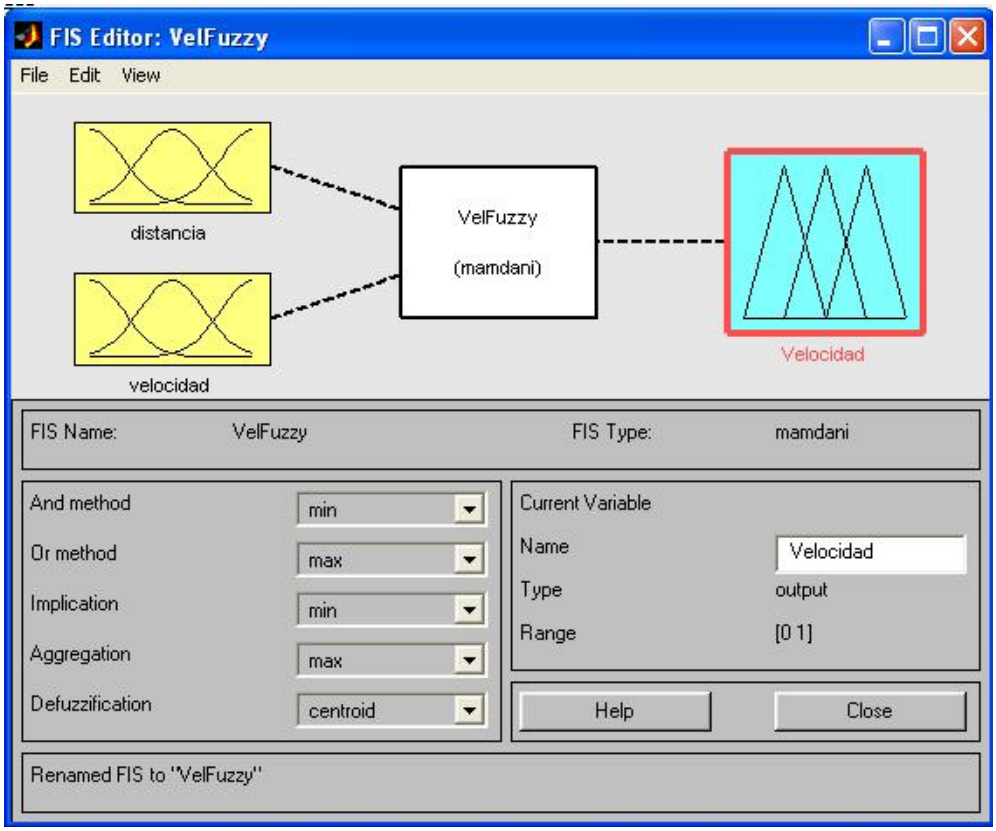


Figura 134.Sistema VelFuzzy

Dejar las opciones de la inferencia en la posición predeterminada por ahora, se encuentran en la izquierda en la parte baja de la ventana. Se habrá introducido toda la información necesaria para este GUI en particular. Luego, se definen la función miembro asociadas a cada una de las variables. Para hacer esto, abrir el Membership Function Editor.

Se puede abrir este editor de una de las siguientes formas:

- Dentro de la ventana del editor FIS, seleccionando Edit > Membership Functions.
- Dentro de la ventana del editor FIS, dando doble click al ícono llamado Velocidad.
- Escribiendo en la línea de comando mfedit.

3.1.2. El Membership Function Editor.

El editor de la función miembro es la herramienta que permite mostrar y editar todas las funciones asociadas con todas las variables de entrada y salida para el sistema de inferencia difuso. El editor de la función miembro comparte todas sus características con el editor FIS, como se muestra en la figura siguiente de hecho, todas las cinco herramientas básicas GUI tienen opciones de menú similares, líneas de estado, y botones de Help y Close.

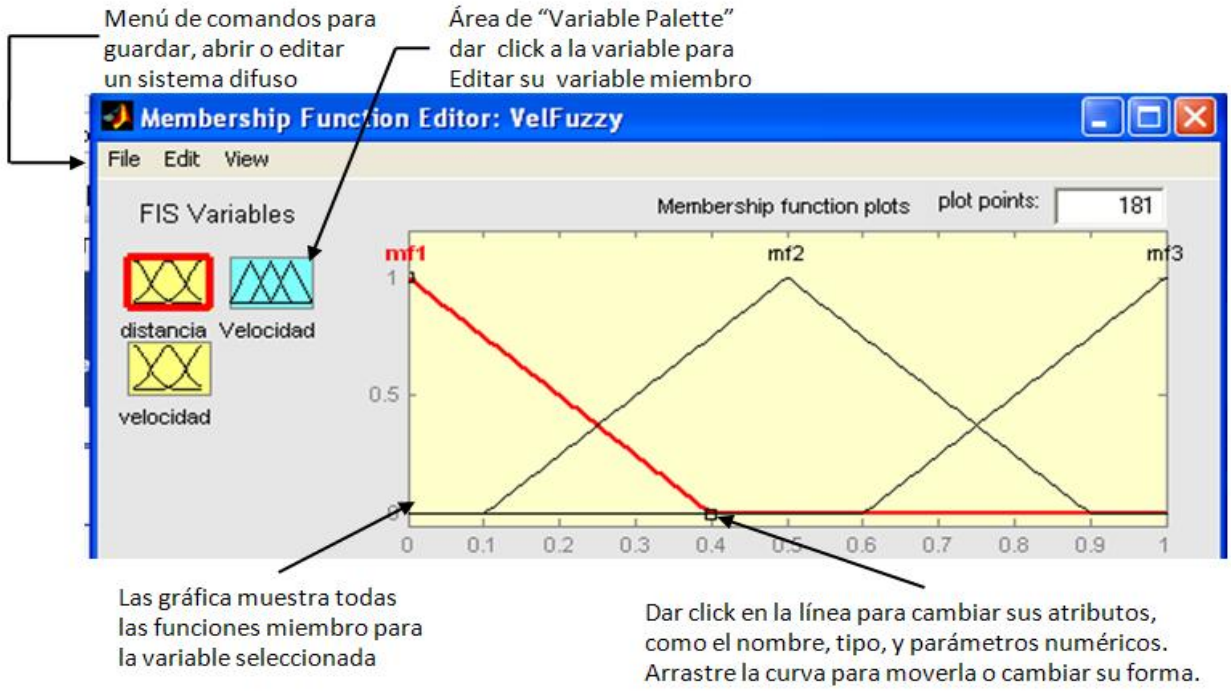


Figura 135.Editor de la función miembro

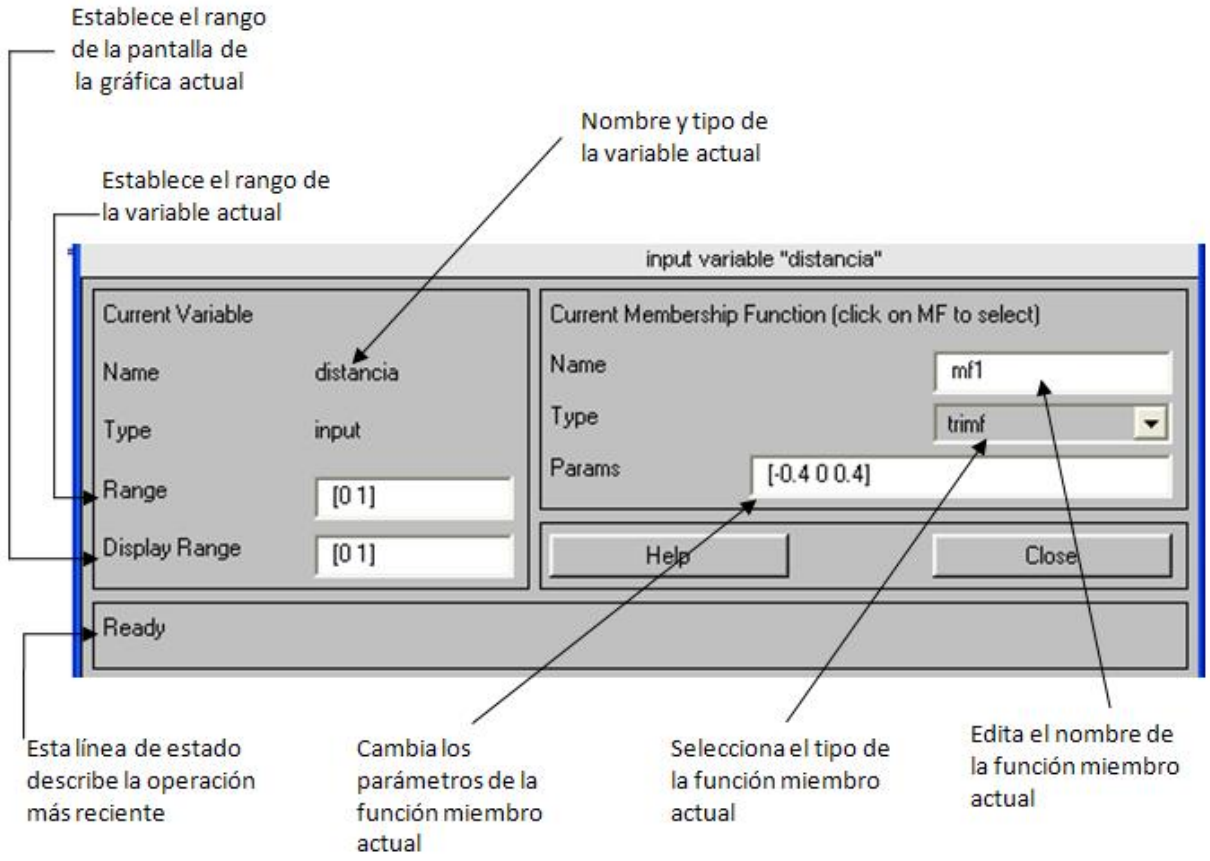


Figura 136.Parámetros del Editor de la función miembro

Cuando se abre el Membership Function Editor para trabajar en un sistema de inferencia difuso que aún no existe en el workspace, no hay funciones miembro asociadas con las variables definidas en el editor FIS.

En la parte de arriba a la izquierda del área de la grafica en el editor de la función miembro está una “Variable Palette” que permite establecer la función miembro para una variable dada.

Para crear la función miembro asociadas con la variable de entrada o salida para el FIS, se selecciona una variable FIS en esta región dando click en ésta.

Luego se selecciona el menú desplegable Edit, y se escoge Add MFs.. Una nueva ventana aparece, lo que permite seleccionar el tipo de función miembro y el número de funciones miembro asociadas con la variable seleccionada. En la esquina

de la parte baja a la izquierda de la ventana están los controles que permiten cambiar el nombre, tipo, y parámetros (forma), de la función miembro, después de seleccionarla.

Las funciones miembro de la variable actual se muestran en la grafica principal. Estas funciones miembro pueden ser manipuladas de dos maneras. Primero se puede usar el mouse para seleccionar una función miembro particular asociada con la cualidad de la variable dada, (la cual puede ser cerca, para la variable distancia), y luego arrastrar la función miembro lado a lado. Esta acción afecta la descripción matemática de la cualidad asociada con la función miembro para la variable dada. La función miembro seleccionada puede también ser seleccionado por dilatación (ampliando el rango de los parámetros de la función miembro) o contracción (disminuyendo el rango de los parámetros de la función miembro) dando click en los cuadros pequeños de punto de arrastre en la función miembro, y luego arrastrando la función con el mouse hacia el exterior, por dilatación, o hacia el interior, por contracción. Esta acción cambia los parámetros asociados con la función miembro.

Debajo la Variable Palette es una especie de información acerca del tipo de nombre de la variable actual. Hay un campo de texto en la región que permite cambiar los límites del rango de la variable actual y la otra permite establecer los límites de la grafica actual.

El proceso de especificar la función miembro para el ejemplo del control de velocidad, VelFuzzy, es como se muestra a continuación:

1. Dar doble click a la variable de entrada distancia para abrir el Editor de la Función Miembro.

2. En el Editor de la Función Miembro, introducir [0 127] en los campos Rango y Display Range.

3. Crear la función miembro para la variable distancia.

- Seleccionar Edit> Remove All MFs. para eliminar las funciones miembro para la variable de entrada distancia.
- Seleccionar Edit> Add MFs. Para abrir el cuadro de dialogo de las Funciones Miembro.
- El cuadro de dialogo de las Funciones Miembro, seleccionar trapmf como el MF Type.

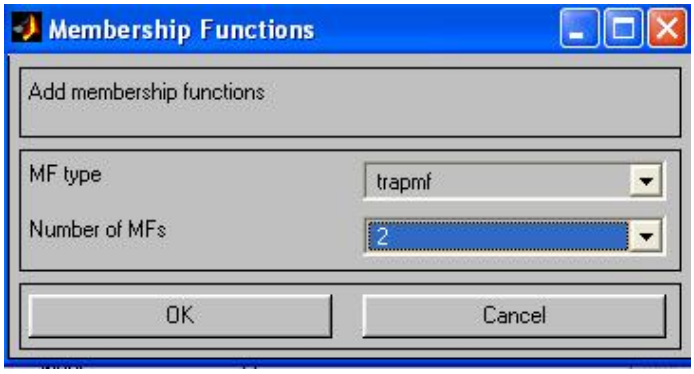


Figura 137. Primera función miembro.

- Se verifica que dos es seleccionado como el Number of MFs.
- Dar click en OK para agregar dos curvas Trapecio para la variable distancia.
- Se debe agregar otra curva para distancia, esta de tipo Triangulo.

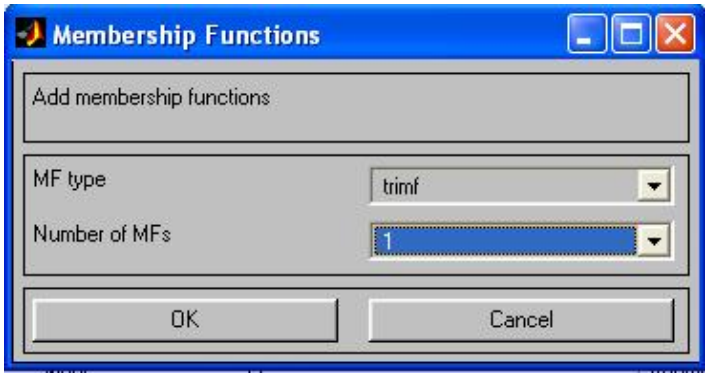


Figura 138. Segunda función miembro

4. Cambiar el nombre de las funciones miembro para la variable distancia, y especificar sus parámetros.

- Dar click en el nombre de la curva llamada *mf1* para seleccionarla, y especificar los campos siguientes en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir cerca.

  En el campo Params, introducir [-56 -28 28 56].

Las cuatro entradas para Params representan la desviación estándar del Trapecio.

Tip: Para ajustar la forma de la función miembro, escriba los parámetros deseados o utilice el mouse, como se describió previamente.

El Membership Function Editor: la ventana VelFuzzy se ve como la siguiente figura:

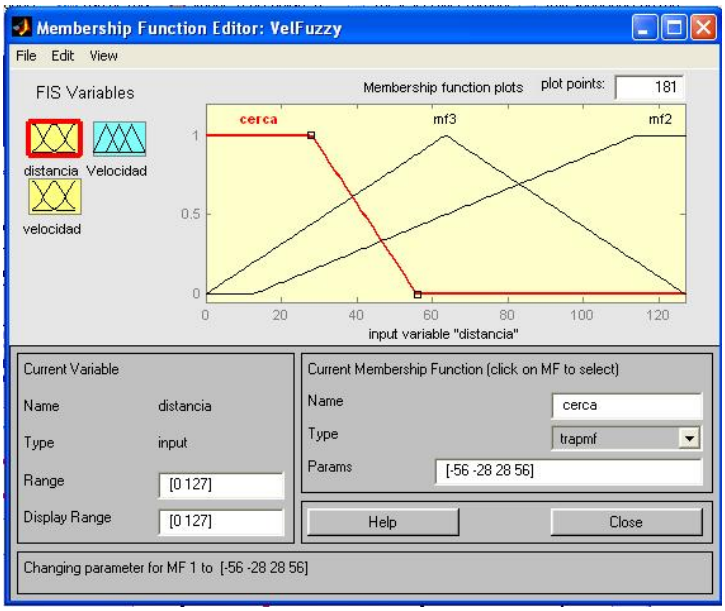


Figura 139.Ventana VelFuzzy

- Dar click en la curva llamada mf2 para seleccionarla, y especificar los siguientes campos en el área Current Membership Function (click on MF to select):

En el campo Name, introducir lejos.

En el campo Params, introducir [56 84 133.4 184.6].

El Membership Function Editor: la ventana VelFuzzy se ve como la siguiente figura:

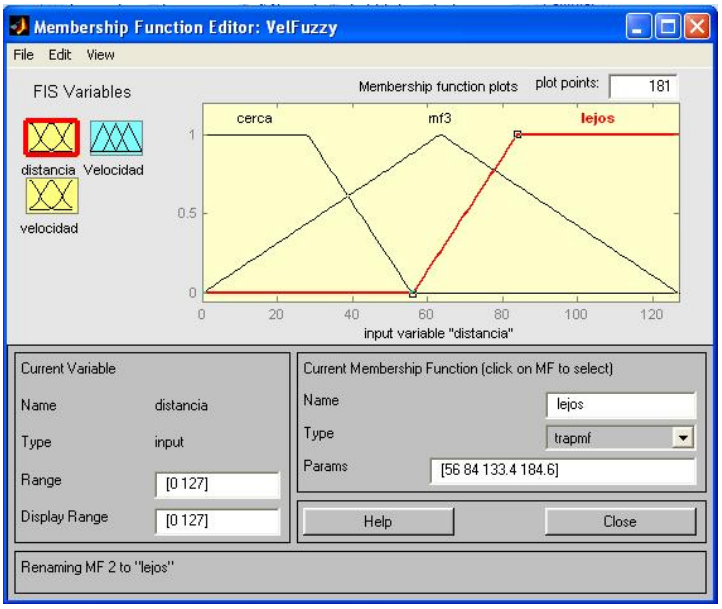


Figura 140.Ventana VelFuzzy para curva, mf2

- Dar click en la curva llamada mf3, y especificar los siguientes campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir media.

  En el campo Params, introducir media.

La ventana Membership Function Editor se verá como la imagen a continuación:

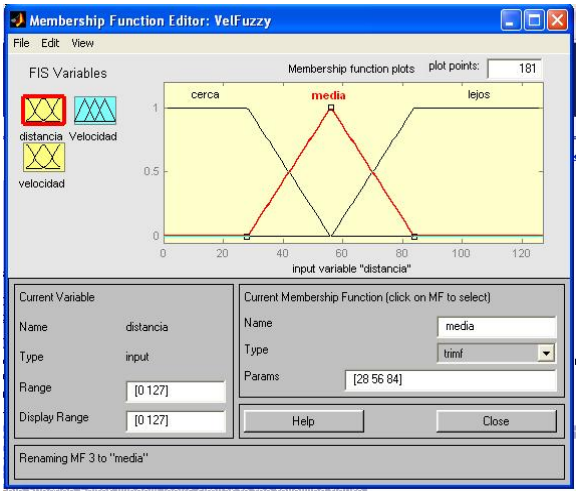


Figura 141.Ventana VelFuzzy para curva, mf3

5. En el área FIS Variables, dar click a la variable de entrada velocidad para seleccionarla.

6. Introducir [0 400] en los campos Range y Display Range.

7. Crear la función miembro para la variable de entrada velocidad.

- Seleccionar Edit>Remove All MFs para eliminar las Membership Functions para la variable velocidad.
- Seleccionar Edit>Add MFs para abrir el cuadro de dialogo de Membership Functions para la variable velocidad.
- En el cuadro de dialogo, seleccionar trapmf como MF Type.
- Seleccionar 2 en la dista desplegable de Number of MFs.
- Dar click en OK para agregar tres curvas tipo trapecio para la variable de entrada velocidad.
- Seleccionar Edit>Add MFs para abrir el cuadro de dialogo de Membership Functions para la variable velocidad.
- En el cuadro de dialogo, seleccionar trimf como MF Type.

- Seleccionar 1 en la dista desplegable de Number of MFs.
- Dar click en OK para agregar la curva tipo triangulo para la variable de entrada velocidad.

8. Cambiar los nombres de las funciones miembro para la variable de entrada velocidad, y especificar sus parámetros:

- En el área FIS Variables, dar click en la variable velocidad para seleccionarla.
- Dar click en la curva llamada mf1, y especificar los siguientes campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir lenta.

  En el campo Params, introducir [-600 -400 100 200].
- Dar click en la curva llamada mf2, y especificar los siguientes campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir rápida.

  En el campo Params, introducir [209 300 400 600].
- Dar click en la curva llamada mf3, y especificar los siguientes campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir media.

  En el campo Params, introducir [100 200 300].

9. Dar click en la variable de salida Velocidad para seleccionarla.

10. Introducir [0 600] en los campos Range y Display Range para cubrir el rango de salida.

11. Cambiar los valores por defecto de las funciones miembro triangulare por la variable de salida Velocidad, y especificar sus parámetros.

- Dar clic en la curva llamada mf1 para seleccionarla, y especificar los campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir p1.

  En el campo Params, introducir [400 500 600].

- Dar clic en la curva llamada mf2 para seleccionarla, y especificar los campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir p2.

  En el campo Params, introducir [300 400 500].

- Dar clic en la curva llamada mf3 para seleccionarla, y especificar los campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir p3.

  En el campo Params, introducir [200 300 400].

- Dar clic en la curva llamada mf4 para seleccionarla, y especificar los campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir p4.

  En el campo Params, introducir [100 200 300].

- Dar clic en la curva llamada mf5 para seleccionarla, y especificar los campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir p5.

  En el campo Params, introducir [0 100 200].

- Dar clic en la curva llamada mf6 para seleccionarla, y especificar los campos en el área Current Membership Function (click on MF to select):

  En el campo Name, introducir p6.

  En el campo Params, introducir [-100 0 100].

La ventana Membership Function Editor se verá como la imagen a continuación:

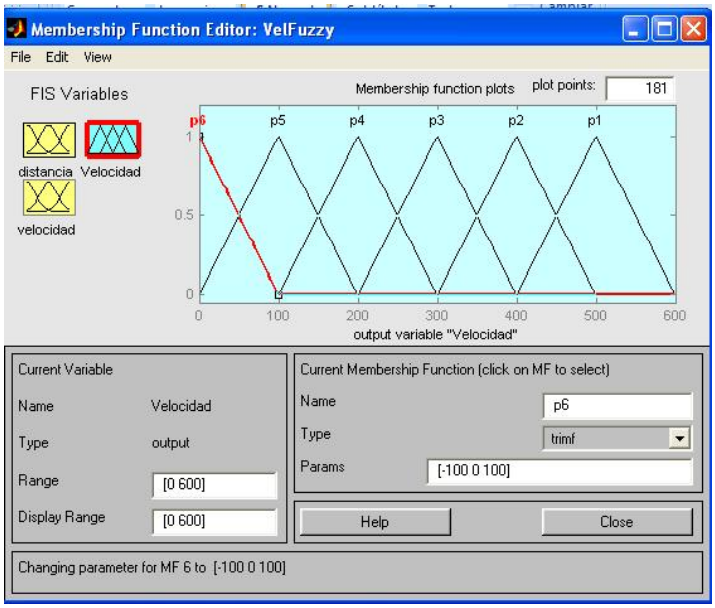


Figura 142.ventana Membership Function Editor

Ahora que las variables han sido nombradas y las funciones miembro han tomado formas y nombres, se pueden introducir las reglas. Para llamar al Rule Editor, se debe ir al menú Edit y seleccionar Rules, o escribir ruleedit en la línea de comando.

3.1.3. Rule Editor.

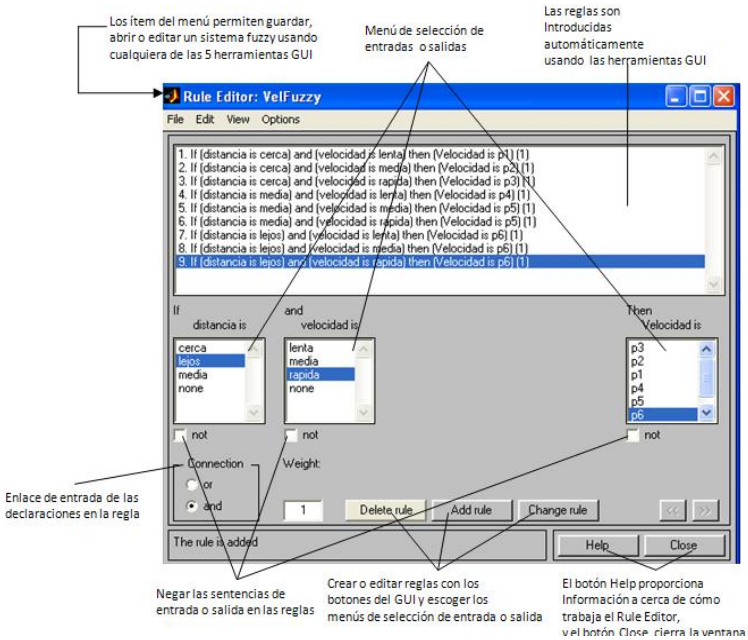


Figura 143.Rule Editor

Para construir las reglas usando la interfaz Rule Editor es bastante evidente. Basado en la descripción de las variables de entrada y salida definidas con el FIS Editor, el Rule Editor permite construir las sentencias automáticamente, desde el GUI, se puede:

- Crear las reglas seleccionando un ítem en cada cuadro de variable de entrada y salida, y un ítem Conection y dar clic en Add Rule. Se puede escoger *none* cono una cualidad de la variable para excluir esa variable de la regla dada y escoger *not* bajo cualquier nombre de variable para negar la cualidad asociada.

- Borrar una regla seleccionando la regla y dando click en Delete Rule.

- Editar una regla cambiando la selección en el cuadro de la variable y dando click en Change Rule.

- Especificar el peso a una regla escribiendo el número deseado entre 0 y 1 en Weight. Si no se especifica el peso, lo asume como 1.

Similar a estos en el FIS Editor y el Membership Function Editor, el Rule Editor tiene la barra de menú y la línea de estado (status line). Los ítems del menú permiten abrir, cerrar, guardar y editar el sistema difuso usando las herramientas básicas GUI. Desde el menú, también se puede:

- Establecer el formato para mostrar, seleccionando Options> Format.
- Establecer el idioma, seleccionando Options> Language.

También se puede acceder a la información acerca del Rule Editor haciendo click en Help y cerrar el GUI usando Close.

Para insertar la primera regla en el Rule Editor, seleccionar lo siguiente:

- cerca, debajo la variable distancia.
- lenta, debajo la variable velocidad.
- El botón de radio and, en el bloque Connection.
- p1, debajo de la variable de salida, Velocidad.

El resultado es:

If (distancia is cerca) and (velocidad is lenta) then (Velocidad is p1) (1)

Los números en el paréntesis representan los pesos.

Se debe seguir un procedimiento similar para insertar las demás reglas en el Rule Editor, para obtener:

1. If(distancia is cerca) and (velocidad is lenta) then (Velocidad is p1) (1)
2. If(distancia is cerca) and (velocidad is media) then (Velocidad is p2) (1)
3. If(distancia is cerca) and (velocidad is rápida) then (Velocidad is p3) (1)
4. If(distancia is media) and (velocidad is lenta) then (Velocidad is p4) (1)

5. If(distancia is media) and (velocidad is media) then (Velocidad is p5) (1)
6. If(distancia is media) and (velocidad is rápida) then (Velocidad is p5) (1)
7. If(distancia is lejos) and (velocidad is lenta) then (Velocidad is p6) (1)
8. If(distancia is lejos) and (velocidad is media) then (Velocidad is p6) (1)
9. If(distancia is lejos) and (velocidad is rápida) then (Velocidad is p6) (1)

Para cambiar una regla, primero dar click en la regla que se cambiará. Ahora hacer los cambios deseados para esa regla, y por último dar click en Change Rule.

En este punto, el sistema difuso ha sido completamente definido, en las variables, en las funciones miembro, y en las reglas necesarias para calcular que la velocidad. Ahora, para mirar el diagrama de la inferencia difusa y verificar que todo funciona de acuerdo a lo esperado se puede usar el Rule Viewer. Desde el menú View, seleccionar Rules.

3.1.4. El Rule Viewer.

El Rule Viewer muestra un mapa de ruta de todo el proceso de inferencia difusa. Está basado en el diagrama de inferencia difusa descrito anteriormente. Se muestra una ventana con graficas anexadas a ésta. La primera línea de gráficas representa el antecedente y el consecuente de la primera regla. Cada regla es una fila de gráficos, cada columna es una variable. El número de reglas es mostrado a la izquierda de cada línea. Se puede dar click al número de la regla para ver la regla en la línea de estado.

- Las primeras dos columnas de gráficas (las 18 amarillas) muestran las funciones miembro referenciadas por el antecedente, o la parte “if” (condicional si) de cada regla.
- La tercera columna de gráficas (las azules) muestra las funciones miembro referenciadas por el consecuente, o la parte “then” (entonces) de cada regla.

- La última gráfica de la tercera columna representa la decisión de la suma ponderada para el sistema de inferencia dado.

  Esta decisión dependerá de los valores de entrada para el sistema. La salida defusificada es mostrada como una línea vertical resaltada en esta gráfica.

Las variables y sus valores son mostrados en la parte de arriba de las columnas. En la parte baja a la izquierda, hay un campo de texto Input en el cual se pueden introducir los valores específicos. También se pueden cambiar los valores moviendo las líneas rojas verticales, luego de hacer esto se muestra en la tercera columna el nuevo cálculo realizado, y se puede ver todo el proceso de inferencia difuso tomando lugar.

El Rule Viewer muestra un cálculo a un tiempo y en gran detalle. En este sentido, lo representa una especie de micro vista del sistema de inferencia difuso. Si se desea ver la superficie de la salida completa del sistema se debe abrir el Surface Viewer. Este visor es la última de las cinco herramientas de la caja de herramientas GUI. Para abrir el Surface Viewer, seleccionar Surface desde el menú View.

3.1.5. The Surface Viewer.

Al abrir el Surface Viewer, se verá la curva tridimensional que representa el mapeo desde la distancia y la velocidad para ponderar la Velocidad de los motores. Debido a esta gráfica representa el caso de dos entradas y una salida, se puede ver la entrada en mapeo entero en una sola gráfica. Cuando se mueve sobre más de tres dimensiones, generalmente, se encuentran problemas para mostrar los resultados.

En consecuencia, es Surface Viewer está equipado con menús desplegables X (input): Y (input): y Z (output): los que permiten seleccionar una de las dos entradas u la salida a graficar. Sobre estos menús están dos campos de entrada X grids: e Y grids: los que permiten especificar cuantas líneas de cuadricula se desean representar para el eje “x” y el eje “y”. Esta capacidad le permite mantener un tiempo de cálculo razonable para problemas complejos.

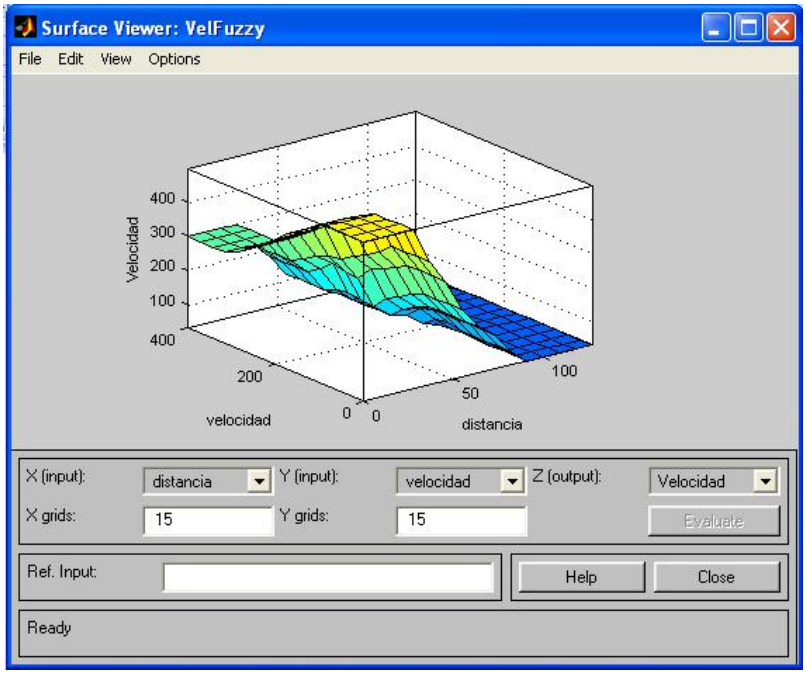


Figura 144.curva tridimensional en el Surface Viewer

Esta herramienta permite observar la gráfica desde el ángulo deseado. En caso de tener un sistema de cuatro entradas y una salida el Surface Viewer detiene dos de las entradas para mostrar el sistema en forma tridimensional ya que las pantallas de los computadores no pueden mostrar gráficos en cuatro o cinco dimensiones.

Los ítems del menú permiten abrir, cerrar, guardar y editar el sistema difuso utilizando las cinco herramientas básicas GUI. Se puede acceder a la información acerca del Surface Viewer dando click en Help y cerrar usando Close.

3.1.6. Importing and Exporting from the GUI Tools.

Cuando se guarda un sistema difuso en un archivo, se está guardando una representación de un FIS en texto ASCII, el nombre del archivo con un sufijo .fis. Este archivo de texto puede ser editado y modificado por un editor de texto común como el “notepad.exe”, de Windows, para un procesamiento sencillo del archivo. Cuando se guarda un sistema en el workspace de MATLA, se está creando una variable (del nombre que se ha escogido) la cual, actúa como una estructura de

MATLAB para el sistema FIS. Los archivos FIS y las estructuras FIS representan el mismo sistema.

Nota: Si no se guarda el FIS como archivo, pero solo se guarda en el workspace de MATLAB, no se puede recuperar para sus usos en una nueva sesión de MATLAB.

ANEXO C

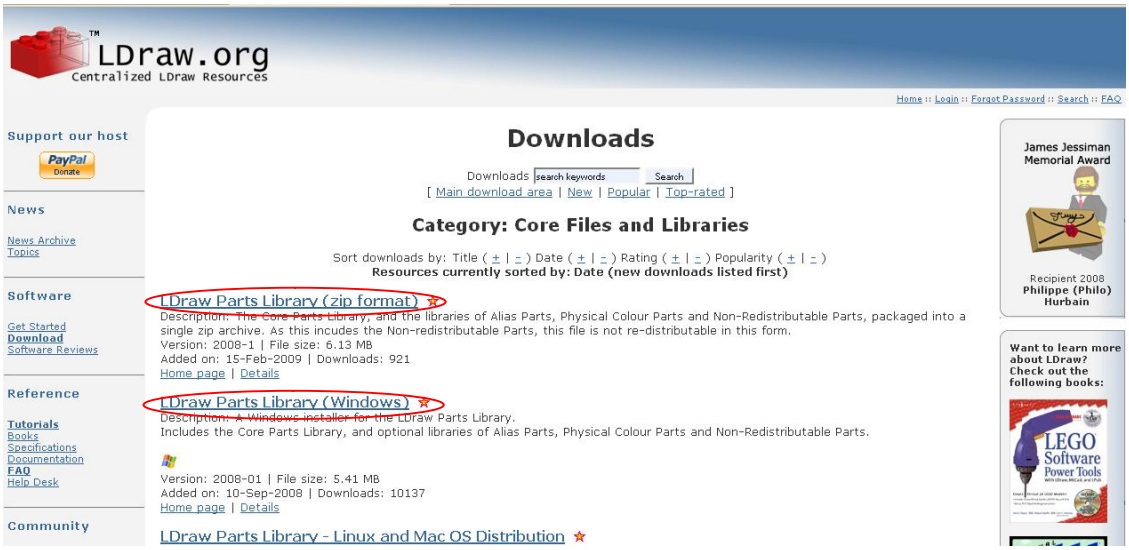
Instalación MLCad para Windows.

Antes de comenzar la instalación es necesario aclarar que para instalar este programa se debe tener Windows 98, ME, NT versión 4, XP o Vista.

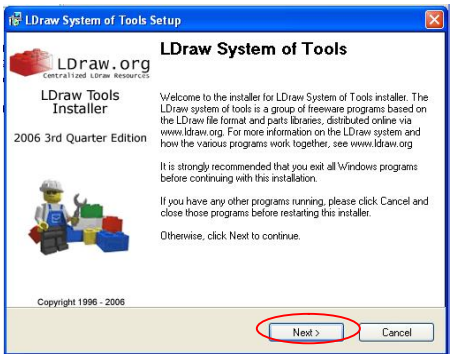
1. El primer paso para instalar MLCad es descargar LDraw en el siguiente vínculo:

http://www.ldraw.org/Downloads-req-viewdownload-cid-1.html

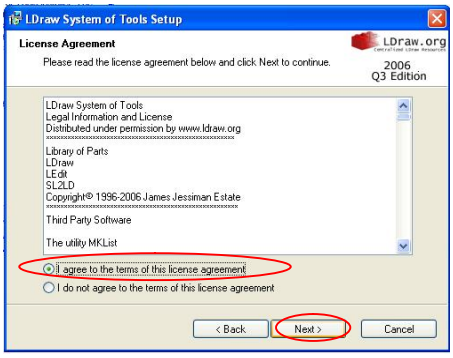
Se encontrará con la siguiente pantalla, allí podrá descargar cualquiera de los dos links resaltados. Se recomienda escoger el link LDraw Parts Library (Windows).

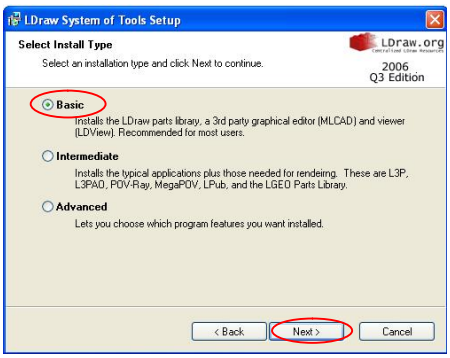


2. Ejecutar el instalador LDraw Parts al ejecutarlo aparecerá la siguiente pantalla. Escoja la opción subrayada.

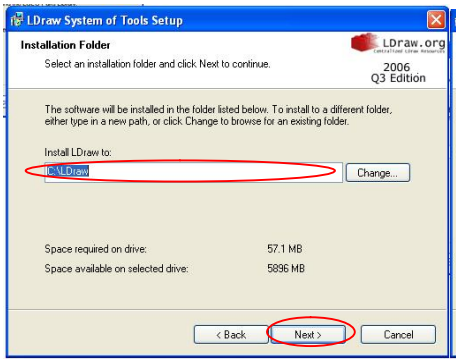


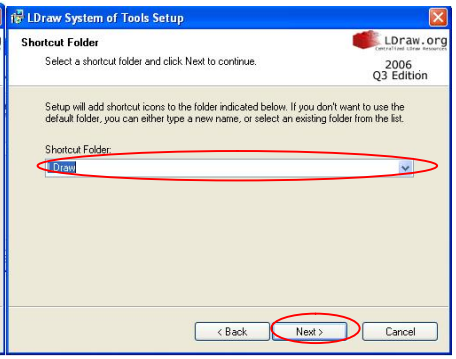
3. Seguir los pasos señalados:

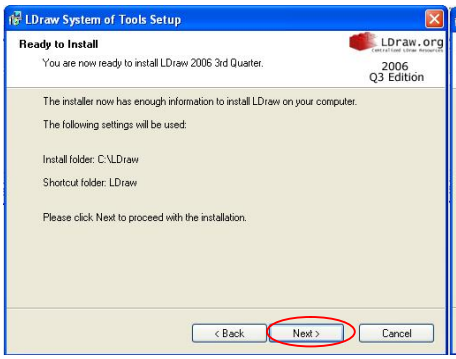


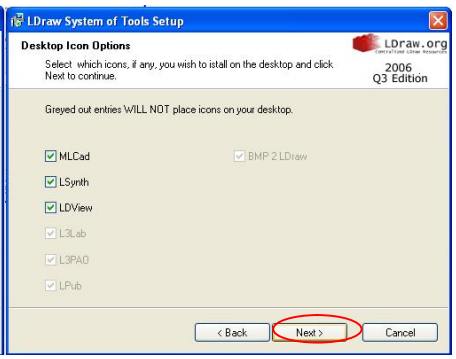


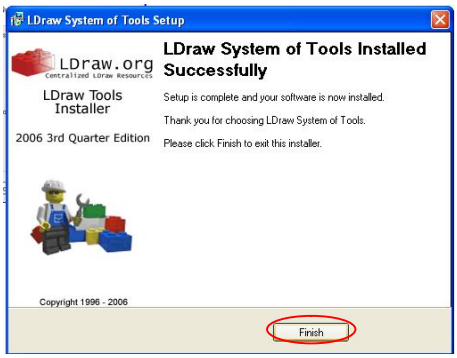
4. Escoger ubicación y seguir los pasos a continuación







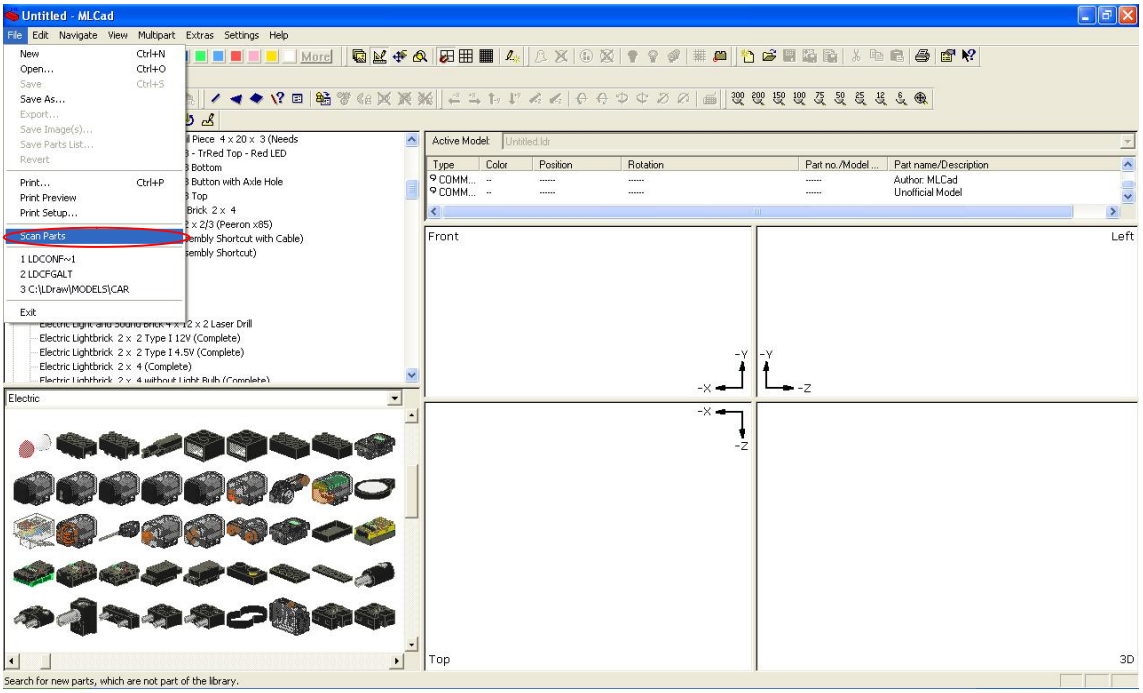


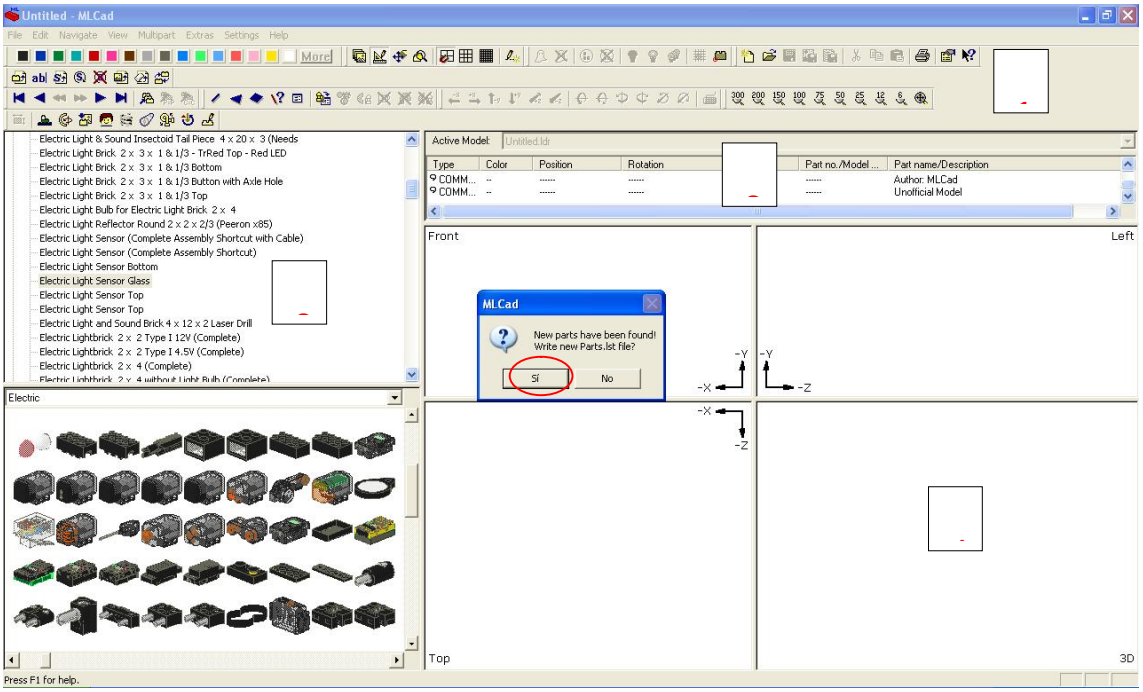


5. Ejecutar MLCAD.exe, se muestra la siguiente pantalla o muy parecida.

Seleccionar carpeta de ubicación, puede seleccionar la misma carpeta contenedora del LDraw.

6. El último paso para la instalación es ejecutar el programa (MLCad) y en la barra de herramientas seleccionar la opción Scan Parts en Inicio, de la siguiente forma:





7. Instalado el programa procede a realizar el diseño. Para esto necesario conocer la pantalla principal de MLCad:

- En esta parte (1) encontraremos diferentes menús y barras de herramientas para rotar el modelo, generar pasos de construcción, aumentar el tamaño del modelo, dar colores a las piezas. Es en esta parte donde se encuentran los elementos necesarios para el diseño virtual.

- En esta ventana (2) se encuentra una lista de nombres con todas las piezas necesarias para el modelo. En la parte de debajo de esta ventana se pueden observar gráficamente las piezas.

- En esta ventana (3) se genera un listado de todas las piezas que se están utilizando, es muy importante tener un orden en el listado separados por (step) ya que esto ayuda a construir un paso a paso más organizado.

- En esta ventana (4) se puede observar el diseño desde diferentes vista izquierda, derecha, arriba y 3D. La vista puede ser modificado dando clic derecho en la ventana y seleccionando la opción View Angle y escoger la opción deseada.

ANEXO D

Conocimientos básicos de Java

Java es un lenguaje de programación orientada a objetos. Si no se tiene conocimiento a cerca de la programación orientada a objetos se necesita aprender algunos conocimientos básicos antes de escribir cualquier código. A continuación se describen algunos conceptos básicos de la programación orientada a objetos.

1. ¿Qué es un objeto?

Un objeto es un software agrupado de comportamiento (método) y estado (campo) relacionado. Los objetos de software son frecuentemente usados para modelar los objetos del mundo real que se encuentran en la cotidianidad. Como por ejemplo cualquier cosa a su alrededor una llave, una puerta, un escritorio, una bicicleta.

2. ¿Qué es un método?

Los métodos son acciones, funciones o procedimientos que realiza el programa y operan sobre el estado interno del objeto, además definen la interacción de los objetos con el mundo exterior. Generalmente la declaración de un método tiene los siguientes componentes:

- Modificadores, como public, static, private, etc.
- Tipo de retorno: tipo de dato o valor devuelto por el método, o void cuando no devuelve un valor.
- Nombre del método.
- Lista de parámetros en paréntesis.
- Cuerpo del método entre llaves.

Ejemplo:

```
public static double areaDeCuadrado(double l){……}
```

3. ¿Qué es una clase?

Una clase es una agrupación de datos (variables o campos) y de funciones (métodos) que operan sobre esos datos. La definición de una clase se realiza de la siguiente forma:

```
[public] class Classname{

//definición de variables y métodos

    ...

}
```

Donde la palabra public es opcional; si no se pone, la clase tiene la visibilidad por defecto, es decir que sólo es visible para las demás clases del package Todos los métodos y variables deben ser definidos dentro del bloque {…} de la clase.

3.1. Características de las clases

- Todas las variables y funciones de Java deben pertenecer a una clase. No hay variables y funciones globales.
- Si una clase deriva de otra (extends), hereda todas sus variables y métodos.
- Java tiene una jerarquía de clases estándar de la que pueden derivar las clases que crean los usuarios.
- Una clase sólo puede heredar una única clase (en Java no hay herencia múltiple). Si al definir una clase no se especifica de que clase deriva, por defecto la clase deriva de Object. La clase Object es la base de toda la jerarquía de clases de Java.
- En un fichero se pueden definir varias clases, pero en un fichero no puede haber más de una clase public. Este fichero se debe llamar como la clase public que contiene con extensión *.java. Con algunas excepciones, lo habitual es escribir una sola clase por fichero.

- Si una clase contenida en un fichero no es public, no es necesario que el fichero se llame como la clase.
- Los métodos de una clase pueden referirse de modo global al objeto de esa clase al que se aplican por medio de la referencia this.
- Las clases se pueden agrupar en packages, introduciendo una línea al comienzo del fichero (package packageName;). Esta agrupación en packages está relacionada con la jerarquía de directorios y ficheros en la que se guardan las clases.

4. ¿Qué es instanciar?

Es el proceso de crear un nuevo objeto a partir de una clase. Cada objeto creado a partir de una clase tiene sus propias copias de variables de instancia, es decir que tienen espacios de memoria distintos a las variables de la clase. En el siguiente ejemplo se muestra la estructura:

Alumno alumno1 = new Alumno ();

Nombre de la clase, objeto nuevo símbolo de asignación (=), la palabra reservada new y el constructor de la clase.

Si se desea que alguna variable tenga el mismo valor para todos los objetos que se instancien de una clase se debe declarar como estática (static).

5. ¿Qué es herencia?

Las diferentes tipos de objetos frecuentemente tienen ciertas características en común con otros. Por ejemplo supongamos que tenemos diferentes tipos de bicicletas de montaña, bicicletas de camino, bicicletas de deportes extremos, por ejemplo, todas comparten las características de las bicicletas (propia velocidad, propia cadencia de pedal, propios cambios). Sin embargo cada una define características adicionales que las hacen diferentes, así en programación orientada objetos, se tiene un objeto general llamado bicicleta, de los cuales se desprenden

otros objetos, que heredan estas características generales, y agregan otras más, como son las bicicletas de montaña, aparte de tener pedales llantas, normales, tienen cambios, amortiguadores, etc. La herencia permite crear una nueva clase a partir de la definición de una clase ya existente.

6. ¿Qué es una interface?

Una interface es un conjunto de declaraciones de funciones. Si una clase implementa (implements) una interface, debe definir todas las funciones especificadas por la interface. Las interfaces pueden definir también variables finales (constantes). Una clase puede implementar más de una interface, representando una alternativa a la herencia múltiple.

Una interface puede derivar de otra o incluso de varias interfaces, en cuyo caso incorpora las declaraciones de todos los métodos de las interfaces de las que deriva (a diferencia de las clases, las interfaces en Java sí tienen herencia múltiple).

7. ¿Qué es un paquete?

Un paquete es un espacio de nombres que organiza un conjunto de clases e interfaces. Es comparable a las carpetas en un computador.

8. ¿Qué es una variable?

Es donde se almacena el estado (campo) de un objeto. Java define el siguiente tipo de variables:

- Variables de Instancia (Campos no-estáticos): su valor es único para cada instancia de una clase (para cada objeto en otras palabras).
- Variables de clase (Campos estáticos): una variable de clase es cualquier campo declarado con el modificador static, significa que el valor de la variable perdurará en el tiempo, es lo equivalente en c a variables globales.
- Variables locales: sirven para que el método pueda almacenar su estado temporalmente y van dentro de las llaves que definen al método.

- Parámetros: los parámetros que requiere el método para su ejecución. Son valores que se utilizan en el código que componen el cuerpo del método. Los parámetros aparecen encerrados en paréntesis.

Si desea adquirir más conocimientos puede acceder a los siguientes links:

- Link de la página de sun

  http://java.sun.com/docs/books/tutorial/getStarted/index.html

  http://www.tecnun.es/asignaturas/Informat1/ayudainf/aprendainf/Java/Java2.pdf

ANEXO E

ANEXO F

Encuesta de expectativas "Robótica y programación". Programa de Ingeniería Electrónica

Fecha: _____
Nombre del encuestado: _____

1- ¿Cuál es tu expectativa para con la materia de robótica y programación?

Marque con una x, respuesta única

A)	Construir un robot	
B)	Por lo menos construir un robot	
C)	Aprender las teorías de la robótica	
D)	Porque necitas los fundamentos para tu desarrollo profesional	
E)	Quieres ver de qué se trata la materia	
F)	Otros ¿Cuál?	

2- ¿Cuánto te sientes preparado para afrontar la materia de Robótica y Programación?

A)	Mucho	
B)	Poco	
C)	Nada	

3- ¿Es la Robótica el área de la electrónica en la que te gustaría hacer énfasis en tu carrera profesional?

A)	Si	
B)	No	

4- ¿Crees que el área de la robótica ofrece oportunidades laborales en la región?

A)	Mucho	
B)	Poco	
C)	Nada	

5- ¿Crees que necesitarás los conocimientos del área de robótica para desempeñarte como profesional?

A)	Mucho	
B)	Poco	
C)	Nada	

6- ¿Cuánto crees que la Institución te motiva a adentrarte en los conocimientos de la robótica?

A)	Mucho	
B)	Poco	
C)	Nada	

7- ¿Cuánto crees que tienes fundamentados los conocimientos necesarios para realizar una aplicación robótica?

A)	Mucho	
B)	Poco	
C)	Nada	

8- ¿Cuánto conocimiento tienes acerca del área de la Robótica?

A)	Mucho	
B)	Poco	
C)	Nada	

9- ¿Sabes que es LEGO?

A)	Si	
B)	No	

10- ¿Sabes que es Lego Mindstorm o Mindstorm NXT?

A)	Si	
B)	No	

11- ¿Cuánto de los avances de la tecnología robótica has podido percibir ya sea por revistas, TV, internet u otros?

A)	Mucho	
B)	Poco	
C)	Nada	

12- En caso de haber respondido mucho en la pregunta 11-, ¿Cuánto crees que se encuentra la institución actualizada en el campo de la robótica a nivel mundial?

A)	Actualizada	
B)	Relativamente atrasada	
C)	Atrasada	

13- ¿Conoces de algún dispositivo robótico "pertenecientes" a la institución?

A)	Si	¿Cuál?
B)	No	

14- ¿Conoces de algún desarrollo o aplicación robótica que se haya hecho en la institución?

A)	Si	¿Cuál?
B)	No	

15- ¿Conoces algún lenguaje de programación?, Mencione cuales.

A)	
B)	
C)	
D)	
E)	

16- Enumere una lista de los 5 conocimientos básicos que cree que son necesarios para la robótica:

A)	
B)	
C)	
D)	
E)	

17- Conoces alguna técnica para el control de Robots. ¿Cuáles?

A)	
B)	
C)	
D)	
E)	

18- Enumere las tres leyes de Asimov:

A)	
B)	
C)	

19- Asocias Robótica con:

A)	Humanoides estilo Robocops	
B)	Otros	¿Cuáles?

20- En conclusión , crees que puedas crear una aplicación de robótica, y para ello consideras que :

A)	Solo Y solo si tengo guía total de: Manuales, profesores, ejemplos.
B)	Con alguna ayuda, sea de manuales o de profesores, y las herramientas de hardware y software.
C)	Con ayuda de los manuales me es suficiente y las herramientas me es suficiente..
D)	Sin ninguna ayuda, solo necesito las herramientas (hardware y software)
E)	Sin ninguna ayuda, y puedo construir y diseñar totalmente todo, no necesito ninguna herramienta aparte de mi conocimiento y creatividad.

Resultados encuesta

El día 12 de febrero del 2010, fueron distribuidos 17 formularios, entre los estudiantes de 8 y 9 semestre de ingeniería Electrónica, estos formularios fueron llenados en su totalidad a excepción de algunas preguntas no contestadas.

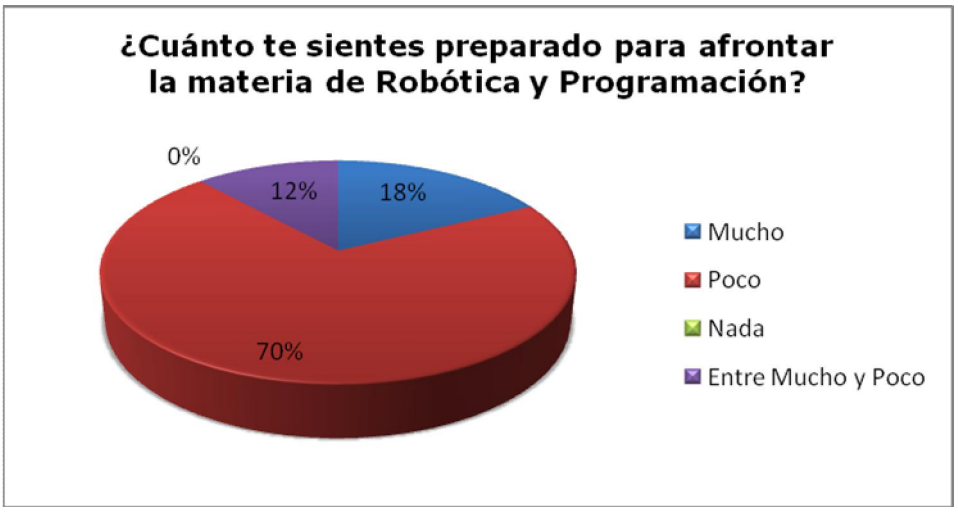
En las siguientes tablas y gráficos observamos la información recopilada:

1. ¿Cuál es tu expectativa para con la materia de robótica y programación?

	Cantidad	%
Construir un robot	5	29
Aprender las teorías de la robótica	7	41
Porque necesitas los fundamentos para tu desarrollo profesional	4	24
Quieres ver de qué se trata la materia	1	6
Otros	0	0
Total	17	100

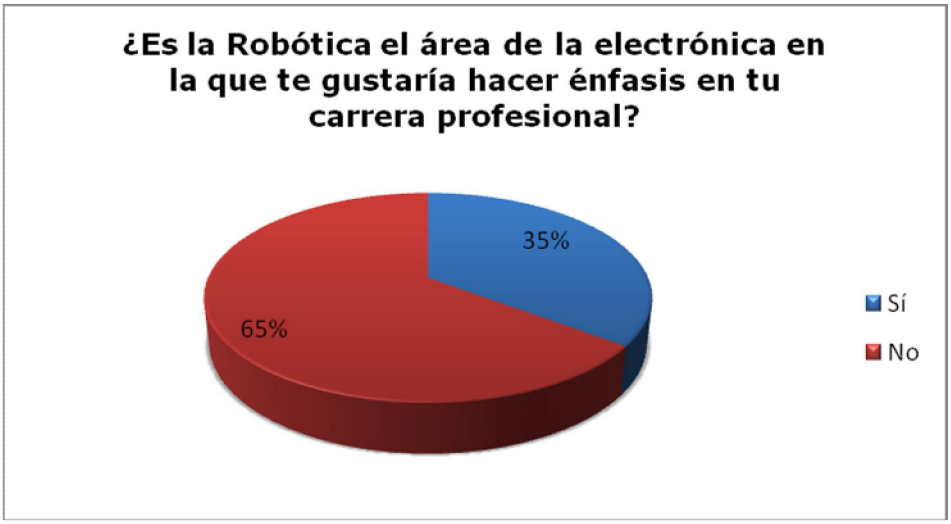
2. ¿Cuánto te sientes preparado para afrontar la materia de Robótica y Programación?

	Cantidad	%
Mucho	3	18
Poco	12	70
Nada	0	0
Entre Mucho y Poco	2	12
Total	17	100



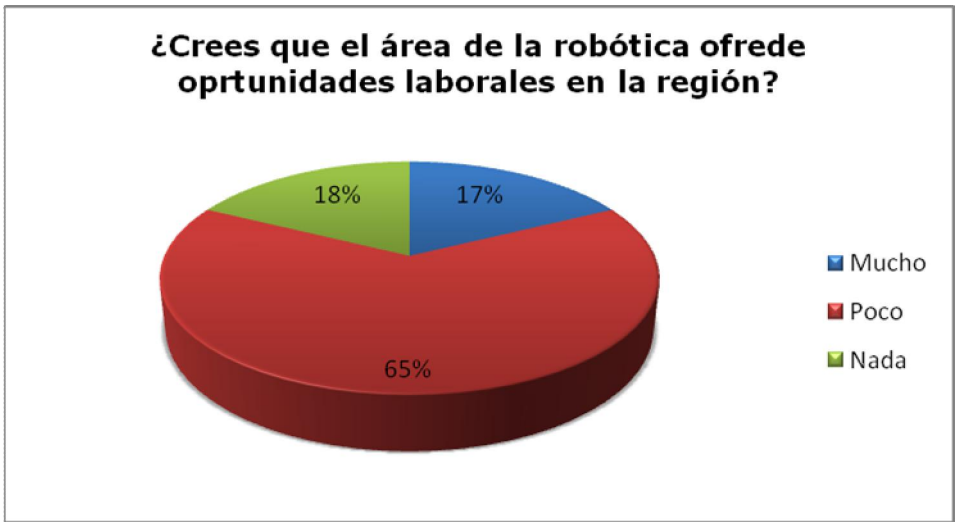
3. ¿Es la Robótica el área de la electrónica en la que te gustaría hacer énfasis en tu carrera profesional?

	Cantidad	%
Sí	6	35
No	11	65
Total	17	100



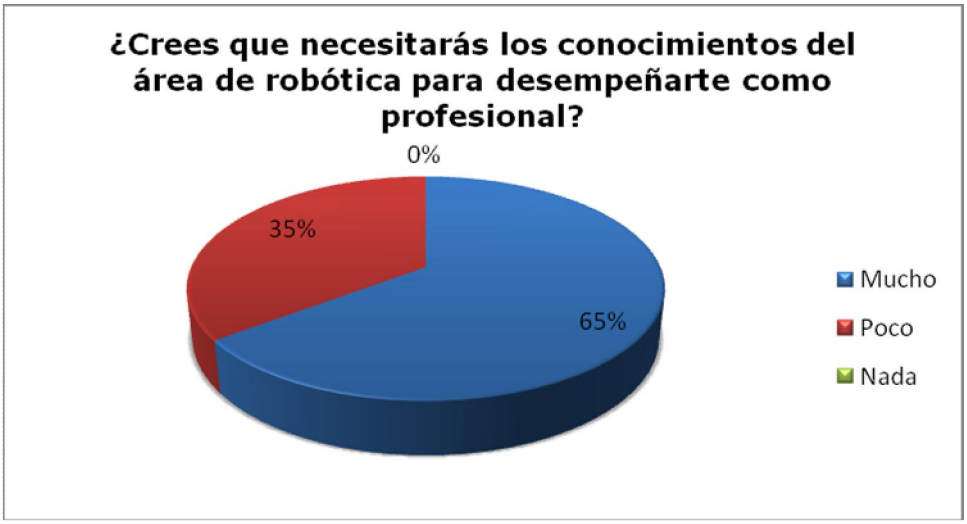
4. ¿Crees que el área de la robótica ofrece oportunidades laborales en la región?

	Cantidad	%
Mucho	3	17
Poco	11	65
Nada	3	18
Total	17	100



5. ¿Crees que necesitarás los conocimientos del área de robótica para desempeñarte como profesional?

	Cantidad	%
Mucho	11	65
Poco	6	35
Nada	0	0
Total	17	100

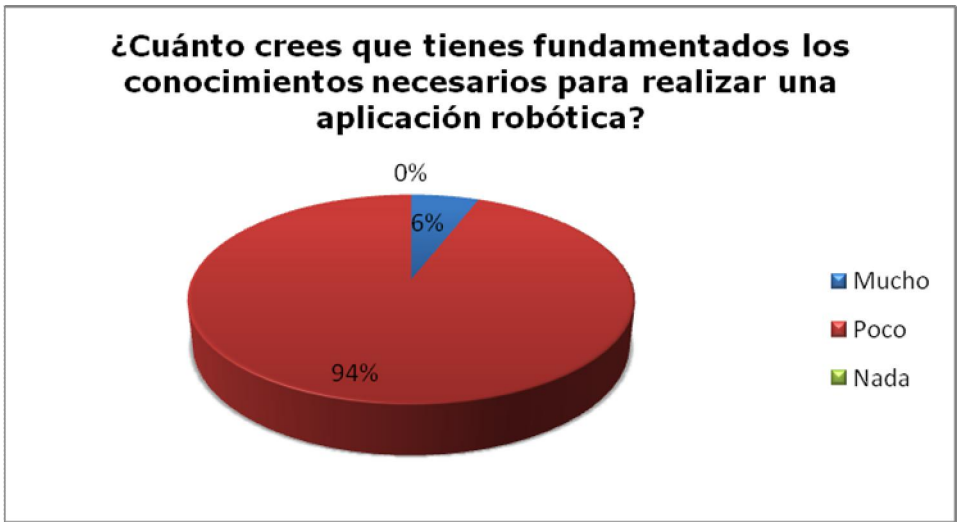


6. ¿Cuánto crees que la Institución te motiva a adentrarte en los conocimientos de la robótica?

	Cantidad	%
Mucho	2	12
Poco	12	70
Nada	3	18
Total	17	100

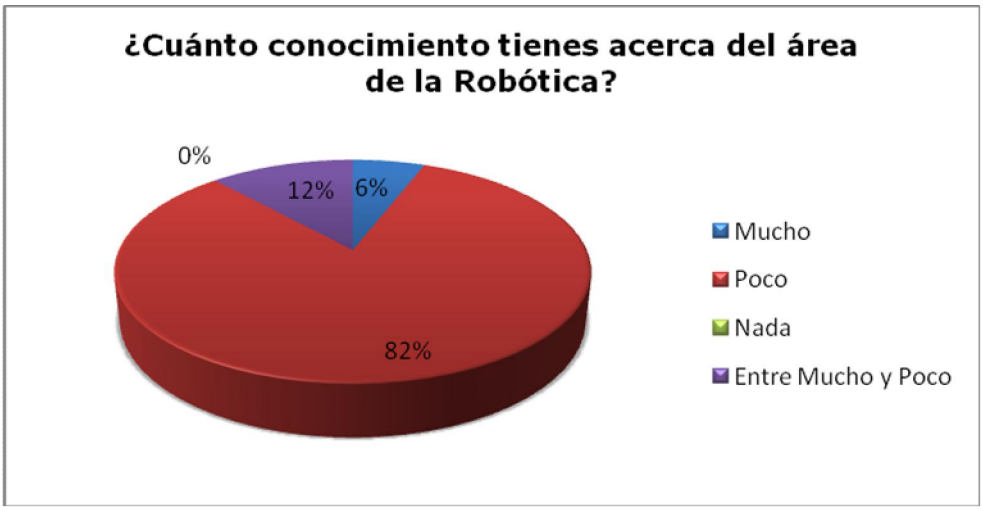
7. ¿Cuánto crees que tienes fundamentados los conocimientos necesarios para realizar una aplicación robótica?

	Cantidad	%
Mucho	1	6
Poco	16	94
Nada	0	0
Total	17	100



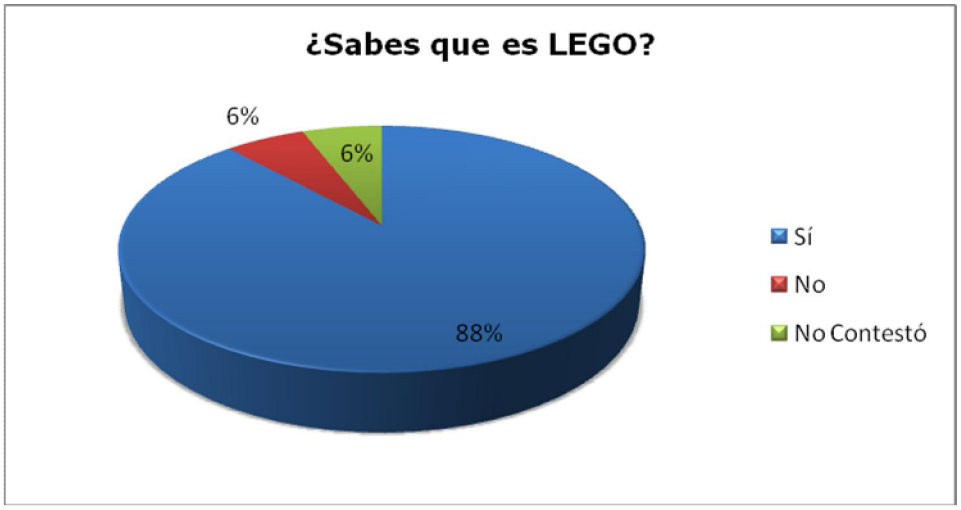
8. ¿Cuánto conocimiento tienes acerca del área de la Robótica?

	Cantidad	%
Mucho	1	6
Poco	14	82
Nada	0	0
Entre Mucho y Poco	2	12
Total	17	100



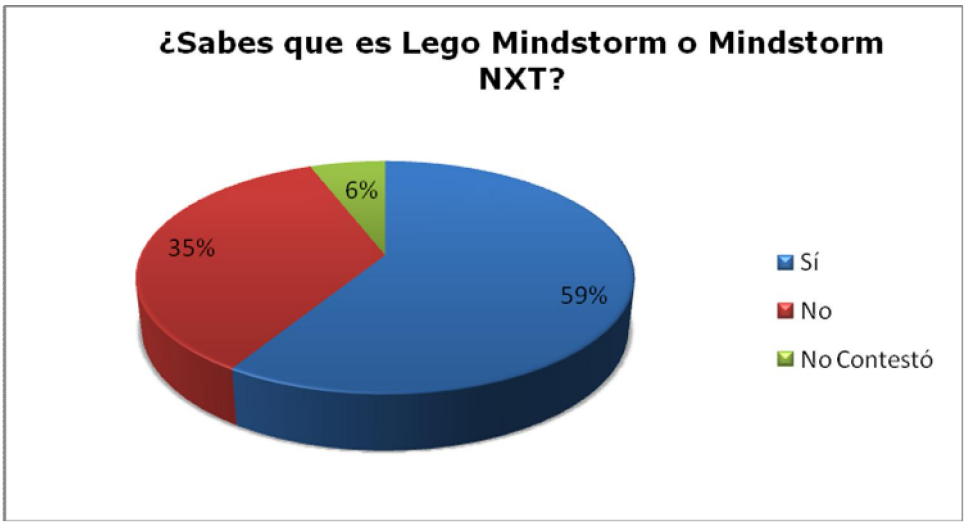
9. ¿Sabes que es LEGO?

	Cantidad	%
Sí	15	88
No	1	6
No Contestó	1	6
Total	17	100



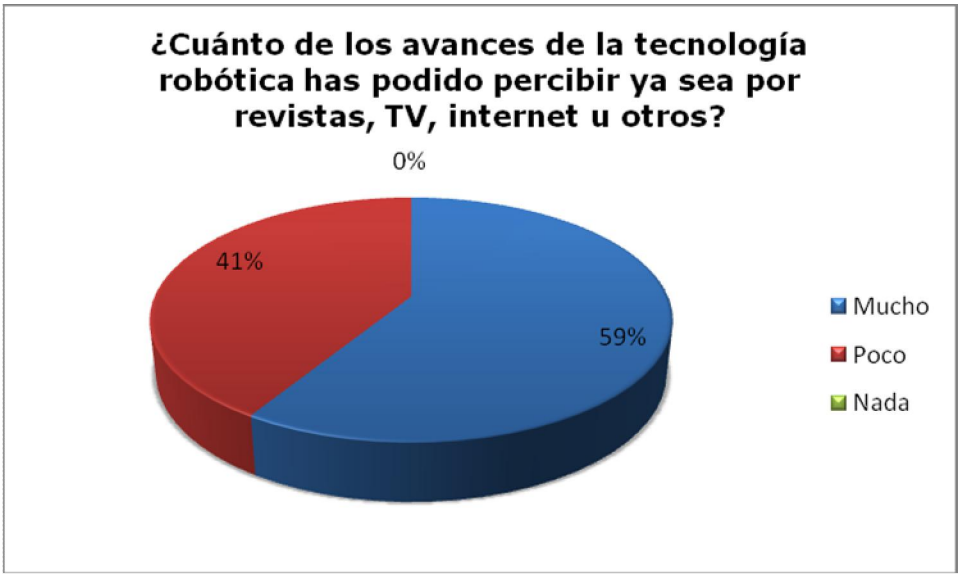
10. ¿Sabes que es Lego Mindstorm o Mindstorm NXT?

	Cantidad	%
Sí	10	59
No	6	35
No Contestó	1	6
Total	17	100



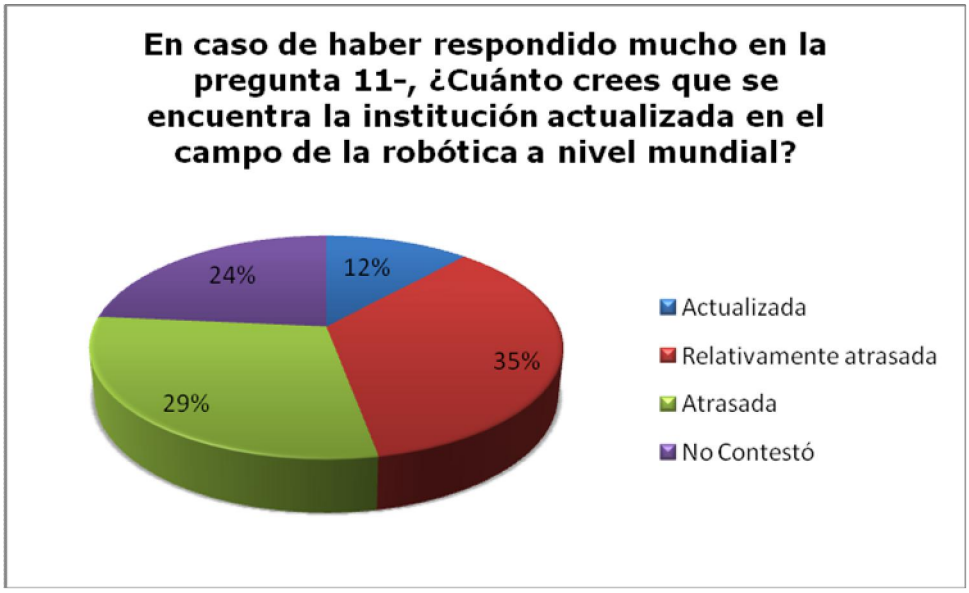
11. ¿Cuánto de los avances de la tecnología robótica has podido percibir ya sea por revistas, TV, internet u otros?

	Cantidad	%
Mucho	10	59
Poco	7	41
Nada	0	0
Total	17	100



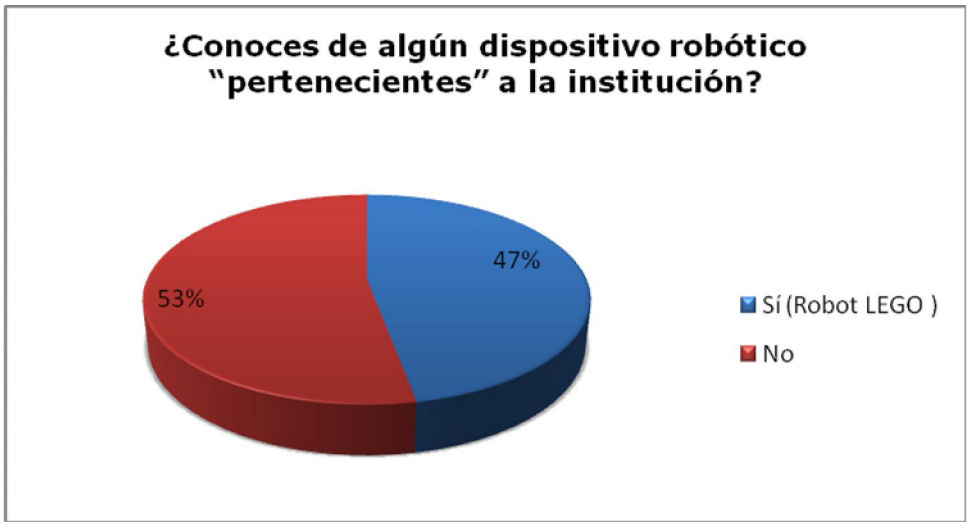
12. En caso de haber respondido mucho en la pregunta 11-, ¿Cuánto crees que se encuentra la institución actualizada en el campo de la robótica a nivel mundial?

	Cantidad	%
Actualizada	2	12
Relativamente atrasada	6	35
Atrasada	5	29
No Contestó	4	24
Total	17	100



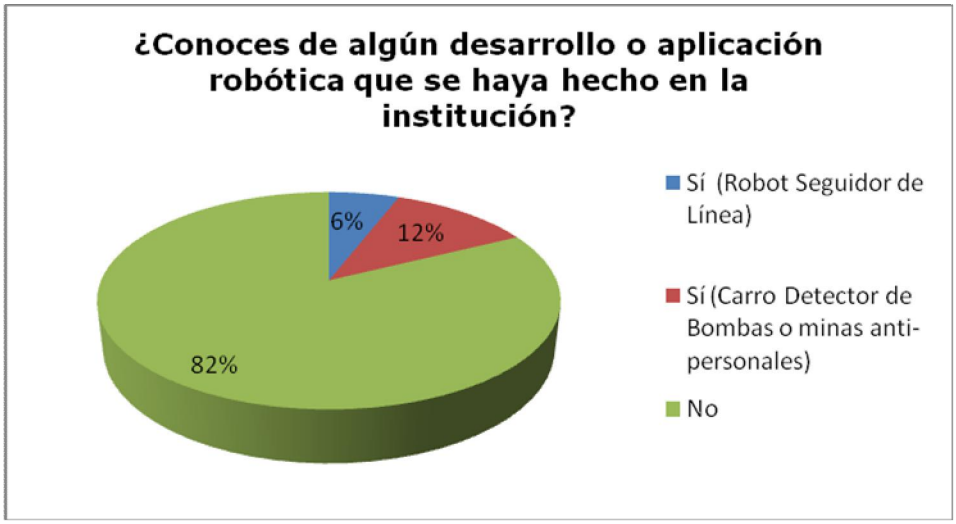
13. ¿Conoces de algún dispositivo robótico "pertenecientes" a la institución?

	Cantidad	%
Sí (Robot LEGO)	8	47
No	9	53
Total	17	100



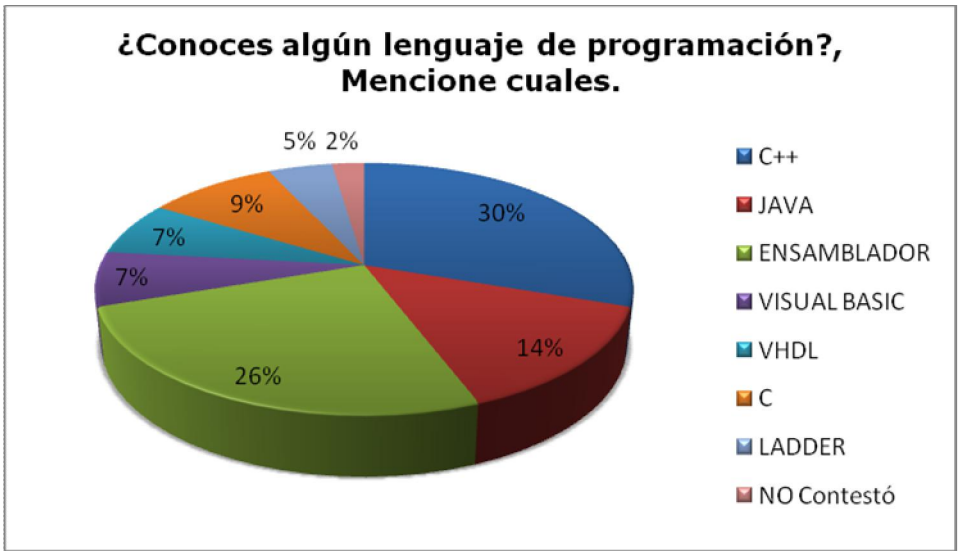
14. ¿Conoces de algún desarrollo o aplicación robótica que se haya hecho en la institución?

	Cantidad	%
Sí (Robot Seguidor de Línea)	1	6
Sí (Carro Detector de Bombas o minas anti-personales)	2	12
No	14	82
Total	17	100



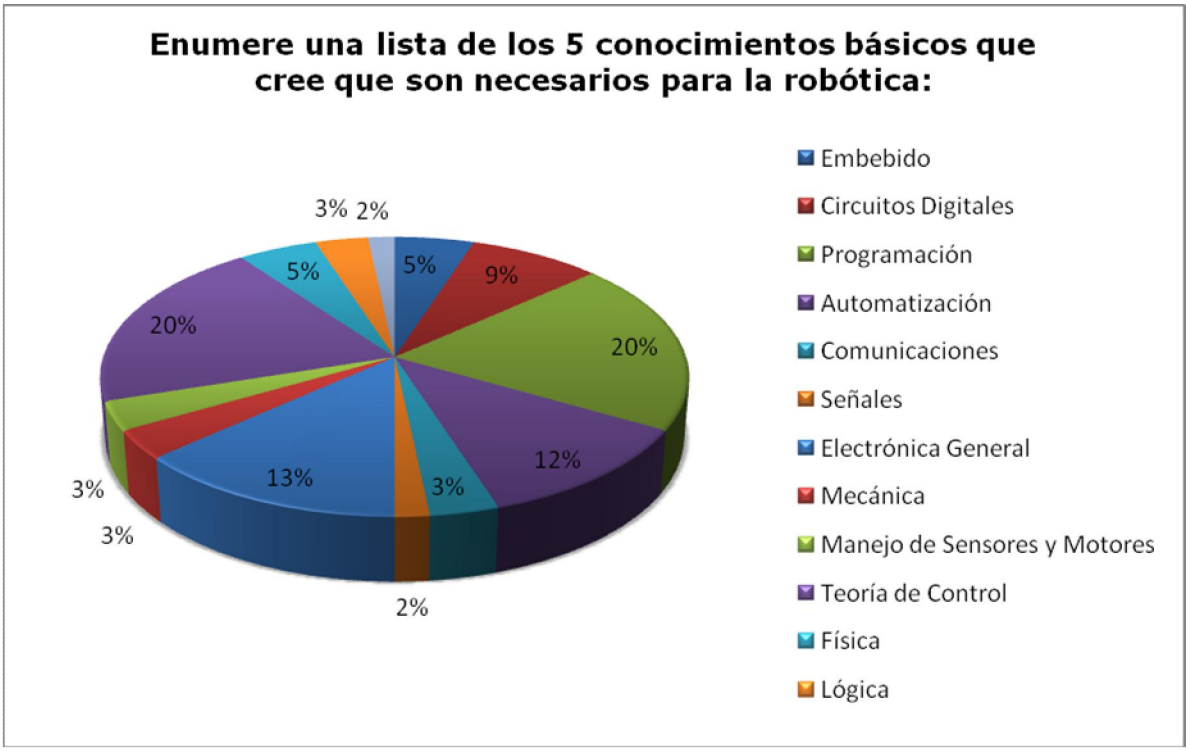
15. ¿Conoces algún lenguaje de programación?, Mencione cuales.

	Cantidad	%
C++	13	30
JAVA	6	14
ENSAMBLADOR	11	26
VISUAL BASIC	3	7
VHDL	3	7
C	4	9
LADDER	2	5
NO Contestó	1	2
Total		100



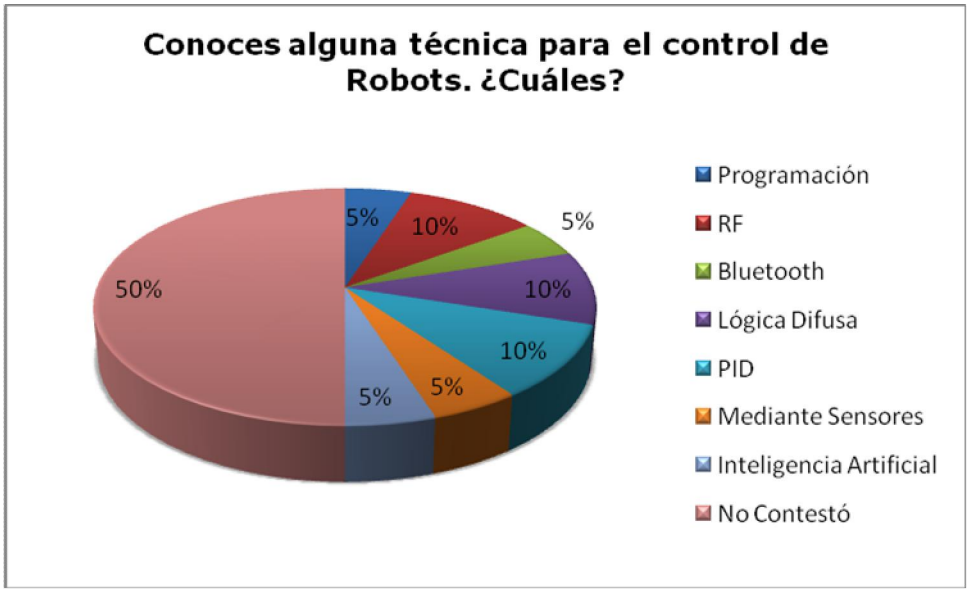
16. Enumere una lista de los 5 conocimientos básicos que cree que son necesarios para la robótica:

	Cantidad	%
Embebido	3	5
Circuitos Digitales	5	9
Programación	12	20
Automatización	7	12
Comunicaciones	2	3
Señales	1	2
Electrónica General	8	13
Mecánica	2	3
Manejo de Sensores y Motores	2	3
Teoría de Control	12	20
Física	3	5
Lógica	2	3
No Contestó	1	2
Total		100



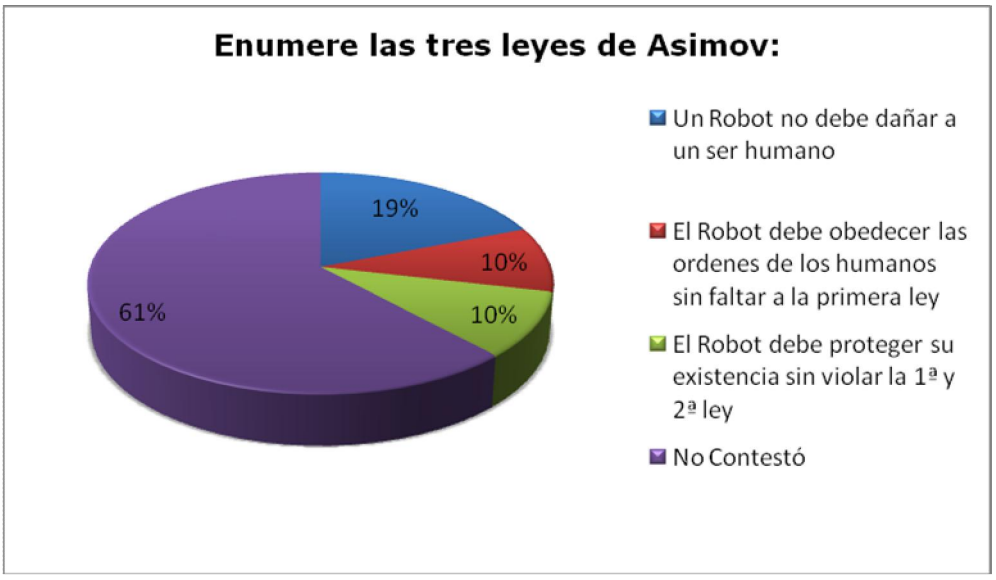
17. Conoces alguna técnica para el control de Robots. ¿Cuáles?

	Cantidad	%
Programación	1	5
RF	2	10
Bluetooth	1	5
Lógica Difusa	2	10
PID	2	10
Mediante Sensores	1	5
Inteligencia Artificial	1	5
No Contestó	10	50
Total		100



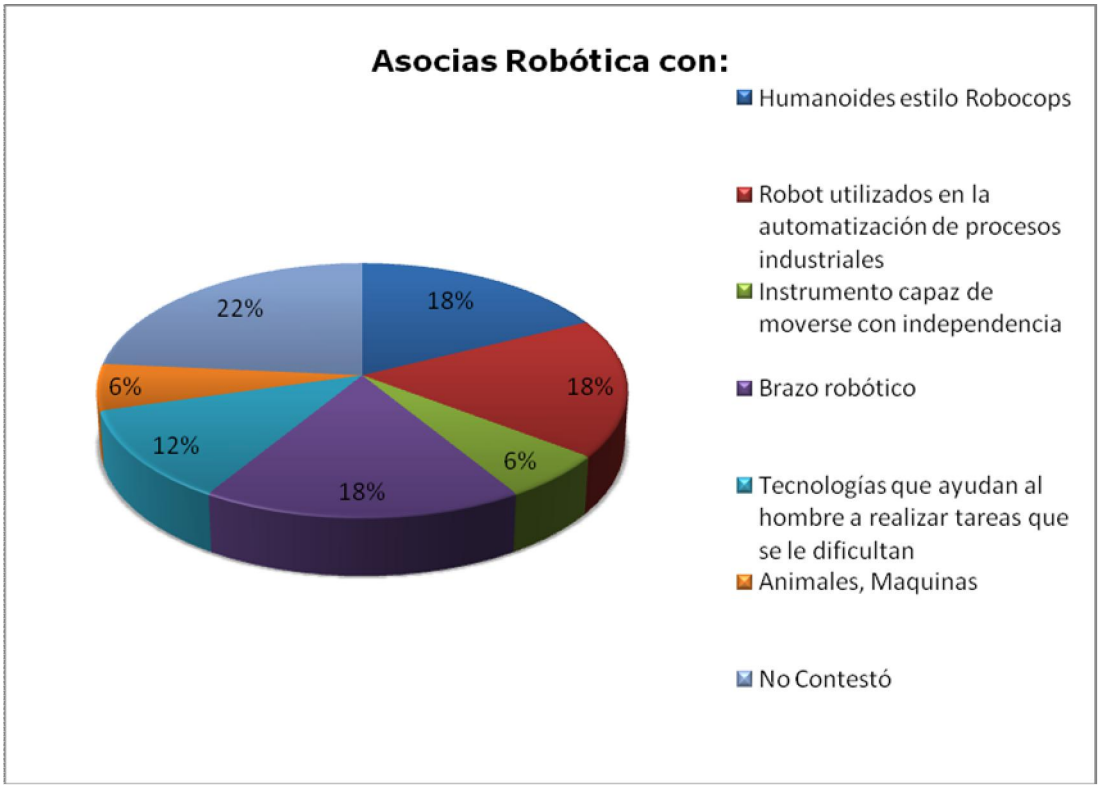
18. Enumere las tres leyes de Asimov:

	Cantidad	%
Un Robot no debe dañar a un ser humano	4	19
El Robot debe obedecer las órdenes de los humanos sin faltar a la primera ley	2	10
El Robot debe proteger su existencia sin violar la 1ª y 2ª ley	2	10
No Contestó	13	61
Total		100



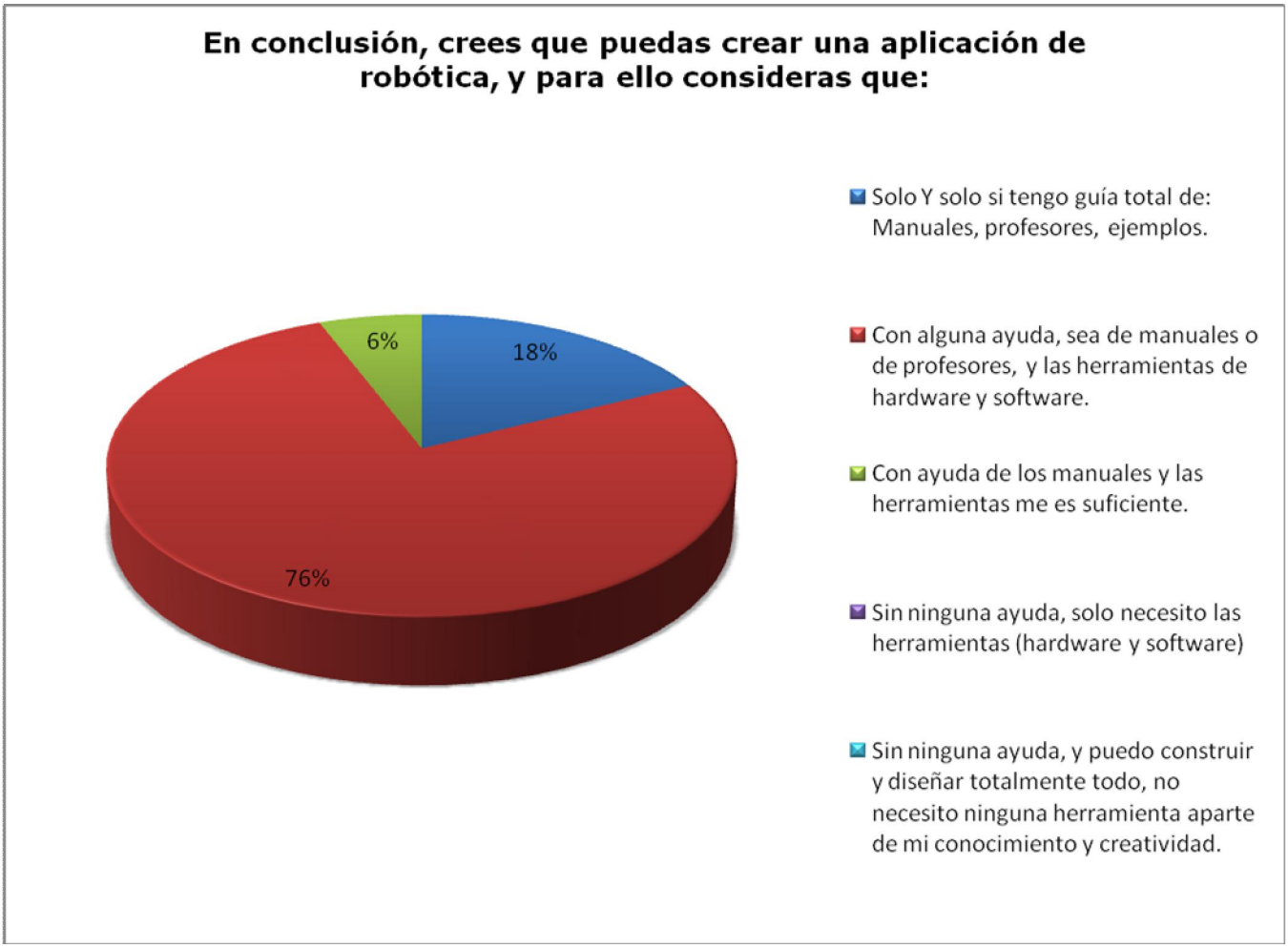
19. Asocias Robótica con:

	Cantidad	%
Humanoides estilo Robocops	3	18
Robot utilizados en la automatización de procesos industriales	3	18
Instrumento capaz de moverse con independencia	1	6
Brazo robótico	3	18
Tecnologías que ayudan al hombre a realizar tareas que se le dificultan	2	12
Animales, Maquinas	1	6
No Contestó	4	22
Total	17	100



20. En conclusión, crees que puedas crear una aplicación de robótica, y para ello consideras que:

	Cantidad	%
Solo Y solo si tengo guía total de: Manuales, profesores, ejemplos.	3	18
Con alguna ayuda, sea de manuales o de profesores, y las herramientas de hardware y software.	13	76
Con ayuda de los manuales y las herramientas me es suficiente.	1	6
Sin ninguna ayuda, solo necesito las herramientas (hardware y software)		0
Sin ninguna ayuda, y puedo construir y diseñar totalmente todo, no necesito ninguna herramienta aparte de mi conocimiento y creatividad.		0
Total	17	100



ANEXO G