UNIVERSIDAD TÈCNICA DE COTOPAXI

CARRERA DE CIENCIAS DE LA INGENIERÍA Y APLICADAS

TESIS DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERA EN INFORMÁTICA Y SISTEMAS COMPUTACIONALES

TEMA:

"EL DESARROLLO DE UN PROTOTIPO DE ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR ASISTIDO POR UN COMPUTADOR PARA LA UNIVERSIDAD TÉCNICA DE COTOPAXI".

PROMOCIÓN:

Sexta.

POSTULANTES:

Herrera Calderón Alexandra de los Ángeles.

Uribe Monga Mariela Cristina.

DIRECTOR DE TESIS:

Ing. Silva Marco Polo.

Latacunga-Ecuador

2007

AUTORÍA

Nosotras; Herrera Calderón Alexandra de los Ángeles y Uribe Monga Mariela Cristina, declaramos bajo juramento que el trabajo aquí presentado es de nuestra autoría; que no ha sido previamente presentado anteriormente; y, que hemos consultado las referencias bibliograficas que se incluyen en este documento.

Herrera Alexandra de los Ángeles.

050276960-7

Uribe Mariela Cristina.

050282022-8

CERTIFICACIÒN

HONORABLE CONSEJO ACADÉMICO DE LA UNIVERSIDAD TÉCNICA DE COTOPAXI.

De mi consideración:

Cumpliendo con lo estipulado en el capitulo IV, (art.9 literal f), del reglamento del curso profesional de la Universidad Técnica de Cotopaxi, informo que las postulantes Herrera Calderón Alexandra de los Ángeles y Uribe Monga Mariela Cristina, han desarrollado su tesis de grado de acuerdo al planteamiento formulado en el plan de tesis con el tema: "EL DESARROLLO DE UN PROTOTIPO DE ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR ASISTIDO POR UN COMPUTADOR PARA LA UNIVERSIDAD TÈCNICA DE COTOPAXI", cumpliendo sus objetivos respectivos.

En virtud de lo antes expuesto considero que la presente tesis se encuentra habilitada para presentarse al acto de la defensa de tesis.

Latacunga, 4 de Enero del 2007.

Ing. Marco Polo Silva Segovia.

Director de tesis.

AGRADECIMIENTO

Al término de nuestro trabajo investigativo, queremos dejar constancia de nuestro profundo reconocimiento y gratitud, a la Universidad Técnica de Cotopaxi que nos abrió las puertas para continuar con los estudios, en especial a nuestro maestro director de tesis Ing. Marco Polo Silva, quien con sus valiosos aportes curriculares facilitó los medios para culminar nuestro trabajo; y, a nuestros profesores que aportaron para mejorar nuestra enseñanza aprendizaje.

EL GRUPO.

DEDICATORIA

Con devoción y orgullo este esfuerzo pequeño pero de justa retribución por los cimientos que sembraron en nuestro espíritu, dedicamos a Dios, a nuestros queridos padres quienes con nobleza y entusiasmo siempre estuvieron a nuestro lado en los momentos más difíciles depositaron en nosotros su apoyo y confianza para ser útil a la sociedad. Ellos hicieron posible la culminación de una etapa importante en nuestra vida.

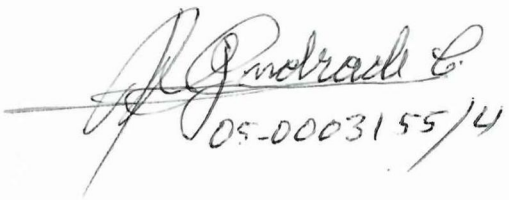
SUMMARY

The present investigation is the development of robot's prototype with capacity of speak and walking attented by a computer for the Technical University of Cotopaxi, the same one that will allow to introduce robotics applications motivating and incentivating to the students to develop technological and scientific projects.

The sequence of steps of robot's prototype is programmed in the lenguaje PBASIC, the same one that needs to be executed and stored in the memory EEPROM.
The system that the user used was designed by the language of Visual Basic 6.0, where the same one has two options the teleoperation and the teleprogramation.

The method of the teleoperation consists in that the user manipulates the arrows of direction of the keyboard depending of the action that wants to carries out robot's prototype. In what it concerns to the teleprogramation the user selected the sequence type and immediately the system Hill request the entrante of the number of steps that it Hill be executed.

The system also possesses a special tool of Microsoft the same one that helps to that it can speak robot's prototype specifying the entrence instructions to the system.



RESUMEN

La presente investigación se trata del desarrollo de un prototipo de robot con capacidad de hablar y caminar asistido por un computador para la Universidad Técnica de Cotopaxi, la misma que permitirá introducir aplicaciones de robótica motivando e incentivando a los estudiantes a desarrollar proyectos tecnológicos y científicos.

La secuencia de pasos del prototipo de robot está programada en el lenguaje PBASIC, la misma que para ser ejecutada necesita ser enviada y almacenada en la memoria EEPROM.
El sistema que utilizara el usuario está diseñado mediante el lenguaje de Visual Basic 6.0, donde el mismo tiene dos opciones la teleoperación y la teleprogramación.

El método de la teleoperación consiste en que el usuario manipule las flechas de direccionamiento del teclado dependiendo la acción que desee realice el prototipo de robot. En lo que respecta a la teleprogramación el usuario seleccionara el tipo de secuencia e inmediatamente el sistema pedirá el ingreso del número de pasos que se ejecutara.

Además el sistema posee una herramienta especial de Microsoft la misma que ayuda a que pueda hablar el prototipo de robot especificando las instrucciones de ingreso al sistema.

ÍNDICE GENERAL

CAPÍTULO II

DETERMINACIÓN DE LOS REQUISITOS

CAPÍTULO III

FASE DE DISEÑO

ÍNDICE DE TABLAS

CONTENIDO **Pág.**

ÍNDICE DE FIGURAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

INTRODUCCIÒN

La investigación, el desarrollo de la robótica tiene énfasis, sobre todo en la industria moderna de alta tecnología, ha dado origen a una asociación cada vez más estrecha entre humanos y empresa. En la actualidad, es preocupación permanente de los estudiantes de nivel superior asumir un papel protagónico y activo en el desarrollo de la informática por ser una de la especialidad más compleja de la ingeniería que existe hoy en día, se involucra tres principios en áreas diferentes que son vitales para gestionar proyectos que permiten diseñar y construir robots de éxito, que es la Informática, la Electrónica, y el Diseño Estructural.

La investigación y el desarrollo tendrán un rol predominante que irá incrementando y permitiendo una vida mejor, uno de los principales campos en que la robótica brinda apoyo a la educación, nace como una disciplina aleada con la informática educativa que tiene como finalidad extender las habilidades intelectuales de los estudiantes apoyando la calidad de su aprendizaje.

La vinculación efectiva de la robótica y la educación favorece, fortalece su desarrollo, requiere de la capacidad de detectar áreas estratégicas para el futuro de las universidades; la realización del proyecto posee un grado de complejidad, será necesario contar con laboratorios de electrónica e informática en cambio para su demostración solo se requerirá de maquinas básicas que están al alcance de toda Institución, como es el caso de una computadora.

A través de un diagnostico desarrollado en la Universidad Técnica de Cotopaxi se ha detectado que no se en encuentra involucrada, en aplicaciones científicas y tecnológicas debido a la escasa motivación por parte de alumnos que estudian en dicha institución, la creación de un prototipo de robot incentivará a todos los estudiantes de las carreras técnicas para que realicen aplicaciones de robótica.

Con lo expuesto el grupo investigador de esta tesis se propone, **"EL DESARROLLO DE UN PROTOTIPO DE ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR ASISTIDO POR UN COMPUTADOR PARA LA UNIVERSIDAD TÉCNICA DE COTOPAXI ",** el desarrollo del prototipo de robot se realizará mediante partes electrónicas en sus extremidades inferiores, se acoplará un cuerpo elemental. La presente investigación tecnológica será el complemento para que la Universidad Técnica de Cotopaxi se incluya en aplicaciones de Robótica coadyuvando al desarrollo científico y tecnológico.

Como objetivo general de la presente investigación se plantea: desarrollar un prototipo de robot con capacidad de hablar y caminar asistido por un computador, que permita introducir aplicaciones de robótica en la Universidad Técnica de Cotopaxi.

Los objetivos específicos de la investigación son los siguientes:

Determinar los requerimientos materiales, estructurales, electrónicos, y de software que nos permitan diseñar un prototipo de robot.

Diseñar un prototipo de robot con capacidad de hablar y caminar controlado por un computador.

Probar el funcionamiento del prototipo de Robot que pueda caminar y hablar asistido por un computador.

Demostrar que las Ciencias de la Informática y la Electrónica están íntimamente ligadas al punto de que el software y el hardware no pueden funcionar si no se complementan.

Se cuenta con suficiente información bibliográfica para la investigación del proyecto, es una disciplina que está en pleno auge en este siglo, el proyecto se lo realiza para incentivar a los estudiantes para que desarrollen la investigación científica y tecnológica.

Por la naturaleza de la investigación los métodos más apropiados serán el método analítico, inductivo, deductivo, científico y experimental, la recolección de información se basará en la entrevista y la observación.

El diseño de la investigación se basará en el alcance de los objetivos planteados por el grupo investigador, por cuanto el tema de tesis con lleva el diseño y la construcción de una maquinaria tecnológica basada en ciencias exactas como es la matemática, la física, la mecánica, la electrónica y la Informática.

Nuestro trabajo ha sido técnicamente diseñado en tres capítulos. El primero corresponde al conocimiento de la robótica y su morfología, el segundo se basa en la determinación de los requisitos y análisis del sistema, el tercero consta de modelos de análisis y diseño del sistema.

En cuanto al tercer capítulo o sea el último podemos afirmar que es el núcleo de todo el trabajo, toda vez que al concluir el ensamblaje se halla enfocada hacia la teleoperación y teleprogramación por parte del usuario.

Aspiramos que la comisión del tribunal se digne evaluar favorablemente la presente tesis, que como queda expresada, es producto de desvelos constantes y sacrificios múltiples.

CAPÍTULO I

CONOCIMIENTOS DE LA ROBÓTICA

1.1. ROBÓTICA

Es la rama de la inteligencia artificial que se ocupa de las máquinas inteligentes, en los años sesenta se desarrolla la robótica industrial se trata fundamentalmente de dotar de flexibilidad los procesos productivos manteniendo al mismo tiempo la productividad, esta se orienta a la manipulación y como tecnología, surge en 1960. Desde entonces hasta el día de hoy, el interés que ha despertado y el estudio del que ha sido objeto, han sido infinitamente superiores a cualquier previsión. Aunque se han logrado avances impresionantes, el campo de desarrollo e innovación que esta ciencia ofrece es tan amplio que, cada día, los investigadores plantean formas robóticas cada vez más evolucionadas. Las posibilidades parecen infinitas.

El término robot aparece por primera vez en 1921, en la obra teatral R.U.R. (Rossum's Universal Robots) del novelista y autor dramático checo Karel Capek en cuyo idioma la palabra checa "robota" significa fuerza de trabajo o servidumbre. Por aquellos años la producción en grandes series se había introducido en numerosas fábricas, se discute ya del poder de las máquinas y la dominación de los hombres por las máquinas, argumento de esto y otras obras teatrales y películas de los años veinte en los que aparecen trabajadores robóticos.

El término tiene amplia aceptación y pronto se aplica en autómatas construidos en los años veinte y treinta que se exhiben en ferias, promociones de productos, películas y otras aplicaciones, se trata de imitar movimientos de seres vivos pero también de demostrar técnicas de control remoto, incluyéndose en algunos casos funciones sensoriales primarias.

El término robot nace asociado a la idea de trabajo y producción. En 1915, Leonardo Torres Quevedo declaró a la revista "Scientific American". "Los antiguos autómatas imitaban la apariencia y movimientos de los seres vivos, lo cual no tiene mucho interés práctico; lo que se busca es una clase de aparatos que sin necesidad de reproducir los gestos más visibles del hombre intentan obtener los mismos resultados de una persona".

Los robots industriales surgen de la convergencia de tecnologías del control automático, en particular, del control de máquinas, de los manipuladores teleoperadores; y, de la aplicación de computadoras en tiempo real. Los servosistemas generan automáticamente señales de control que tratan de anular la diferencia entre la señal de consigna y la señal medida del objeto que se pretende controlar.

Tanto los servosistemas como los reguladores se basan en el principio de la realimentación, las señales de consigna o referencia se comparan con medidas de variables del objeto que se pretende controlar, su diferencia se emplea para generar acciones de control sobre el propio objeto. En los sistemas de control automático esta cadena cerrada de acción-medida-acción se realiza sin intervención del hombre.

La automatización industrial con utilización de sistemas de control automático comienza también en el siglo XIX pero no es hasta el siglo XX introducida en los sectores industriales. De esta forma se generalizan los sistemas de control automático de variables de procesos industriales, en particular sistemas de control y posición de velocidad, se emplean también sistemas de control realimentado en barcos o aviones que deben seguir automáticamente una determinada trayectoria.

Tradicionalmente, en la realización de sistemas de control automático se han empleado diversas tecnologías tales como: la neumática, hidráulica, posteriormente la eléctrica. A finales de los años sesenta y comienzos de los setenta los mini computadores encuentran una importante acogida en aplicaciones de control. En 1972 la aparición del microprocesador suministra un impulso decisivo al control por computador, haciendo rentables numerosas aplicaciones entre las que se cuenta el control de robots. Los avances en microelectrónica de los años ochenta, con la tecnología de los circuitos de gran escala de integración acentúan esta tendencia.

Con respecto a las máquinas herramientas de control numérico hay que señalar los proyectos que se desarrollan en EE.UU, a finales de los años cuarenta y principio de los cincuenta se combinaban los progresos en el diseño de servosistemas con las recientes experiencias en técnicas de computación digital, de esta forma el contorno de corte era codificado en cinta de papel perforada, utilizándose para generar automáticamente las ordenes a los servomecanismos de la maquina.

Los Teleoperadores se desarrollan en los años cuarenta para manejar materiales radioactivos que consistían en un par de pinzas "maestra" y "esclavo" acoplados por mecanismos que permitían que la pinza "esclava" en contacto con el material peligroso reprodujera los movimientos de la pinza "maestra" accionada por un operador detrás de un muro protector con ventanas apropiadas para observar la operación. El primer teleoperador accionado por servomecanismos eléctricos se presentó en 1947, poco después en 1948 se introdujeron servosistemas con realimentación de fuerza hacia la pinza "maestra" para permitir que el operador percibiera el esfuerzo desarrollado.

En 1954 el ingeniero americano George Devol patentó el primer robot industrial con un dispositivo que combinaba la articulación de un teleoperador con el eje servocontrolado de una máquina de control numérico.

1.2. ESQUEMA GENERAL DEL SISTEMA ROBOT

En su esquema general se puede identificar un sistema mecánico, actuadores, sensores; y, el sistema de control como elemento básico necesario para cerrar la cadena actuación-medidas-actuación. En el sistema mecánico puede distinguir entre el órgano terminal, el brazo articulado, y un vehículo, en la mayor parte de los robots industriales no existe tal vehículo, estando fija la base del brazo.

Los sensores externos permiten dotar de sentidos al robot la información que suministran es utilizada por el sistema de percepción para aprender la realidad del entorno, haciendo frente a

situaciones imprevistas, para ello el sistema de control del robot incorpora bucles de realimentación sensorial del entorno, generado automáticamente acciones en función de la comparación de dicha información sensorial con patrones de referencia.

Los sensores internos miden el estado de la estructura mecánica en particular giros o desplazamientos relativos entre articulaciones, velocidades, fuerzas y pares, estos sensores permiten cerrar bucles de control de las articulaciones de la estructura mecánica.

1.2.1. SISTEMA MECÁNICO

El sistema mecánico está compuesto por diversas articulaciones, normalmente se distingue entre el brazo y el órgano terminal que puede ser cambiado, empleando pinzas o dispositivos específicos para distintas tareas.

El aumento del número de articulaciones aporta mayor maniobrabilidad obteniéndose normalmente mayor precisión. Muchos robots industriales actuales tienen menos de los seis grados de libertad de rotación o traslación que se requieren en general para posicionar y orientar en el espacio el órgano terminal, sin embargo también se desarrollan manipuladores altamente redundantes con múltiples articulaciones para aplicaciones en áreas de trabajo de difícil acceso, entre estos cabe destacar los robots tipo serpiente.

Se investiga en robots flexibles que permiten un largo alcance con un peso reducido.

Conviene indicar que las ecuaciones que describen el movimiento del brazo articulado son ecuaciones diferenciales no lineales y acopladas en un caso general resulta difícil obtener soluciones analíticas, físicamente los términos de acoplamiento son representados por: pares gravitacionales dependen de la posición de las articulaciones, los pares de reacción funcionan mediante aceleraciones de otras articulaciones, la magnitud de estas interacciones depende de las características del brazo y de la carga.

1.2.2 ACTUADORES

Los actuadores generan la fuerza o pares necesarios para animar la estructura mecánica, se utilizan tecnologías hidráulicas, para desarrollar potencias importantes; y, neumáticas en la actualidad se ha extendido el empleo de motores eléctricos, en particular motores de corriente continúa servocontroladores empleándose en algunos casos motores paso a paso y otros actuadores electromecánicos sin escobillas.

Existen también robots industriales de accionamiento directo que permiten eliminar los problemas mecánicos inherentes al empleo de engranajes y otras transmisiones, se investiga en nuevos actuadores que disminuyan la inercia al suministrar un par elevado aumenten la precisión, origina menos ruido magnético, bajen de peso y consumo.

Por otra parte, se trata de buscar otras opciones al sistema convencional de accionamiento de articulaciones empleándose para ello conceptos biomecánicos, de esta forma se investiga en manipuladores con actuadores tipo músculo tanto para el brazo como para la mano del robot.

1.2.3. SENSORES Y SISTEMAS DE CONTROL

Los sistemas de control de un robot pueden considerarse funcionalmente descompuestos según una estructura jerárquica; en el nivel inferior se realizan las tareas de servocontrol y supervisión de las articulaciones, la mayor parte de lo robots industriales actuales emplean servomecanismos convencionales con realimentación de posición y velocidad para generar señales de control sobre los actuadores de las articulaciones, los parámetros del controlador son fijos aunque varíen significativamente las condiciones de trabajo con la carga o con el propio movimiento.

Las cargas inerciales, acoplamientos entre articulaciones, y efectos de gravedad son todos dependientes de la posición. El problema se amplía al aumentar la velocidad como resultado en la mayor parte de los robots industriales actuales la velocidad de operación debe ser pequeña.

Existen técnicas para identificar modelos suficientemente fiables de la dinámica del robot y en métodos de control de articulaciones que permitan compensar las no linealidades y acoplamientos, optimiza el comportamiento dinámico, se trabaja en nuevos métodos de control adoptivo que permitan tener en cuenta los cambios en las condiciones de trabajo, y en métodos de control con aprendizaje para mejorar progresivamente la respuesta en operaciones repetitivas típicas en robots industriales.

El segundo nivel de control se ocupa de la generación de trayectorias entendiendo por tal evolución del órgano terminal cuando se desplaza de una posición a otra, el generador de trayectorias debe suministrar a los servomecanismos las referencias apropiadas para conseguir la evolución deseada del órgano terminal a partir de la especificación del movimiento deseado en el espacio de la tarea.

Para obtener las referencias que corresponden a las articulaciones en un determinado punto del espacio de trabajo es necesario resolver el modelo geométrico inverso que no es lineal. Los niveles superiores se ocupan de la comunicación con el usuario interpretación de los programas, percepción sensorial y planificación.

Los primeros robots industriales eran programados manualmente almacenando la secuencia de posiciones en memoria digital, la interacción con la tarea se limitaba a la apertura o cierre de una pinza u otro órgano terminal, indicándolo a un equipo externo o esperando una señal de sincronización, las aplicaciones típicas eran de "pick and place" tales como la carga y descarga de máquinas realizando tareas con movimientos absolutamente definidos y fijos, es decir, se primaba la repetibilidad sobre la adaptación. En cualquier caso los robots podían ser reprogramados para la realización de otras tareas.

En la robótica industrial se ha integrado los progresos en el control por computador de telemanipuladores, simultáneos al desarrollo de los primeros robots, entre estos cabe destacar los trabajos de Shannon y Minsky, en 1958 propusieron un dispositivo al que denominaran "sensor-controlled robot" que consistía en un teleoperador equipado con distintos sensores

conectados a un computador que le suministraba información suficiente para decidir las acciones necesarias en orden a alcanzar un determinado objetivo.

Aunque el dispositivo no llegó a realizarse motivó a otras investigadores tales como Ernst (1961) en su tesis doctoral en el MIT construyó un robot con sensores de tacto en la mano que podía ser programado para realizar tareas tales como la localización, sujetación, transportación y descarga de pequeñas piezas en cajas, este robot que puede ser considerado como el primero controlado mediante sensores externos se programaba mediante instrucciones parecidas a las de un lenguaje ensamblador incorporando órdenes relativas a la información de los sensores de tacto.

Continuando con el empleo de sensores de percepción del entorno, puede mencionarse el importante trabajo de Roberts (1963), demostrando la posibilidad de procesar una imagen digitalizada para obtener una descripción matemática de los objetos incluidos en la escena, expresando su posición y orientación mediante transformaciones homogéneas. Wichman (1967) presentó en Stanford un equipo con cámara de televisión conectada a un computador que podía en tiempo real identificar objetos y sus posiciones.

Pieper (1968) obtuvo una solución al expresar movimientos en coordenadas cartesianas aplicando resultados teóricos de la cinemática, de esta forma fue posible facilitar la programación del robot utilizando llamadas a subrutinas que realizan las transformaciones geométricas correspondientes, en general la utilización de coordenadas cartesianas fue un

importante paso que abrió el camino al empleo de modelos del universo para la toma de decisiones, planificación y verificación de tareas.

Desde el comienzo de los años setenta, se investiga en robots con sensores de visión, resolviendo en tiempo real problemas básicos de manipulación con visión en color (Feldaman, 1971) se ensamblan estructuras de bloques (Eijiri y otros 1972).
Se progresa también en métodos de cálculo de trayectorias con generaciones correspondiente a los servos de las articulaciones y en lenguajes de programación de mayor nivel que incorporan primitivas relaciones con sensores de percepción del entorno y especificación de movimientos en coordenadas cartesianas.

Al final de los años setenta y comienzo de los ochenta se adoptan lenguajes estructurados (Paúl, 1981) con herramientas de programación en tiempo real que progresivamente se introducen en los robots industriales comerciales.

Se trabaja en lenguajes de programación orientados a las tareas basadas en la incorporación de métodos de la Inteligencia Artificial para generación automática de planes, permitiendo también coordinar la actividad de un robot en sistemas de fabricación flexible.

Por lo que respecta a la planificación de caminos libres de obstáculos, el método típico se basa en construir una estructura de datos que represente la geometría del espacio de trabajo o las restricciones existentes, no obstante estos métodos son en general costosos desde el punto de vista computacional lo que suele impedir su aplicación en tiempo real. Existen también

métodos que incorporan la planificación de caminos en el control de bajo nivel utilizando para ello métodos tales como el de los campos potenciales.

En la pasada década surge también el concepto de realimentación visual en control de manipuladores con aplicaciones al sujetar objetos en movimiento. En IEEE (1998) pueden encontrarse diversas aplicaciones de realimentación visual.
Desde los años ochenta se progresa en la manipulación hábil de objetos para una mejor compresión de la mecánica y su planificación.

Se han aplicado también técnicas de control tales como las basadas en redes neuronales, y otras estructuras de control inteligente que están permitiendo resolver problemas que son de elevada complejidad con métodos tradicionales, en general junto a los progresos tecnológicos se requieren planes teóricos que permitan formular una metodología de diseño de estos nuevos sistemas de control en los que se involucran bucles de realimentación sensorial y procesos de decisión y aprendizaje que son difíciles de tratar con los métodos convencionales de la teoría de control.

Desde el punto de vista del procesamiento de la información en robótica se involucran funciones de control de movimientos, percepción y planificación. En un sentido amplio, el sistema de control involucra tanto bucles de realimentación de la información suministrada por los sensores internos, como del entorno.

El desarrollo de sistemas de percepción en Robótica surge a partir de los progresos tecnológicos en sensores tales como los de visión, tacto e incluso audición, sin embargo la percepción involucra no sólo la captación de la información sensorial, sino también su tratamiento e interpretación, por tanto es necesario realizar una abstracción a partir de un cierto conocimiento previo del entorno, es claro que la complejidad de la percepción artificial depende de lo estructurado que esté dicho entorno.

Por último, la planificación tiene como objetivo encontrar una trayectoria desde una posición inicial a una posición objetiva, sin colisiones, y minimizando un determinado índice. En el caso más simple, el problema se plantea en un entorno que se supone conocido y estadístico, además que el robot es ominidireccional que se mueve suficientemente lento y que es capaz de seguir el camino en forma perfecta.

1.3. PARÁMETROS PARA CLASIFICAR A LOS ROBOTS

El robot consigue mantener el equilibrio sobre el suelo gracias a sus múltiples sensores que proporcionan información sobre su estabilidad. Tiene suficiente fuerza como para desplazar a un ser humano sobre él.

1.3.1. ROBOT HUMANOIDE

Es un robot que presenta las características de un ser humano: dos piernas, dos brazos, torso y cabeza.

1.3.2. ROBOT BÍPEDO

Es un robot que dispone de dos piernas para desplazarse.

1.3.3. ROBOT BÍPEDO DINÁMICO

Robot cuyo sistema de locomoción está basado en dos piernas y que es capaz de andar sin necesidad de interrumpir su avance.

1.3.4. ROBOT BÍPEDO ESTÁTICO

Robot cuyo sistema de locomoción está basado en dos piernas y que debe interrumpir su avance al andar para garantizar el equilibrio.

1.3.5. ROBOTS COLABORATIVOS

Robots que interactúan entre sí para desarrollar una labor conjunta, también se les llama Enjambre de Robots.

1.4. CLASIFICACIÓN DE LOS ROBOTS

La potencia del software en el controlador determina la utilidad y flexibilidad del robot dentro de las limitantes del diseño mecánico y la capacidad de los sensores.

Los robots han sido clasificados de acuerdo a su generación, a su nivel de inteligencia, a su nivel de control, y a su nivel de lenguaje de programación. Estas clasificaciones reflejan la potencia del software en el controlador, en particular, la sofisticada interacción de los sensores.

1.4.1. SEGÚN SU GENERACIÓN

La generación de un robot se determina por el orden histórico de desarrollos en la robótica. Cinco generaciones son normalmente asignadas a los robots industriales:

1.4.1.1 ROBOTS DE 1ª GENERACIÓN

El sistema de control usado en la primera generación de robots está basado en la "paradas fijas" mecánicamente.

Como ejemplo de esta primera etapa están los mecanismos de relojería que mueven las cajas musicales o los juguetes de cuerda.

1.4.1.2. ROBOTS DE 2ª GENERACIÓN

El movimiento se controla a través de una secuencia numérica almacenada en disco o cinta magnética. Por regla general, este tipo de robots se utiliza en la industria automotriz y son de gran tamaño.

1.4.1.3. ROBOTS DE 3ª GENERACIÓN

Utilizan las computadoras para su control y tienen cierta percepción de su entorno a través del uso de sensores. Con esta generación se inicia la era de los robots inteligentes y aparecen los lenguajes de programación para escribir los programas de control.

1.4.1.4. ROBOTS DE 4ª GENERACIÓN

Se trata de robots altamente inteligentes con más y mejores extensiones sensoriales, para entender sus acciones y captar el mundo que los rodea. Incorporan conceptos "modélicos" de conducta.

1.4.1.5. ROBOTS DE 5ª GENERACIÓN

Actualmente en desarrollo. Esta nueva generación de robots basará su acción principalmente en modelos conductuales establecidos.

1.4.2. SEGÚN SU ARQUITECTURA GENERACIONAL

1.4.2.1. ROBOTS PLAY-BACK

Los cuales regeneran una secuencia de instrucciones grabadas, como un robot utilizado en recubrimiento por spray o soldadura por arco. Estos robots comúnmente tienen un control de lazo abierto.

1.4.2.2. ROBOTS CONTROLADOS POR SENSORES

Estos tienen un control en lazo cerrado de movimientos manipulados, y hacen decisiones basados en datos obtenidos por sensores.

1.4.2.3. ROBOTS CONTROLADOS POR VISIÓN

Donde los robots pueden manipular un objeto al utilizar información desde un sistema de visión.

1.4.2.4. ROBOTS CONTROLADOS ADAPTABLEMENTE

Donde los robots pueden automáticamente reprogramar sus acciones sobre la base de los datos obtenidos por los sensores.

1.4.2.5. ROBOTS CON INTELIGENCIA ARTIFICIAL

Donde los robots utilizan las técnicas de inteligencia artificial para hacer sus propias decisiones y resolver problemas.

1.4.2.6. ROBOTS MÉDICOS

Fundamentalmente, prótesis para disminuirlos físicos que se adaptan al cuerpo y están dotados de potentes sistemas de mando. Con ellos se logra igualar al cuerpo con precisión los movimientos y funciones de los órganos o extremidades que suplen.

1.4.2.7. ANDROIDES

Robots que se parecen y actúan como seres humanos. Los robots de hoy en día vienen en todas las formas y tamaños, pero a excepción de los que aparecen en las ferias y espectáculos, no se parecen a las personas y por tanto no son androides. Actualmente, los androides reales sólo existen en la imaginación y en las películas de ficción.

1.4.2.8. ROBOTS MÓVILES

Provistos de patas, ruedas u orugas que los capacitan para desplazarse de acuerdo su programación. Elaboran la información que reciben a través de sus propios sistemas de

sensores y se emplean en determinado tipo de instalaciones industriales, sobre todo para el transporte de mercancías en cadenas de producción y almacenes.

También se utilizan robots de este tipo para la investigación en lugares de difícil acceso o muy distantes, como es el caso de la exploración espacial y las investigaciones o rescates submarinos.

1.4.3. POR NIVEL DE INTELIGENCIA

La Asociación de Robots Japonesa (JIRA) ha clasificado a los robots dentro de seis clases sobre la base de su nivel de inteligencia

1.4.3.1. DISPOSITIVOS DE MANEJO MANUAL

Controlados por una persona.

1.4.3.2. ROBOTS DE SECUENCIA VARIABLE

Donde un operador puede modificar la secuencia fácilmente.

1.4.3.3. ROBOTS REGENERADORES

Donde el operador humano conduce el robot a través de la tarea.

1.4.3.4. ROBOTS DE CONTROL NUMÉRICO

Donde el operador alimenta la programación del movimiento, hasta que se enseñe manualmente la tarea.

1.4.3.5. ROBOTS INTELIGENTES

Los cuales pueden entender e interactuar con cambios en el medio ambiente.

1.4.4. POR NIVEL DE CONTROL

Los programas en el controlador del robot pueden ser agrupados de acuerdo al nivel de control que realizan o su predecibilidad en las formas para realizar su función.

1.4.4.1. NIVEL DE INTELIGENCIA ARTIFICIAL

Donde el programa aceptará un comando como "levantar el producto" y descomponerlo dentro de una secuencia de comandos de bajo nivel basados en un modelo estratégico de las tareas.

1.4.4.2. NIVEL DE MODO DE CONTROL

Donde los movimientos del sistema son modelados, para lo que se incluye la interacción dinámica entre los diferentes mecanismos, trayectorias planeadas, y los puntos de asignación seleccionados.

1.4.4.3. NIVELES DE SERVOSISTEMAS

Donde los actuadores controlan los parámetros de los mecanismos con el uso de una retroalimentación interna de los datos obtenidos por los sensores, y la ruta es modificada sobre la base de los datos que se obtienen de sensores externos. Todas las detecciones de fallas y mecanismos de corrección son implementadas en este nivel.

1.4.5. POR LENGUAJE DE PROGRAMACIÓN

En la clasificación final se considerara el nivel del lenguaje de programación. La clave para una aplicación efectiva de los robots para una amplia variedad de tareas, es el desarrollo de lenguajes de alto nivel.

Existen muchos sistemas de programación de robots, aunque la mayoría del software más avanzado se encuentra en los laboratorios de investigación. Los sistemas de programación de robots caen dentro de tres clases:

1.4.5.1. SISTEMAS GUIADOS

En el cual el usuario conduce el robot a través de los movimientos a ser realizados.

1.4.5.2. SISTEMAS DE PROGRAMACIÓN DE NIVEL-ROBOT

En los cuales el usuario escribe un programa de computadora al especificar el movimiento.

1.4.5.3. SISTEMAS DE PROGRAMACIÓN DE NIVEL-TAREA

En el cual el usuario especifica la operación por sus acciones sobre los objetos que el robot manipula.

1.5. ÁREAS DE LA ROBÓTICA

En la robótica existen dos grandes áreas: **manipulación** y **Locomoción**. La manipulación es la capacidad de actuar sobre los objetos, trasladándolos o modificándolos. Esta área se centra en la construcción de manipuladores y brazos robóticos.

La locomoción es la facultad de un robot para poder desplazarse de un lugar a otro. Los robots con capacidad locomotiva se llaman **robots móviles**.

1.5.1. MANIPULACIÓN

La mayor parte de los robots industriales actuales son esencialmente brazos articulados, según "Robot Institute of America", un robot industrial es un manipulador programable multifunciónal diseñado para mover materiales, piezas, herramientas o dispositivos especiales, mediante movimientos programados para la ejecución de distintas tareas. Véase en la (FIG. 1.1)

En la robótica se sustituye a equipos capaces de automatizar operaciones concretas por máquinas de uso general que puedan realizar distintas tareas, la realización de programa de las funciones de control ofrecen mucha mayor flexibilidad; y, la posibilidad de implantar funciones complejas necesarias para controlar el manipulador.

1.5.2. LOCOMOCIÓN

Según los **efectores empleados** para conseguir la locomoción, tradicionalmente se ha establecido la siguiente **clasificación**:

1.5.2.1. CON RUEDAS

Los efectores son ruedas. Por ejemplo, el microbot tritt, tiene tres ruedas, dos motrices y una loca. (FIG. 1.2)

1.5.2.2. CON ORUGAS

Por ejemplo los robots tipo ``carro de combate'', como el Robot de Docencia. (FIG. 1.3)

1.5.2.3. CON PATAS

Cualquier robot que use patas para conseguir la locomoción: perros, gatos, hexápodos, arañas... Normalmente se trata de robots bio-inspirados. Por ejemplo el robot perro Puchobot, o como el Hexápodo Sheila. (FIG. 1.4)

1.5.2.4. OTROS

Aquí se sitúa cualquier otro tipo de locomoción no clasificable en ninguna de las anteriores categorías, como por ejemplo el robot gusano **Cube 2.0.** (FIG. 1.5)

Esta clasificación no es del todo satisfactoria porque existen demasiados robots que se pueden introducir en la categoría *otros*. Mark Yim introdujo una nueva taxonomía que re describe en un apartado.

Según cómo se realice la locomoción, existen dos tipos:

1.5.2.5. **LOCOMOCIÓN ESTÁTICAMENTE ESTABLE** (STATICALLY STABLE LOCOMOTION)

El robot debe tener suficientes puntos de apoyo, que conforman el **polígono de apoyo**. El centro de gravedad (CG) debe caer siempre dentro de este polígono.

1.5.2.6. **LOCOMOCIÓN DINÁMICAMENTE ESTABLE** (DINAMICALLY STABLE LOCOMOTION)

El robot tiene que ser estable en movimiento (no caerse) aunque puede no ser estable en reposo (ejemplo, un robot unípodo). Este tipo de locomoción requiere más control pero aporta mayor velocidad.

Dentro de la locomoción hay tres puntos claves:

1.5.2.7. **DESPLAZARSE UN INCREMENTO EN LÍNEA RECTA**

La complejidad depende del tipo de robot. Es muy sencillo que un robot con ruedas avance en línea recta, pero no es tan sencillo que lo haga un robot con patas.

1.5.2.7. GIROS Y TRASLACIONES EN MÚLTIPLES DIRECCIONES

Nuevamente depende del tipo de robot. El hacer que un robot ápodo pueda desplazarse por un plano es más complejo que en un robot con ruedas.

1.5.2.8. PLANIFICACIÓN DE TRAYECTORIAS Y NAVEGACIÓN

Que el robot sepa qué camino elegir para llegar a un determinado lugar.

1.6. LOCOMOCIÓN MEDIANTE PATAS

Permiten asistir el cuerpo del terreno empleado únicamente puntos discretos de soporte, es posible adaptar el polígono de soporte para mantener la estabilidad y pasar sobre obstáculos, por consiguiente tiene mejores propiedades que las ruedas para atravesar terrenos difíciles llenos de obstáculos.

Mediante la locomoción de patas es posible conseguir la omnidireccionalidad y el deslizamiento en la locomoción es mucho menor en los robots con patas la complejidad de los mecanismos necesarios es mayor, así como el consumo de energía en la locomoción, en principio los problemas de planificación y control son más complejos que en los vehículos con ruedas.

La configuración más común es la de seis patas, vea el anexo de la (FIG. 1.6) existen también de ocho patas tales como el construido en el Robotics Institute de la Carnegie Mellon University que ha sido empleado para la exploración de un volcán en Alaska (1994), vea el anexo de la (FIG. 1.7). Hay cuadrúpedos, vea el anexo de la (FIG. 1.8) y bípedos como se muestra en el anexo (FIG.1.9)

En este punto conviene mencionar también a los robots trepadores, que son de interés para realizar operaciones tales como inspecciones (Abderrahim y otros, 1999) y reparaciones en paredes verticales, estos robots se sostienen mediante garras empleando dispositivos de succión o magnético en las patas, existen también sistemas mixtos de locomoción con ruedas para terrenos lisos y patas para salvar obstáculos, subir escaleras, trepar, etc.

1.7. ROBOTS BÍPEDOS

El primer gran humanoide que realmente llego a la sociedad fue el robot diseñado y fabricado por HONDA desde 1986 hasta la actualidad.

Otros robots bípedos y humanoides también se destacaron durante este tiempo, pero "el robot que recibio el nombre de ASIMO" fue el principal referente. Por encima de proyectos desarrollados en el Instituto Tecnológico de Massachussets o la Universidad Carnegie Mellon.

En 1986 los ingenieros de Honda empezaron a trabajar en la problemática de caminar, la pregunta era ¿qué necesita un robot para poder caminar dinámicamente?, hasta esa fecha

muchos documentos científicos habían señalado la dificultad de fabricar robots caminantes, pero muy pocos científicos se atrevían a señalar la respuesta a la pregunta.

El primer ingenio robótico de HONDA era el E0, diseñado en 1986, para la época era un autentico prodigio que podía moverse sobre dos piernas no sin caerse en numerosas ocasiones. (FIG. 1.10) Y (FIG. 1.11)

Entre 1987 y 1991 HONDA trabajó en las siguientes versiones del robot: E1, E2, E3.

Entre 1991 y 1993 con las nuevas versiones E4, E5 y E6 se empezaron a emplear conceptos como el ZMP (Zero Moment Point), que hoy en día componen el ABC de la robótica bípeda.

Durante esta epoca el proyecto se mantuvo bajo un relativo secretismo, diversas instituciones sabían que HONDA estaba trabajando en robótica humanoide pero pocos se podían imaginar los avances que los científicos nipones estaban alcanzando.

Entre 1993 y 1997 surgieron P1, P2 y P3, autenticas maquinas humanoides con tanto nivel de detalle que el público empezó a asombrarse con los resultados obtenidos. El modelo más voluminoso llegó a ser el P2 que pesaba 210 Kg y medía 1,82 mts.

Cuando HONDA llegó a estos extremos, en los que veía claramente que controlaba la robótica humanoide básica (la capacidad de caminar) analizó sus posibilidades comerciales, llegando a la conclusión de que un robot que pudiera aplastar a su propietario no era muy comercial.

Entonces surgió ASIMO, un pequeño robot de 1,20 cm de altura y 43 kg de peso que podría maravillar al mundo saliendo en la televisión sin riesgo para sus coetáneos.

En realidad ASIMO ha cambiado mucho desde su primera aparición a principios de milenio. En un principio pesaba 54 Kg. pero a base de "dieta tecnológica" ha pasado a pesar 43 Kg. (en Enero de 2004).

La "dieta tecnológica" más popular es sin lugar a dudas la Japonesa, la cultura de la nanotecnologia y miniaturizacion arrasa en el mercado tecnologico japones y eso se traduce en perdidas de peso para ASIMO.

1.8. ROBOTS MÓVILES

El desarrollo de robots móviles responde a la necesidad de extender el campo de aplicación de la Robótica restringido inicialmente al alcance de una estructura mecánica anclando en uno de sus extremos, se trata también de incrementar la autonomía limitando todo lo posible la intervención humana.

Desde el punto de vista de la autonomía, los robots móviles tienen como precedentes los dispositivos electromecánicos tales como los denominados "micro-mouse" creados desde los años treinta para desarrollar funciones inteligentes tales como descubrir caminos en laberintos, ejemplo la tortuga de Walter presentada en 1948 está podía reaccionar ante la presencia de obstáculos subir pendientes.

Estos trabajos de investigación no guardan una relación directa con los vehículos autónomos que comenzaron a aplicarse desde los años sesenta en la industria, siendo guiados por cables bajo el suelo o mediante sensores ópticos para seguir líneas trazadas en la planta. Estas aplicaciones hoy en día comunes en muchos procesos de fabricación, se caracterizan por un entorno fuertemente estructurado para facilitar la automatización.

En los años setenta se vuelve a trabajar en el desarrollo de robots móviles dotados de una mayor autonomía, la mayor parte de las experiencias se desarrollan empleando plataformas que soportan sistemas de visión, sin embargo el desarrollo tecnológico todavía no era el suficiente para lograr la navegación autónoma de forma eficiente.

En los años ochenta el incremento espectacular de la capacidad computacional y el desarrollo de nuevos sensores y sistemas de control permite aumentar la autonomía, en esta década los desarrollos de robots móviles tanto para interiores como para navegación exterior realizados en la Carnegie Mellon University, se trata que el robot tenga la suficiente inteligencia como para reaccionar y tomar decisiones basándose en observaciones de su entorno sin suponer que este es perfectamente conocido.

La autonomía de un robot se basa en el sistema de navegación automática, en estos sistemas se incluyen tareas de planificación, percepción y control, en los robots móviles el problema de la planificación en el caso más general puede descomponerse en planificación global de la misión, de ruta, de trayectoria; y, finalmente evitar obstáculos no esperados.

En un robot para interiores la misión podría consistir en determinar a qué habitación debe 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.

Existen numerosos métodos de planificación de caminos para robots móviles que se basan en hipótesis simplificadoras. El entorno es conocido y estático robots omnidireccionales con movimiento lento y ejecución perfecta de trayectoria, en particular hay muchos métodos que buscan caminos libres 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 la utilización de restricciones de varios niveles de resolución y búsqueda jerarquizada que permiten hacer más eficiente el proceso con vistas a su aplicación en tiempo real.

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 acontecimientos no considerados, la definición de la trayectoria debe tener en cuenta las características cinemáticas del vehículo.

Además de las características geométricas y cinemáticas puede ser necesario tener en cuenta modelos dinámicos de comportamiento del vehículo contemplando la interacción vehículo-terreno, se puede plantearse 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 dichos 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 debe seguir.

En cualquier caso el problema del control automático preciso de un vehículo con ruedas puede resultar más complejo que el de los manipuladores debido a la presencia de restricciones no holónomas. Los bucles de control se plantean tanto en el espacio de las variables articulares como en coordenadas del mundo y las ecuaciones de movimiento son complejos si se considera la interacción con el terreno, mientras en los manipuladores es relativamente fácil el cálculo la medida de los pares y fuerzas que se ejercen sobre la estructura mecánica la determinación de estos pares en vehículos con ruedas es muy difícil, en la actualidad se emplean fundamentalmente métodos geométricos y modelos cinemáticas simplificados, no obstante la consideración de aspectos dinámicos es necesaria cuando la velocidad es alta.

La técnica más simple consiste en la utilización de la edometría a partir de las medidas suministradas por los sensores situados en los ejes de movimiento típicamente codificadores ópticos, sin embargo la acumulación de error puede ser muy grande, se emplean también sistemas de navegación inercial incluyendo acelerómetros aunque estos sistemas también acumulan error especialmente en la determinación de la posición empleando los

acelerómetros, la combinación de las técnicas odométricas con la medida de los ángulos de orientación puede dar buenos resultados en intervalos de tiempo y distancia.

La incertidumbre de la posición se reduce a intervalos suficientemente grandes empleando el sistema de percepción, en particular suelen emplearse marcas especiales cuya detección permite estimar con precisión la posición del robot en entornos no estructurados con ausencia de marcas especiales, la estimación de la posición mediante el sistema de percepción es notablemente más compleja.

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 y estimar la posición del vehículo de forma precisa, el sistema de percepción de estos robots puede aplicarse no sólo para navegar sino también para aplicaciones tales como el control de un manipulador situado en el robot.

Para el diseño de estos sistemas de percepción deben tenerse en cuenta diferentes criterios, algunos de los cuales son conflictivos entre si, de esta forma 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 diversas condiciones de navegación con requerimientos de percepción diferentes, de esta manera 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 se necesita una gran precisión en su localización.

Existen también 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 precisa aunque relativamente lenta, la posición del robot a la vez debe detectar obstáculos lo suficientemente rápido aunque no se necesite una gran precisión en su localización.

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

Con respecto a los sensores específicos 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 originales por el vehículo y el entorno, su tamaño, consumo, seguridad de funcionamiento y desgaste.

Las cámaras de vídeo tienen la ventaja de su amplia difusión y precio su carácter pasivo que no es necesario en principio el empleo de dispositivos mecánicos para la captación de la imagen, las desventajas son los requerimientos computacionales la sensibilidad a las condiciones de iluminación, los problemas de calibración y fiabilidad.

La percepción activa mediante láser 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 distancia y reflectancia a las superficies interceptadas por el haz.

Los sensores de ultrasonido son económicos y simples para la navegación, se basan en la determinación del dominado tiempo de vuelo de un pulso de sonido, sin embargo la influencia de las condiciones ambientales puede ser significativa, debiendo corregirse mediante una calibración adecuada, la relación señal/ruido es normalmente muy inferior a la de los otros sensores lo que puede hacer necesario el empleo de múltiples frecuencias técnicas de filtrado y tratamiento de la incertidumbre de mayor complejidad computacional, la resolución lateral es mala, existiendo técnicas para evitar enfoque mediante lentes acústicos o transmisores curvos.

CAPÍTULO II

DETERMINACIÓN DE REQUISITOS (Análisis).

2.1. INTRODUCCIÓN A LOS REQUISITOS.

Un proyecto no puede ser exitoso sin una especificación correcta de los requisitos, para ello se necesita muchas habilidades; un examen riguroso de las mismas permitirá obtener un buen resultado en la culminación del proyecto.

Para esto se llevo a cabo una investigación de la cual se determina una propuesta de los posibles requisitos para introducir aplicaciones de robótica en la Universidad Técnica de Cotopaxi, esta propuesta consiste en el desarrollo de un prototipo de robot con capacidad de hablar y caminar asistido por un computador, el envió de datos se lo realiza desde Visual Basic los cuales son trasferidos al PIC Basic, para luego ser ejecutados los movimientos del robot, la voz del robot se lo realizara con un componente especial de Microsoft que será ejecutada desde el programa.

En lo que respecta al prototipo de robot se desarrollara con piezas electrónicas la parte inferior y la superior se le acoplara un cuerpo elemental su conexión será a través de cables, asistido por un computador.

2.1.1. REQUISITOS.

Estos son una descripción de las necesidades o deseos de un producto, la meta primaria de la fase de requerimientos es identificar y documentar lo que en realidad se necesita en forma clara se lo comunique al cliente y a los miembros del equipo de desarrollo, el reto consiste en definirlos de modo que se detecten los riesgos y no se presenten sorpresas al momento de entregar el producto.

Se recomienda los artefactos en la fase de inicio:

- Presentación general
- Usuarios
- Metas
- Funciones del sistema
- Atributos del sistema
- Funciones del prototipo de robot.
- Elementos del prototipo de robot.

2.1.2. PRESENTACIÓN GENERAL.

Este proyecto tiene por objeto crear un prototipo de robot con capacidad de hablar y caminar asistido por un computador que permita introducir aplicaciones de robótica motivando a los estudiantes a desarrollar proyectos tecnológicos y científicos en la Universidad Técnica de Cotopaxi.

2.1.3. USUARIOS.

Los estudiantes y docentes de la Universidad Técnica de Cotopaxi.

2.1.4. METAS.

Es el desarrollo de un prototipo de robot con capacidad de hablar y caminar que permita introducir aplicaciones de robótica, ahora es cuando se tiene la oportunidad de realizar una tesis, momento de cristalizar un proyecto ambicioso que se convertirá en una prioridad.

2.1.5. FUNCIONES DEL SISTEMA.

Las funciones del sistema es lo que habrá de hacer; por ejemplo en el ingreso de datos, las funciones se dividen en dos categorías:

Evidente.- Debe realizarse el usuario para saber que se ha realizado.

Oculta.- Debe realizarse auque no va visible para los usuarios, esto se aplica a muchos servicios técnicos subyacentes como guardar información en un mecanismo persistente de almacenamiento.

2.1.5.1. FUNCIONES BÁSICAS.

Las siguientes funciones del sistema para el funcionamiento del prototipo de robot son las mínimas necesarias para el buen funcionamiento del sistema planteado:

Tabla No 1.1 (Funciones Básicas del Sistema)

Fuente: Investigador

Ref. No.	Función	Categoría
R1.1	La teleprogramación del prototipo de robot por el usuario mediante ingreso de datos.	Evidente
R1.2	Proveer un método estándar para la depuración de ingreso de datos.	Evidente
R1.3	La teleoperación del prototipo de robot por el usuario.	Evidente
R1.4	Ofrecer mecanismos de comunicación entre los procesos y los sistemas.	Oculta
R1.5	Ofrecer mecanismos de comunicación entre el sistema y el prototipo de robot.	Oculta

2.1.6. ATRIBUTOS DEL SISTEMA.

Los atributos del sistema son sus características o dimensiones:

Tabla No 1.2 (Atributos del Sistema) **Fuente: Investigador**	
Atributo	**Detalles y restricciones de frontera**
Tiempo de respuesta	(restricción de frontera) Antes de ingresar los datos se tiene que almacenar la secuencia de pasos del Pic Basic en la memoria, para que pueda interactuar con el programa de Visual Basic. En el menor tiempo posible.
Metáfora de interfaz	(detalle) maximiza una navegación fácil con teclado.
Tolerancia a fallas	(restricción de frontera) debe prever que el hardware este conectado correctamente ya que podría producir un mal manejo del prototipo de robot.
Plataformas del sistema operativo	(detalle)Windows.

2.1.7. FUNCIONES BÁSICAS DEL PROTOTIPO DE ROBOT

Las siguientes funciones del prototipo de robot son las mínimas necesarias para el buen funcionamiento del prototipo planteado se lo realizara mediante dos categorías:

- Evidente.
- Oculto.

Tabla No 1. 3 (Funciones Básicas del Prototipo de Robot) Fuente: Investigador		
Ref. No.	**Función**	**Categoría**
R1.1	Ofrecer un mecanismo de comunicación con el prototipo de robot y el sistema.	**Oculto**
R1.2	Ofrecer un mecanismo de almacena niento de datos.	**Oculto**
R1.3	El prototipo de robot realizara la secuencia de pasos que le permitirá caminar.	**Evidente**
R1.4	Al ingresar el usuario al sistema el prototipo de robot automáticamente realizará una secuencia de frases.	**Evidente**

2.1.8. ELEMENTOS DEL PROTOTIPO DE ROBOT

Los elementos del prototipo de robot, sus complementos se describen y se muestran en la (Tabla.N. 1.4) y el anexo de la (Fig. 2.1).

Tabla No 1. 4 (Elementos del Prototipo de Robot) Fuente: Investigador	
Descripción	**Cantidad**
Componentes Electrónicos	
Servos Motores	2
DB9	4
Carcasas DB9	4
Condensador 1000 uf 25V	1

Regulador Lm 317	1
Resistencias 1 kΩ y 270Ω	2
Transformador 1 A.	1
Puente rectificador 1 A	1
Disipador	1
Conector de tres salidas	1
Circuito board con el microcontrolador Basic Stamp 2.	1
Circuito de Audio	1
Partes Metàlicas	
Plato azul	1
Pies azules	2
Piernas plomas	4
Caja de protección de lo servos	1
Tobillos plomos	2
HARDWARE	
Junturas, incluida con sus roscas.	4
Varas de aceros de diámetro 5.4", consta de 2/56 hilo en cada extremo,	2
Junturas de plásticas de 1/16" con 2/56 hilos.	4
Parlantes	2
Cuerpo elemental.	1
Tornillos de 3/16 x 12", incluido sus roscas.	4
Tornillos 4/40	4
Llave hexadecimal en L 3/32"	1
Tornillos de 4/40 ¼.	4

Varas de acero largas de 3/16" 3".	2
Rodelas plásticas	4
Collares de 3/16"	4
Llave hexadecimal en L 3/16"	1
Guardianes del alambre plásticos 4/40 (Conectores E-Z)	2
Alambres de ángulo recto de latón	2
Varas de acero largas de 3/16" 3".	2
Rodelas plásticas	4
Cuernos de servos	2
Tornillos negros pequeños para sostener el cuerno de los servos.	2
Guardianes de alambre de latón (Conectores E/Z).	2
Tornillos de 4/40 3/8"	12
Roscas de 4/40.	12
MISCELANIA	
Cable serial	1
Cable UTP categoría 6p	5m
Cable de poder	1
Caja metálica	1
Crema disipadora	1
Cloruro fèrrico	1
Placa de baquelita 10x6 cm.	1
Taipe.	1

2.2. INTRODUCCIÓN A LOS CASOS DE USO.

Una técnica excelente que permite mejorar la compresión de los requisitos es la creación de casos de uso es decir descripciones narrativas de los procesos del dominio.

2.2.1. ACTIVIDADES Y DEPENDENCIAS.

Los casos de uso requieren tener al menos un conocimiento parcial de los requerimientos del sistema.

2.2.2. CASOS DE USO.

El uso es un documento narrativo que describe la secuencia de eventos de un actor (agente externo) que utiliza un sistema para completar un proceso como: la teleoperación y la teleprogramaciòn.

2.2.3. ACTORES.

Es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso, por lo regular estimula el sistema con eventos de entrada o recibe algo de el, los actores están representados por el papel que desempeñan en el caso de: usuario, prototipo de robot u otro.

Los actores suelen ser representados por seres humanos pero pueden ser cualquier tipo de sistema como un sistema computarizado externo

2.3. MODELO DE CASOS DE USO DE LA APLICACIÓN.

En el anexo de la (FIG 2.2) se muestra el diagrama de casos de uso para el prototipo de robot asistido por computador para la Universidad Técnica de Cotopaxi.

2.3.1 DESCRIPCIÓN DE PROCESOS.

La descripción de procesos a través del relato de las actividades que pueden generar los casos de uso permiten comprender de mejor manera los objetos o elementos que se constituirán en las piezas a desarrollarse en el sistema para este efecto se puede incluir la narrativa de los casos de uso de alto nivel que son la explicación generalizada de un proceso y los casos de uso expandidos en los cuales se incluye una buena dosis de detalle de tal forma que permitan determinar los pasos a seguirse en cada uno de los procesos sin llegar a detallar el 100% de los elementos mas significativos.

2.3.2. CASOS DE USO DE ALTO NIVEL.

El uso de alto nivel describe claro y conciso el proceso que se quiere especificar, los encabezados y la estructura de estos casos de uso son representativos, sin embargo el UML (Lenguaje Unificado de Modelado) no específica un formato rígido puede modificarse para

atender las necesidades y ajustarse al espíritu de la documentación ante todo una comunicación clara, se iniciara con los casos de uso de alto nivel para lograr rápidamente entender los principales procesos globales que intervienen en el prototipo de robot con capacidad de hablar y caminar asistido por un computador.

Caso de uso:	**Teleprogramación.**
Actores:	Usuario, Prototipo de Robot.
Tipo:	Primario.
Descripción:	El usuario escoge el método de la teleprogramación, seleccionara el tipo de secuencia, tendrá la opción de ingresar el número de secuencias para el prototipo de robot.

Caso de uso:	**Teleoperación.**
Actores:	Usuario, Prototipo de Robot.
Tipo:	Primario.
Descripción:	El usuario escoge el método de la teleoperación, selecciona el tipo de secuencia que realizara el prototipo de robot.

2.3.3. CASOS DE USO EXPANDIDOS DEL SISTEMA.

Un caso de uso extendido muestra mas detalles que uno de alto nivel, este tipo de casos suele ser útiles para alcanzar un conocimiento más profundo de los procesos y de los requerimientos, damos en seguida la descripción de los casos de uso expandidos del sistema.

Caso de uso: **Teleprogramación.**

Actores: Usuario, Prototipo de Robot.

Propósito: Realice una secuencia de pasos el prototipo de robot.

Tipo: Primario.

Descripción: El usuario luego de escoger el método de teleprogramación, selecciona el tipo de secuencia e inmediatamente ingresa el número de pasos a realizar el prototipo de robot.

Referencias cruzadas: R.1.1, R.1.2, R.1.4, R.1.5.

Tabla N 1.5 Curso normal de los eventos de la teleprogramación **Fuente: Investigador.**	
Acción del actor	**Respuesta del Sistema**
1. Este caso de uso comienza cuando se ha seleccionado el método de teleprogramación.	
	2. Adicionalmente el sistema realiza el proceso automático que el prototipo de robot comience a decir una secuencia de frases indicando como es el proceso.
3. El usuario escucha que el prototipo de robot habla.	
4. Posteriormente el usuario inicia la selección de la secuencia de pasos.	
5. El usuario ingresa el numeró de pasos a	

a realizar el prototipo de robot.	
	6. El sistema se encarga de bloquear el ingreso de caracteres, números negativos, etras. Admitiendo solo números enteros positivos y decimales.
	7. El sistema detecta cuando el número ingresado es correcto.
	8. El sistema envía los datos al prototipo de robot.
9. El usuario puede escoger la opción de parar en cualquier momento dicha ejecución.	
	10. En este momento finalmente el sistema permite la ejecución de la secuencia de pasos el prototipo de robot.
11. El Usuario visualiza que el prototipo de robot camina.	
	12. Finalmente el sistema finaliza la ejecución de los procesos.

Caso de uso: **Teleoperación.**

Actores: Usuario, Prototipo de Robot.

Propósito: Realice una secuencia de pasos el prototipo de robot.

Tipo: Primario.

Descripción: El usuario luego de escoger el método de teleoperación, selecciona el tipo de secuencia que realizara el prototipo de robot.

Referencias cruzadas: R.1.3.R.1.4.R.1.5

Tabla N 1.6 Curso normal de los eventos de la teleoperación **Fuente: Investigador.**	
Acción del actor	**Respuesta del Sistema**
1. Este caso de uso comienza cuando se ha seleccionado el método de teleoperación	
	2. Adicionalmente el sistema realiza el proceso automático que el prototipo de robot comience a decir una secuencia de frases indicando como es el proceso.
3. El usuario escucha que el prototipo de robot habla.	
	4. El sistema permite que el usuario tele_ opere el prototipo de robo mediante las flechas de direccionamientos de arriba y de abajo.
	5. El sistema inmediatamente ejecutara las secuencias de pasos para el prototipo de robot.
6. El Usuario visualiza que el prototipo de robot camina.	
	7. Finalmente el sistema finaliza la ejecución de los procesos.

2.3.4. CLASIFICACIÓN DE LOS CASOS DE USO

Es necesario clasificar los casos de uso y los casos de alto rango, han de tratarse al inicio del ciclo de desarrollo, la estrategia general consiste en escoger primero los casos que influyen profundamente en la arquitectura básica, he aquí algunas cualidades que aumentan la clasificación de un caso:

1. Tener una fuerte repercusión en el diseño arquitectónico.
2. Con poco esfuerzo obtener información e ideas importantes sobre el diseño.
3. Incluir funciones riesgosas urgentes o complejas.
4. Requerir una investigación a fondo o tecnología nueva y riesgosa.
5. Representar procesos primarios de la línea de negocios.
6. Apoyar directamente el aumento de ingresos o la reducción de costos.

Clasificación	Caso de Uso	Justificación
Alto	◆ Teleprogramación ◆ Teleoperación	Corresponden a los criterios de clasificación más altos dentro del sistema por cuanto corresponden al ingreso de datos y manipulación, para que el prototipo de robot realice los pasos.

2.4. CICLO DE DESARROLLO DE LA APLICACIÓN.

El ciclo de desarrollo se lo realizara mediante cuatro fases: inicio, elaboración, construcción y transición vea el anexo de la (FIG 2.3), cada una de estas fases tendrán sus respectivos artefactos vea el anexo de la (FIG 2.4).

En lo que respecta a la fase de inicio ha concluido, los casos de uso han sido identificados y clasificados, se presenta un cambio importante, inicia la fase de elaboración en la cual se cumple un diseño de alto nivel una de las primeras actividades es el modelo conceptual.

Posteriormente se realizara la fase de construcción donde se tendrá un diseño de bajo nivel la cual consistirá en refinamiento de los artefactos de la fase de elaboración y finalizando se tiene la fase de transición la cual se refinara los artefactos de la fase de construcción.

2.5. MODELO CONCEPTUAL DE LA APLICACIÒN

Un modelo conceptual explica (a sus creadores) los conceptos significativos en un dominio del problema es el artefacto más importante a crear durante el análisis orientado a objetos, la creación de un modelo conceptual se agrupa a conceptos idóneos, contiene muchas categorías comunes que vale la pena tener en cuenta sin que importe el orden.

Tabla No 1.7 (Conceptos) **Fuente: Investigador**	
Categoría del concepto	**Ejemplos**
Objetos físicos o tangibles	Computador
Especificaciones, diseño o descripciones de cosas	Especificaciones de cada tipo de secuencias.
Lugares	Instituciones educativas donde exista un computador.
Datos	Envió y almacenamiento de datos en la EEPROM.
Persona	Usuario.
Otros sistemas de computo o electromecánicos externos al sistema	Prototipo de robot. Pic Basic. Placa del prototipo de robot.
Conceptos de nombres abstractos	Teleoperación. Teleprogramación.
Organizaciones	Laboratorio de robótica.
Eventos	Ingreso, envió y almacenamiento.
Manuales.	Procedimientos lenguaje Pic Basic.

La lista de categorías de conceptos y del análisis de frases nominales de los casos de uso definidos anteriormente generamos una lista de conceptos adecuados para incluirlos en la aplicación del desarrollo de un prototipo de robot con capacidad de hablar y caminar asistido por un computador, la lista está sujeta a la restricción de los requerimientos y simplificaciones que se consideren en el momento.

- Usuario.
- Prototipo de Robot.
- Teleprogramación.
- Teleoperación.

La lista anterior de los nombres de conceptos puede representarse gráficamente vea en anexos la (FIG 2.5) en la notación del diagrama de estructura estática de UML a fin de mostrar la génesis del modelo conceptual.

2.5.1. AGREGACIÓN DE ASOCIACIONES

Es necesario identificar las asociaciones de los conceptos que se requieren para satisfacer los requerimientos de información de los casos de uso en cuestión, los que contribuyen a entender el modelo conceptual.

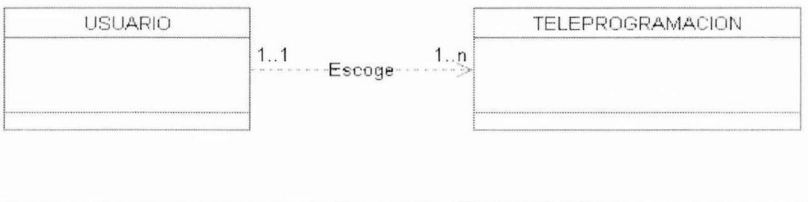


Figura No 2.6 Asociación Usuario Teleprogramación
Fuente: Investigador

Figura No 2.7 Asociación Usuario Teleoperación
Fuente: Investigador

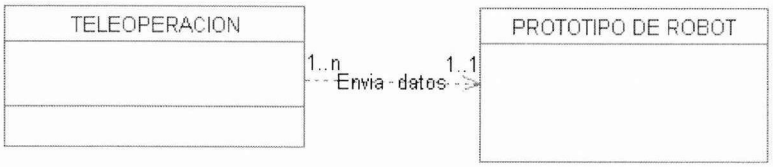


Figura No 2.8 Asociación Teleoperación Prototipo de Robot.
Fuente: Investigador

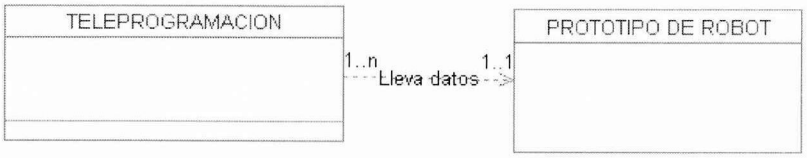


Figura No 2.9 Asociación Teleprogramaciòn Prototipo de Robot
Fuente: Investigador

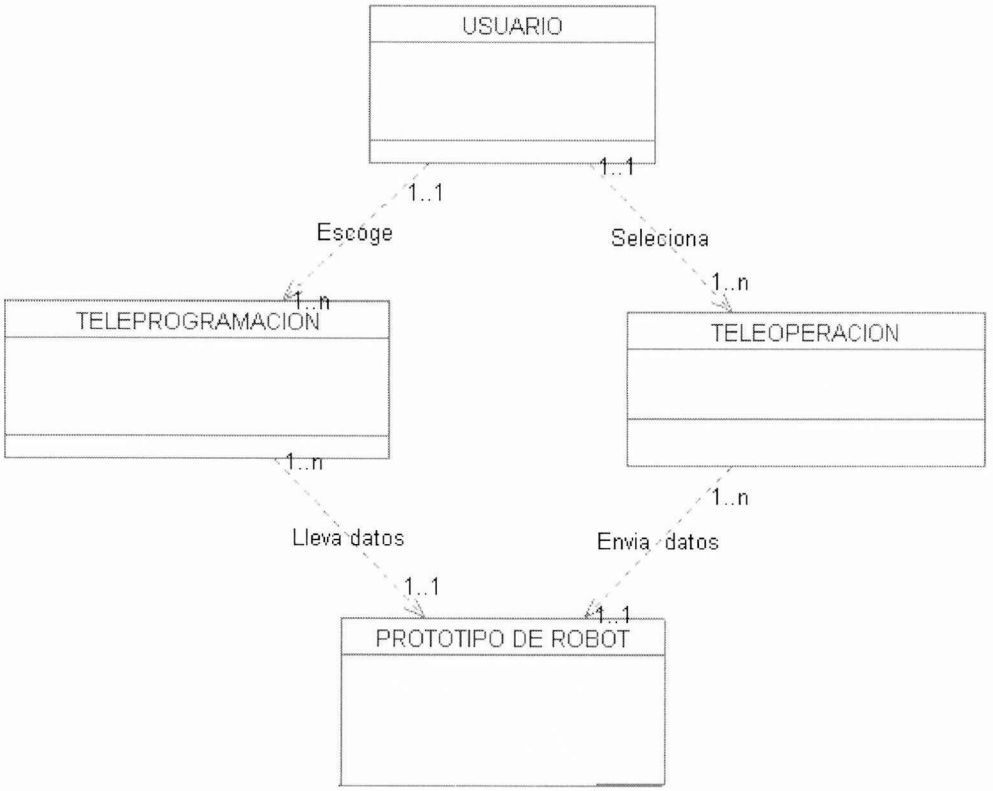


FIGURA No 2.10 MODELO CONCEPTUAL APLICADO AL SISTEMA.

FUENTE: INVESTIGADOR

2.5.2. AGREGACIÓN DE LOS ATRIBUTOS

Es necesario identificar los atributos de los conceptos que se necesitan para satisfacer los requerimientos de información de los casos de uso en cuestión, un atributo es un valor lógico de un dato de un objeto.

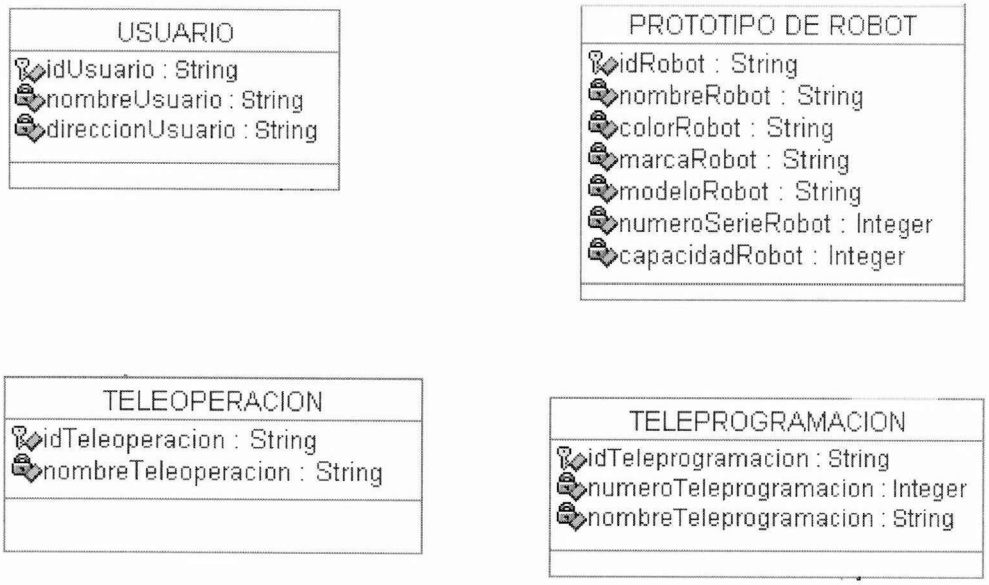


Figura No 2.11 Conceptos y Atributos

Fuente: Investigador

2.6. DICCIONARIO DE DATOS

El glosario es un documento simple en el cual se definen términos, este define todos los términos que requieren explicarse para mejorar la comunicación y menorar el riesgo de malos entendidos.

Un significado uniforme y compartido resulta extremadamente importante durante el desarrollo de las aplicaciones.

Tabla No 1.8 Glosario de Términos

Fuente: Investigador

Término	**Categoría**	**Comentarios**
Teleprogramaciòn	Caso de uso	Permite la iniciación de la secuencia de pasos del prototipo de robot mediante el ingreso de datos.
Teleoperación.	Caso de uso	Inicia la secuencia de pasos del prototipo de robot, mediante la acción del usuario con las flechas de direccionamiento de arriba y de abajo.
Usuario	tipo	Alumnos o docentes que ejecutan el prototipo de robot siendo una aplicación de robótica.
idUsuario	Atributo	Identificación del usuario
nombreUsuario	Atributo	Nombre del usuario
direcciònUsuario	Atributo	Dirección del usuario
Teleprogramaciòn	Tipo	Permite ingresar al método tele_ programación del prototipo de robot, este proceso se lo realiza mediante el ingreso de datos
idTeleprogramaciòn.	Atributo	Identificación de la teleprogramación.
numeroTeleprogramaciòn	Atributo	Cuantos pasos va ha realizar el prototipo de robot.
nombreTeleprogramaciòn	Atributo	Nombre de la secuencia.
Teleoperacciòn	Tipo	Permite ingresar al método de teleoperación mediante las flechas de direccionamiento de arriba y de abajo.
idTeleoperacciòn.	Atributo	Identificación de la Teleoperación
nombreTeleoperacion	Atributo	Nombre de la secuencia.
Prototipo de Robot	Tipo	Hardware que permite realización de pasos
idRobot	Atributo	Identificación del prototipo de robot.

nombreRobot	Atributo	Nombre del prototipo de robot.
colorRobot	Atributo	Color del prototipo de robot.
marcaRobot	Atributo	Marca del prototipo de robot.
modeloRobot	Atributo	Modelo del prototipo de robot.
numeroRobot	Atributo	Numeró de serie del prototipo de robot.
capacidadRobot	Atributo	Capacidad de almacenamiento de memoria que tiene el prototipo de robot.

2.7. ESPECIFICACIONES ADICIONALES.

2.7.1. COMPORTAMIENTO DE LOS SISTEMAS.

El diagrama de la secuencia de un sistema muestra gráficamente los eventos que fluyen de los actores al sistema, la creación de los diagramas de la secuencia de un sistema forma parte de la investigación para conocer el sistema se incluye pues dentro del modelo de análisis.

El UML ofrece una notación con los diagramas de la secuencia que muestran gráficamente los eventos que pasan de los actores al sistema. Previó al inicio del diseño lógico, cómo funciona una aplicación de software es necesario investigar y definir su comportamiento como una

"caja negra", el comportamiento del sistema es una descripción de lo que hace sin explicar la manera en que lo hace, una parte de la descripción es un diagrama de la secuencia del sistema.

2.7.2. DIAGRAMAS DE SECUENCIA DEL SISTEMA.

Los casos de uso indican cómo los actores interactúan con el sistema de software que es en realidad lo que se desea crear, durante la interacción un actor genera eventos dirigidos a un sistema, solicitando alguna operación a cambio.

Conviene aislar y explicar gráficamente las operaciones que un actor solicita a un sistema porque contribuye de manera importante a entender el comportamiento del sistema, el UML incluye entre su notación los diagramas de secuencia que dan una descripción gráfica de las interacciones del actor y de las operaciones a que da origen.

El diagrama de secuencias de un sistema es una representación que muestra en determinado escenario en un caso de uso, los diagramas se centran en los eventos que trascienden las fronteras del sistema y que influyen de los actores a los sistemas como se vera en el diseño de los diagramas de secuencia del sistema del prototipo de robot.

Figura N 2.12 Diagrama de secuencia para el caso de uso: Teleprogramaciòn

Fuente: Investigador

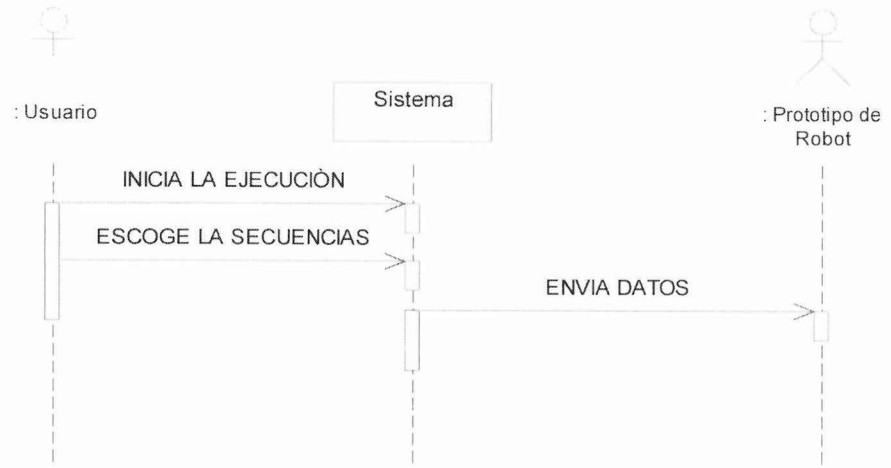
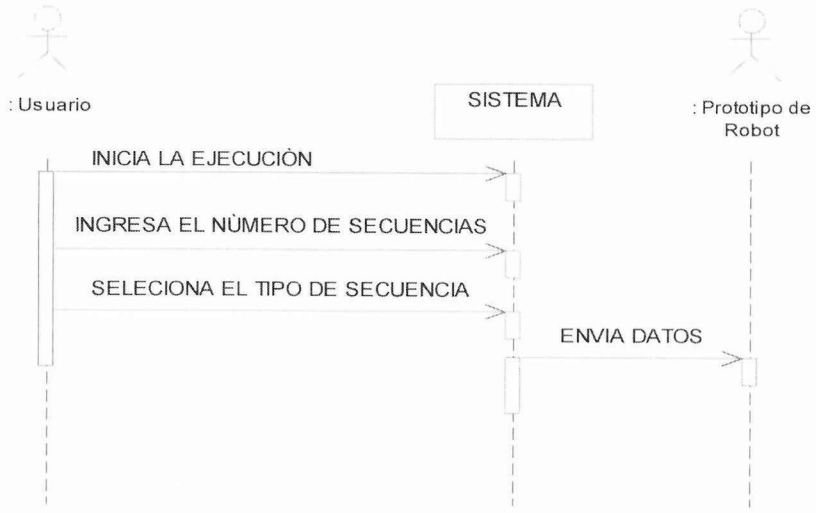


Figura No 2.13 Diagrama de secuencia para el caso de uso: Teleoperación

Fuente: Investigador



2.7.3. CONTRATOS.

Los contratos contribuyen a definir el comportamiento de un sistema, describen el efecto que sobre él tienen las operaciones, el lenguaje UML ofrece un soporte para definir los contratos, permite definir las precondiciones y las poscondiciones de las operaciones.

Los contratos de operación del sistema se elaboran durante la fase de análisis en el ciclo de desarrollo, su preparación depende del desarrollo previo del modelo conceptual de los diagramas de la secuencia del sistema y la identificación de sus operaciones en términos generales, un contrato es un documento que describe lo que una operación se propone lograr, suele redactarse en un estilo declarativo enfatizando lo que sucederá, no cómo se conseguirá; los contratos suelen expresarse a partir de los cambios de estado de las precondiciones y de las poscondiciones, puede elaborarse un contrato para un método de una clase de software o para una operación más global del sistema.

El contrato de operación del sistema describe los cambios del estado del sistema total cuando se llama una de sus operaciones. A continuación se presenta la definición de los contratos que se identifican en el sistema del prototipo de robot con capacidad de hablar y caminar:

Contrato

Nombre:	Introducir Datos (Número de Secuencias).
Responsabilidades:	Capturar el dato para el envió del puerto serial.
Tipo:	Sistema

Referencias	R.1.1, R1.2, R.1.4, R.1.5.
Cruzadas:	Teleprogramación del prototipo de robot para caminar.
Notas:	Utilizar acceso rápido del ingreso de secuencias.
Excepciones:	Números negativos, caracteres y neutro.
Salida:	Visualización de la secuencia de pasos del prototipo de robot.
Precondiciones:	El Pic basic espera una respuesta de la interfaz por parte del usuario para enviar los datos para que se ejecute la secuencia de pasos del prototipo de robot.

Poscondiciones:

- Siempre para el ingreso de datos los números serán enteros positivos o decimales.
- Se ingresa el dato correcto, el usuario predeterminara si para la secuencia en ejecución.
- El sistema envía los datos para que el software y el hardware se interactúen donde el prototipo de robot realice la secuencia de pasos.

Contrato

Nombre:	Manipulación de datos.
Responsabilidades:	Capturar el dato para el envió del puerto serial.
Tipo:	Sistema
Referencias	R.1.3, R.1.4, R.1.5.
Cruzadas:	Teleoperación del prototipo de robot para caminar.
Notas:	Utilizar acceso rápido del ingreso de secuencias.
Excepciones:	Ninguna.

Salida: Visualización de la sec uencia de pasos del prototipo de robot.

Precondiciones: El Pic basic espera una respuesta de la interfaz por parte del usuario para enviar los datos para que se ejecute la secuencia de pasos en el prototipo de robot.

Poscondiciones:

- El usuario realiza la selección mediante las flechas de direccionamiento de arriba y de abajo.
- El sistema permite enviar el dato para que el prototipo de robot realice la secuencia de pasos.

2.8. ANALISIS DE CIRCUITOS.

2.8.1. CIRCUITO DEL PROTOTIPO DE ROBOT

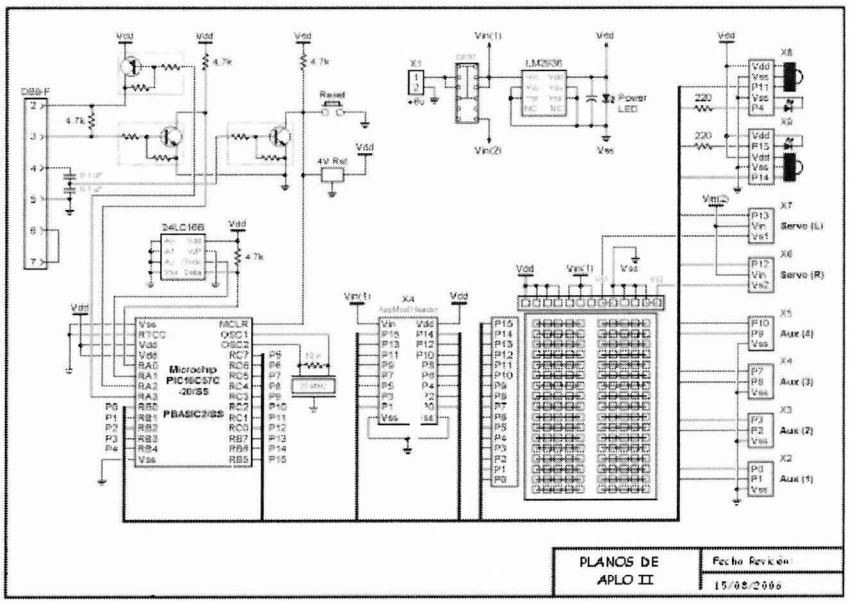


Figura No 2.14: Circuito del Prototipo de robot.

Fuente: Investigador

2.8.2. CIRCUITO IMPRESO DE LA FUENTE DE PODER

El circuito se desarrolló mediante el software Eagle versión 4.03, se debe crear el diagrama PCB, se utilizará los siguientes componentes:

Un puente rectificador de 1 A; un condensador de 1000 uf de 25 V; un regulador LM 317; un transformador de 110 AC a 12 V AC 1A; una resistencia de 270 Ω y una1k Ω.

Luego imprima el circuito para posteriormente elaborarlo como se muestra en la (FIG 2.15).

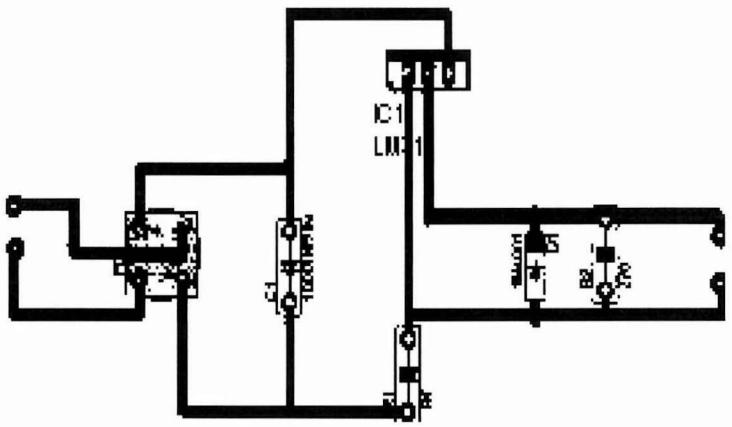


Figura No 2.15: Circuito de la fuente.

Fuente: Investigador

CAPÍTULO III

FASE DE DISEÑO

3.1. INTRODUCCIÓN AL MODELO DE ANÁLISIS Y DISEÑO

En la fase de inicio, del desarrollo se da prioridad al conocimiento de los requisitos, los conceptos, las operaciones relacionadas con el sistema, a menudo la investigación, el análisis se caracterizan por centrarse en cuestiones concernientes a qué, cuáles son los procesos, los conceptos, etc. En el UML hay otros artefactos que sirven para capturar los resultados de una investigación se describe un grupo mínimo de ellos que fueron plasmados en la etapa anterior:

Tabla No 2.1 Del análisis al diseño

Fuente: Investigador

Artefacto de análisis	Preguntas que se contestan
Casos de uso	¿Cuáles son los procesos del dominio?
Modelo conceptual	¿Cuáles son los conceptos, los términos?
Diagrama de las secuencias de un sistema	¿Cuáles son los eventos y las operaciones del sistema?
Contratos	¿Qué hacen las operaciones del sistema?

Durante el ciclo de desarrollo iterativo es posible pasar a la fase de diseño una vez terminados estos documentos del análisis, durante este paso se logra una solución lógica que se funda en el paradigma orientado a objetos; su esencia es la elaboración de diagramas, de interacción que muestran gráficamente cómo los objetos se comunicarán entre ellos a fin de cumplir con los requisitos.

El advenimiento de los diagramas de interac ión nos permite dibujar diagramas de diseño de clases que resumen la definición de las clases (e interfaces) impleméntables en software.

3.1.1. DESCRIPCIÓN DE LOS CASOS REALES DE USO.

Los casos reales de uso presentan un diseño concreto de cómo se realizará el caso, la definición de estos es una de las primeras actividades dentro de un ciclo de desarrollo, su creación depende de los casos esenciales conexos que hayan sido generados antes.

Un caso real de uso describe el diseño concreto del caso de uso a partir de una tecnología particular de entrada y salida así como implementación global por ejemplo: si interviene una interfaz gráfica para el usuario, el caso de uso real incluirá diagramas de las ventanas en cuestión y una explicación de la interacción de bajo nivel con los artefactos de la interfaz.

Caso de uso:	**Teleprogramación.**
Actores:	Usuario, Prototipo de Robot.
Propósito:	Realice una secuencia de pasos el prototipo de robot.

Tipo: Primario.

Descripción: El usuario luego de escoger el método de teleprogramación, selecciona el tipo de secuencia e inmediatamente ingresa el número de pasos a realizar el prototipo de robot.

Referencias cruzadas: R.1.1, R.1.2, R.1.4, R.1.5.

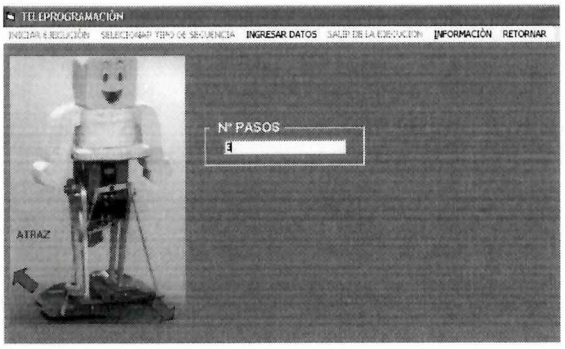


Tabla N. 2.2 Curso normal de los eventos de la teleprogramación. **Fuente: Investigador.**	
Acción del actor	**Respuesta del Sistema**
1. Este caso de uso comienza cuando se ha seleccionado el método de tele_ programación.	
	2. Adicionalmente el sistema realiza el proceso automático, que el prototipo de robot comience a decir una secuencia de frases indicando como es el proceso.
3. El usuario escucha que el prototipo de robot habla.	
4. Posteriormente el usuario inicia la selección de la secuencia de pasos.	

5. El usuario ingresa el número de pasos a realizar el prototipo de robot.	
	6. El sistema se encarga de bloquear el ingreso de caracteres, números negativos, letras. Admitiendo solo números enteros positivos y decimales.
	7. El sistema detecta cuando el número ingresado es correcto.
	8. El sistema envía los datos al prototipo de robot.
9. El usuario puede escoger la opción de parar en cualquier momento dicha ejecución.	
	10. En este momento finalmente el sistema permite la ejecución de la secuencia de pasos del prototipo de robot.
11. El usuario visualiza que el prototipo de robot camina.	
	12. Finalmente el sistema finaliza la ejecución de los procesos.

Caso de uso: **Teleoperación.**

Actores: Usuario, Prototipo de Robot.

Propósito: Realice una secuencia de pasos el prototipo de robot.

Tipo: Primario.

Descripción: El Usuario luego de escoger el método de teleoperación, selecciona el tipo de secuencia que realizara el prototipo de robot.

Referencias cruzadas: R.1.3.R.1.4.R.1.5

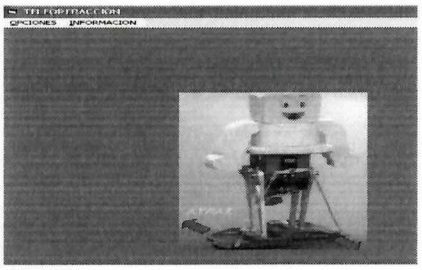


Tabla 2.3 Curso normal de los eventos de la teleoperación. **Fuente: Investigador.**	
Acción del actor	**Respuesta del Sistema**
1. Este caso de uso comienza cuando se ha seleccionado el método de teleoperación	
	2. Adicionalmente el sistema realiza el proceso automático, que el prototipo de robot comience a decir una secuencia de frases indicando como es el proceso.
3. El usuario escucha que el prototipo de robot habla.	
	4. El sistema permite que el usuario teleopere el prototipo de robo mediante las flechas de direccionamiento de arriba y de abajo.
	5. El sistema inmediatamente ejecutará

	las secuencias de pasos para el prototipo de robot.
6. El Usuario visualiza que el prototipo de robot camina.	
	7. Finalmente el sistema finaliza la ejecución de los procesos.

3.1.2. DIAGRAMAS DE CLASES DE DISEÑO.

3.1.2.1. MODELO DE ANÁLISIS Y DISEÑO.

Para el ciclo actual de desarrollo de la aplicación podemos identificar la especificación de las clases de software (y las interfaces) que participan en la solución de software y complementarlas con detalles de diseño, la definición de este tipo de diagrama se lleva a cabo en la fase de diseño del ciclo de desarrollo, su preparación exige crear:

- Diagramas de interacción a partir de ellos el diseñador identifica las clases de software que interviene en la solución así como los métodos de las clases.
- Modelo conceptual: a partir de éste el diseñador agrega detalles a la definición de las clases.

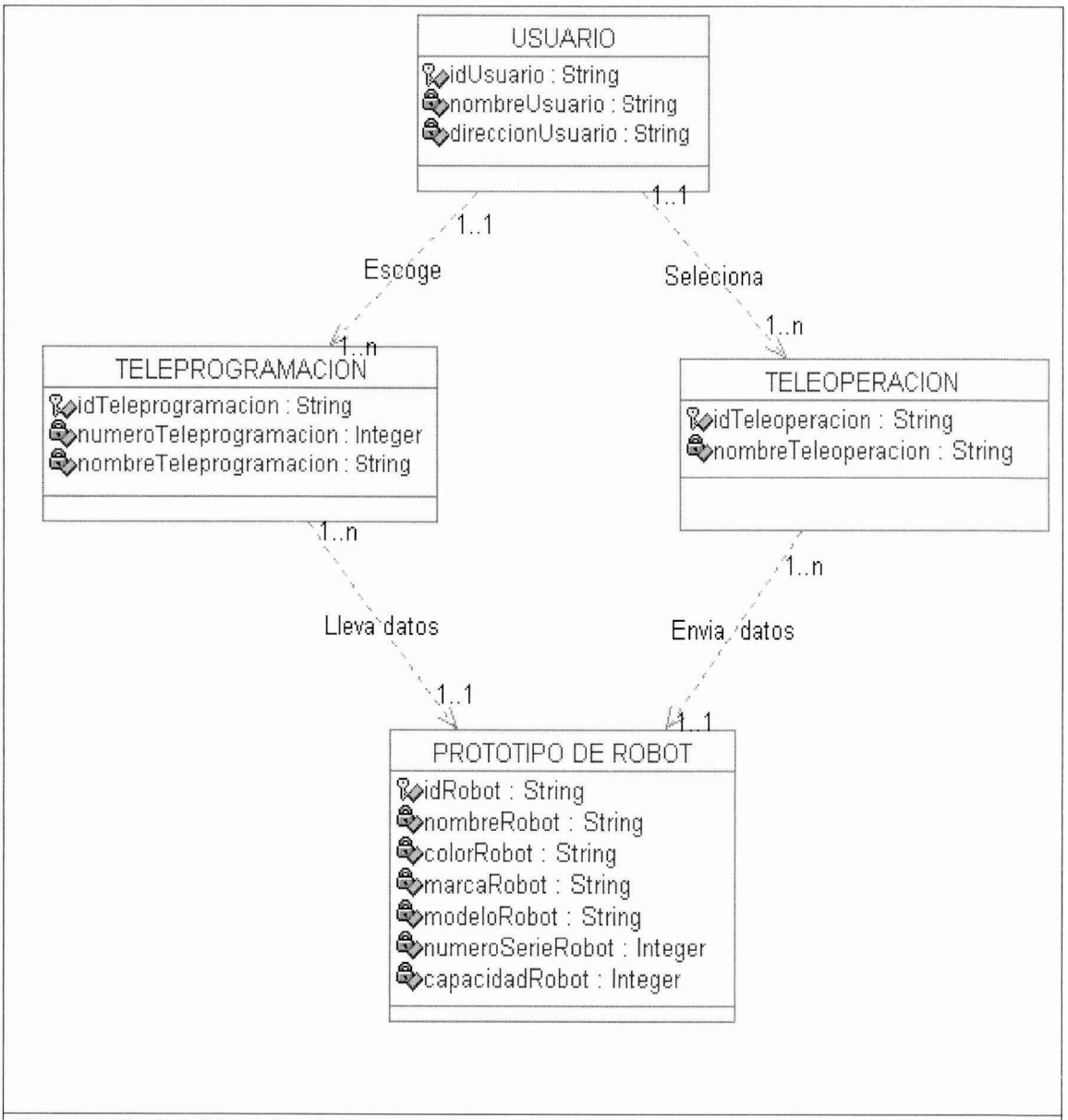


Figura No 3.1 Modelo de Diseño: Diagrama de clases

Fuente: Investigador

3.1.3. DISEÑO DEL PROTOTIPO DE ROBOT

3.1.3.1. DISEÑO Y CONSTRUCIÒN DE PLACAS

3.1.3.1.1. PLACA DE LA FUENTE DE PODER

El diseño del esquema PCB se coloca en la placa de baquelita mediante un marcador indeleble, posteriormente retiramos el cobre de la placa mediante cloruro ferrico, finalmente instalamos y soldamos los componentes de acuerdo al diseño desarrollado.

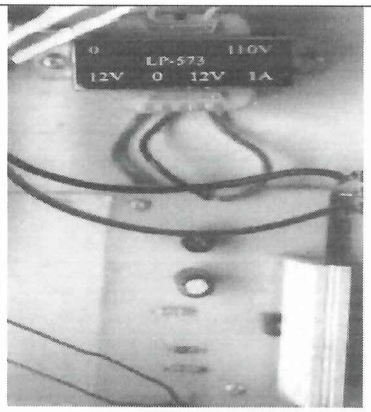


Figura No 3.2:
Placa de la Fuente de Poder.
Fuente: Investigador

3.1.3.1.2. PLACA DEL PROTOTIPO DE ROBOT

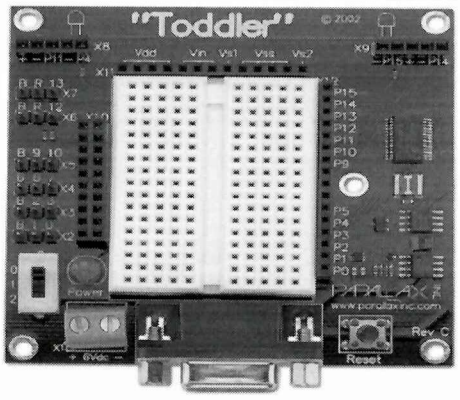


Figura No 3.3: Placa del Prototipo de robot.

Fuente: Investigador

3.1.3.2. CONSTRUCCIÓN DE PROTOTIPO DE ROBOT

3.1.3.2.1. Paso #1: Instalación del Servo de Inclinación en la caja contenedora

Para este paso se requiere las siguientes partes:

- (4) Tornillos de 4/40 3/8"
- (4) Roscas de 4/40.
- (1) Caja contenedora de servos.
- (1) Servo motor.

Debe instalarse el servo en la caja contenedora, éste debe de estar orientado hacia el fondo del contenedor como muestra en el anexo de la (FIG 3.4), ubique el servo en ángulo recto usando cuatro tornillos de 4/40 3/8", las roscas de 4/40 deben quedar en el interior del contenedor, atornille el servo de inclinación en el contenedor.

3.1.3.2.2. Pasó #2: Instalación del Servo de paso en la parte inferior de la caja contenedora.

Para este paso se requiere las siguientes partes:

- (4) Tornillos de 4/40 3/8"
- (4) Roscas de 4/40.
- (1) Servo motor.

En el servo debe estar en la parte inferior del contendor como se muestra en el anexo de la (FIG 3.5), usando cuatro tornillos de 4/40 3/8", las roscas de 4/40 deben quedar en el interior del contenedor, atornille el servo de paso en el contenedor.

3.1.3.2.3. Pasó #3: Centrar Servos eléctricamente

Para este paso se requiere las siguientes partes:

• 1 Fuente.

• (1) Cable Serial.

•Circuito del Microcontrolador de Basic Stamp 2.

Los servos deben ser centrados antes de cualquier ensamblaje, esto permitirá cualquier ajuste de multi-afinación debe ser realizado mediante el lenguaje Pic Basic. Introduzca los conectores de los servos en el circuito del board X6 para el servo de paso y X7 para el servo de la inclinación comenzando desde la etiqueta B el hilo negro de los conectores, luego coloque un alambre en el circuito del board entre el Vs1 y Vss, ponga otro alambre entre Vs2 y Vss, esto conecta las tierras del los servos como se muestra en el anexo de la (FIG 3.6).

Coloque el switch en la posición cero, conecte los terminales de fuente al circuito, el terminal café en el positivo del circuito y el negativo al terminal plomo, use el cable serial para la conexión entre la PC y el circuito del microcontrolador.

El circuito del Apolo 2, consta de tres posiciones:

Posicione O: Ningún Power.

Posicione 1: Impulse a todo excepto los servos.

Posicione 2: Todo se impulsa.

Ponga el interruptor de poder en la posición 2, el próximo paso es programar en el Basic Stamp abra el editor de Windows y escriba o cargue el pedazo siguiente del código que centra ambos servos:

```
' -----[ Titulo ]-------------------------------------------------------
' Programa que permite centrar los servos
' {$STAMP BS2}
' {$PBASIC 2.5}
' -----[ Declaraciones ]-------------------------------------------------
InclinacionServo      CON    12     ' Inclinaciòn del servo P12
PasoServo             CON    13     ' Inclinaciòn del servo P13
' -----[ Inicio del Cuerpo del Programa ]-------------------------------------------------
CENTRAR :                           'Centra los servos con 1500 us de pulsaciones.
  PULSOUT  InclinacionServo,750
  PULSOUT  PasoServo, 750
  PAUSE 25                          ' PAUSA DE 25 ms
GOTO CENTRAR                        ' REPITE
```

Estando en Basic Stamp corra el programa o seleccione este icono ▶ , el programa centrara los servos, luego coloque en la posición de reposo, desconecte la fuente y los conectores del prototipo de robot.

3.1.3.2.4. Pasó #4: Colocación de los cuernos de los servos

Para este paso se requiere las siguientes partes:

• (2) Cuernos de servos

• (2) Tornillos negros pequeños para sostener el cuerno de los servos.

Coloque los cuernos del servo estos deben instalarse en forma recta como muestra en el anexo de la (FIG. 3.7), esto permitirá que se centren correctamente luego atornille cada servo.

3.1.3.2.5. Pasó #5: Instalación de los guardianes de alambre de latón en el paso largo del servo

Para este paso se requiere las siguientes partes:

• (2) Guardianes de alambre de latón (Conectores E-Z).

Ate a los dos guardianes de alambre de latón en los agujeros extremos del cuerno del servo de paso use un alicate el cual permita apretar el caucho hacia el poste del guardián de alambre de latón como se muestra en el anexo de la (FIG 3.8), ponga los dos tornillos pequeños en los hilos del latón.

3.1.3.2.6. Pasó #6: Instalación del plato en la cima de la caja protectora de los servos

Para este paso se requiere las siguientes partes:

• Plato del prototipo de robot

• (4) Tornillos de 4/40 3/8"

• (4) Roscas de 4/40

Instale el plato encima de la caja protectora de los servos, gire al revés la caja protectora como se muestra en el anexo de la (FIG 3.9), coloque el tornillo 4/40 3/8" en los agujero y las rosca de 4/40, deben quedar en el interior del contenedor, atornille al plato, repita el proceso para los otros agujeros.

3.1.3.2.7. Pasó #7: Instalación de las varas en la caja protectora de los servos para las Piernas

En este paso se requiere las siguientes partes:

- (2) Varas de acero largas de 3/16" 3".
- (4) Rodelas plásticas.

Coloque en el interior de los agujeros de la caja protectora las dos varas de 3/16" 3", como se muestra en el anexo de la (FIG 3.10), ponga las rodelas encima de cada vara.

3.1.3.2.8. Pasó #8: Colocación de las piernas en la caja protectora de los servos

Se requiere las siguientes partes:

- (4) Piernas.

Coloque las cuatro piernas hacia los extremos de las varas que van a través de la caja protectora como se muestra en el anexo de la (FIG 3.11).

3.1.3.2.9. Pasó #9: Aseguramiento de las piernas con los collares

En este paso se requiere las siguientes partes:

- (4) Collares de 3/16"
- (1) Llave hexadecimal en L.

Coloque los collares 3/16" en la vara de acero apriete cada collar con la llave haxadecimal como se muestra en el anexo de la (FIG 3.12).

3.1.3.2.10. Pasó #10: Ensamble de las piernas.

Para este paso se requiere las siguientes partes:

- (2) Guardianes del alambre plásticos 4/40 (Conectores E-Z)
- (2) Tornillos 3/8"4/40
- (2) Alambres de ángulo recto de latón

Primero, coloque 38" 4/40 a través de los agujeros en la pierna izquierda trasera como se muestra en el anexo de la (FIG 3.13) ajuste con un desarmador o a mano el tornillo. Repita el proceso para la pierna trasera derecha para este proceso ruede el cuerpo a 180^0.

Usando dos latones del ángulo recto de alambre resbale el extremo recto a través del guardián de alambre de latón en el agujero como se muestra en el anexo de la (FIG 3.14).

Inserte el extremo corto a través de la cima del ángulo recto de plástico, atornille el tornillo para sostener el alambre, ajuste las uniones para que las piernas son verticales, no sesgado esto podría afectarle al caminar, repita para el otro el lado trasero.

3.1.3.2.11. Pasó #11: Colocación de los Tobillos

En este paso se requiere las siguientes partes:

- (2) Tobillos.
- (4) Tornillos de 4/40 ¼.

Ate los tobillos a las piernas usando cuatro tornillos de 4/40 ¼ como se muestra en el anexo la (FIG. 3.15), la parte más larga del tobillo debe orientarse hacia la parte de atrás del la caja contenedora.

3.1.3.2.12. Pasó #12: Colocación de los Pies

Para este paso se requiere las siguientes partes:

- (2) Pies (Derecho /Izquierdo).
- (4) Tornillos 4/40
- (1) Llave hexadecimal en L 3/32"

Coloque el pie en el tobillo como se muestra en el anexo de la (FIG 3.16), si es demasiado firme, ligeramente doble las etiquetas de los pies afuera, ate el tobillo izquierdo al pie izquierdo usando los tornillos 4/40 y ajuste con la llave 3/32, estos tornillos tienen una cabeza redonda que actúa como una superficie productiva en el pie del prototipo de robot, repita para el lado correcto, asegure hacer una inclinación para cada pie antes de proceder al próximo paso.

3.1.3.2.13. Pasó #13: Instalación de las Junturas en los pies

Para este paso se requiere las siguientes partes:

• (4) Junturas (viene con dos roscas).

Instale una juntura en el agujero extremo de cada pie con su respectiva rosca como se muestra en el anexo de la (FIG 3.17), apriete con un alicate la otra rosca que queda en el interior del pie, no enganche la juntura con los alicates puede causarle daño, instale dos junturas en el servo de inclinación controle el cuerno como se muestra en el anexo de la (FIG 3.18).

3.1.3.2.14. Pasó #14: Instalación de Varas

Para este paso se requiere las siguientes partes:

• (2) Varas de aceros de diámetro 5.4", consta de 2/56 hilo en cada extremo,

• (4) Junturas de plásticas de 2/56.

Coloque las junturas en las varas de acero en cada uno de sus extremos, luego posicione las varas en las junturas del servo de inclinación y los otros extremos coloquen en las junturas de los pies del prototipo de robot como se muestra en el anexo la (FIG 3.19).

3.1.3.2.15. Pasó #15: Instalación de los parlantes y cuerpo elemental.

Para este paso se requiere las siguientes partes:

- (2) Parlantes
- (1) Taipe.
- (1) Cuerpo elemental.
- (4) Tornillos de 3/16 x 12", incluido sus roscas.

Bien centrado coloque los parlantes en el plato de la caja protectora de los servos, asegure con taipe, coloque el cuerpo elemental sobre los parlantes y el plato, sujete con los tornillos el cuerpo como se muestra en la (FIG 3.20).

3.2. ALGUNOS ASPECTOS DEL DISEÑO DEL SISTEMA.

Un sistema se compone de subsistemas uno de los cuales son los objetos del dominio, un sistema ordinario de información ha de conectarse a la interfaz del usuario y a un mecanismo de almacenamiento. Una arquitectura común de los sistemas de información que abarca una interfaz para el usuario y el nivel físico que consta de: almacenamiento, E/S de datos, el prototipo de robot se conoce con el nombre de arquitectura de tres capas, se realiza una descripción clásica de las tres capas verticales:

1. Presentación: ventanas etc.
2. Lógica de aplicaciones: tareas y reglas que rigen el proceso.
3. Nivel Físico: se encuentra el almacenamiento, entrada y salida de datos y el prototipo de robot.

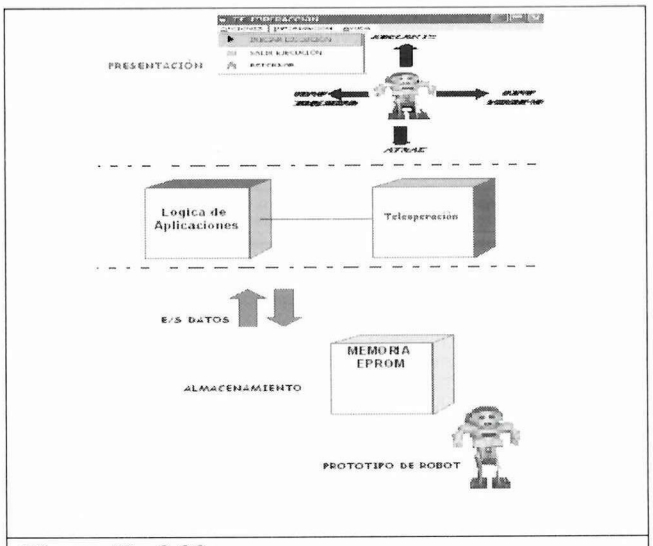


Figura No 3.22:
Vista clásica de una arquitectura de 3 capas.
Fuente: Investigador

3.3. MODELO DE DESPLIEGUE

Una vez concluidos los diagramas de clase del diseño, destinados al ciclo de desarrollo actual en la aplicación, se quiere reducir el riesgo y aumentar la probabilidad de conseguir una aplicación adecuada el desarrollo debería basarse en un suficiente modelado del análisis y diseño antes de iniciar la codificación presentamos el diagrama de despliegue.

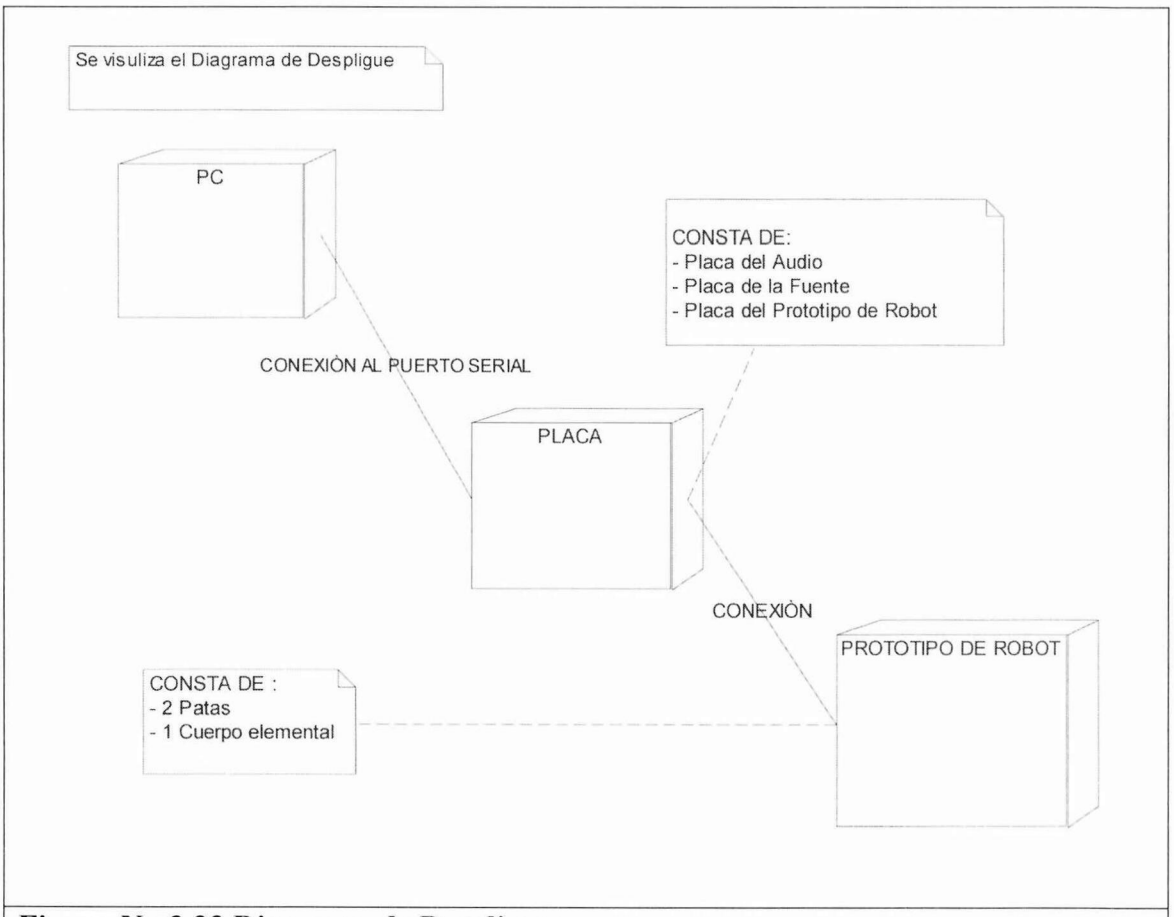


Figura No 3.23 Diagrama de Despliegue

Fuente: Investigador

3.4. CASOS DE PRUEBAS

Una vez desarrollado el sistema y el hardware se procede a realizar los pasos para el funcionamiento del prototipo de robot:

1. Instalado correctamente todas las conexiones entre la PC y el prototipo de robot.

2. Se abre el editor de Basic Stamp para la carga del programa o se copia el siguiente código

```
'SECUENCIA DE PASOS DEL ROBOT
' {$STAMP BS2}
' {$PBASIC 2.5}
I VAR Nib
J VAR Word
C_INCLI VAR Word
C_PASO VAR Word
NUEVO_VALOR VAR Word
DX VAR J
MX VAR Word
CAMINA_ADELANTE DATA 2,3,0,4,1,255
GIRO_IZ DATA  0,5,1,3,2,3,0,5,255
CAMINA_ATRAZ DATA   2,5,0,4,1,255
GIRO_DR DATA   2,3,1,5,0,3,1,5,255
A VAR Byte
Main_Program :
SERIN 16,16468,[A]
GOSUB RESTABLECER

          'DEBUG " RESULTADO= " , DEC CAMINA_ADELANTE,CR

          'DEBUG "Movement routine = ", DEC Mx,CR

IF(A=49)THEN
MX= CAMINA_ADELANTE
GOSUB MOVIMIENTO
ENDIF
IF (A=50) THEN
MX= CAMINA_ATRAZ
GOSUB MOVIMIENTO
ENDIF
IF(A=51) THEN
MX=GIRO_IZ
GOSUB MOVIMIENTO
ENDIF
IF(A=52 )THEN
MX= GIRO_DR
GOSUB MOVIMIENTO
ENDIF
GOSUB Main_program
END
MOVIMIENTO :
READ MX, DX
MX=MX+1
```

```
IF DX=255 THEN MOVIMIENTO_HACE
GOSUB HACE_MOVIMIENTO
GOTO MOVIMIENTO
HACE_MOVIMIENTO:
BRANCH DX,[INCLI_IZ, INCLI_CR, INCLI_DR, PASO_IZ, PASO_CR, PASO_DR]
MOVIMIENTO_HACE:
RETURN
'---------------INCLINACIÒN IZQUIERDA-----------------
INCLI_IZ:
NUEVO_VALOR=880
GOTO MOVIMIENTO_INCLINACION
'---------------INCLINACIÒN DERECHA----------------------
INCLI_DR :
NUEVO_VALOR=620
MOVIMIENTO_INCLINACION:
FOR J=C_INCLI TO NUEVO_VALOR STEP 5
PULSOUT 13, J
PULSOUT 12, C_PASO
PAUSE 25
NEXT
C_INCLI=NUEVO_VALOR
RETURN
'--------------INCLINACIÒN CENTRO ----------------------------
 INCLI_CR:
NUEVO_VALOR=750
GOTO MOVIMIENTO_INCLINACION
'-------PASO CENTRO  ----------------------------------------
PASO_CR:
NUEVO_VALOR=750
GOTO MOVIMIENTO_PASO
'------------------PASO DERECHO-------------------------------
PASO_DR:
NUEVO_VALOR= 650
GOTO MOVIMIENTO_PASO
'------------PASO IZQUIERDO-------------------------------------
PASO_IZ:
NUEVO_VALOR=850
MOVIMIENTO_PASO:
FOR J=C_PASO TO NUEVO_VALOR STEP 5
PULSOUT 13,C_INCLI
PULSOUT 12,J
PAUSE 25
NEXT
C_PASO = NUEVO_VALOR
RETURN
RESTABLECER:
C_INCLI=750
C_PASO=750
FOR J=1 TO 100 STEP 5
PULSOUT 13,750
```

```
PULSOUT 12,750
PAUSE 25
NEXT
RETORNA:
RETURN
```

Proceda a ejecutar la secuencia de instrucciones, se verifica si no tiene errores.

3. Ingrese al sistema y seleccione una de las dos opciones, sea la teleoperación o tele-programación, el usuario escuchara una serie de palabras dichas por el prototipo de robot.

4. Visualizará el usuario la secuencia ingresada o teleoperada, donde el prototipo de robot caminará como se muestra en la (figura 3.24).

5. Realizados los numérales anteriores el usuario terminara la ejecución del sistema y desconectara cuidadosamente el prototipo de robot.

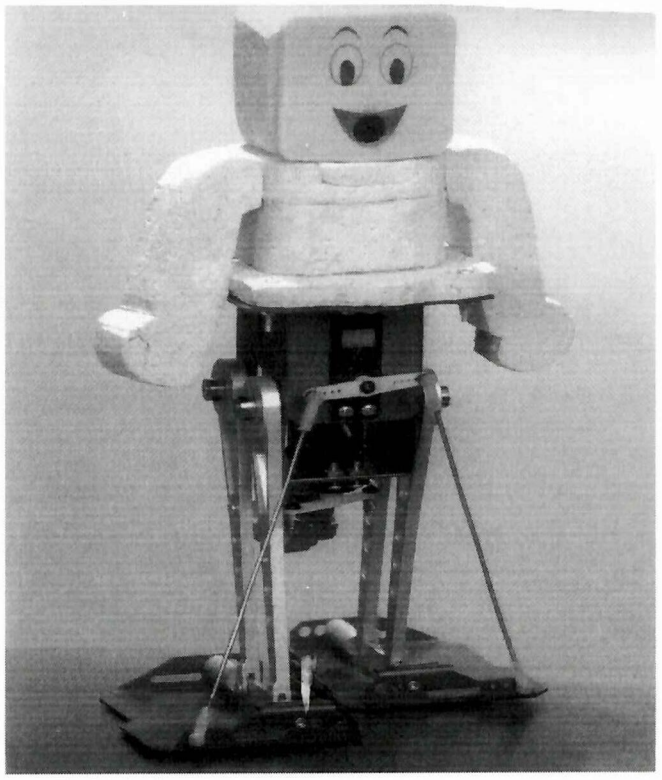


FIGURA Nº 3.24:
PRIMER PASO.
FUENTE: INVESTIGADOR

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

- Se construyó un prototipo de robot con capacidad de hablar y caminar asistido por un computador, una aplicación de robótica para la Universidad Técnica de Cotopaxi.

- La determinación de los requerimientos: materiales, estructurales, electrónicos, y de software nos permitió diseñar el prototipo de robot.

- Se creó un sistema adecuado que permite ser asistido por computador el prototipo de robot.

- El funcionamiento del prototipo de robot se probó que puede caminar y hablar mediante un sistema adecuado.

- Se ha demostrado que las ciencias de la informática, la electrónica están íntimamente ligadas al punto que el software, el hardware no pueden funcionar si no se complementan.

RECOMENDACIONES

- Poner en práctica el sistema, el prototipo de robot construido como material didáctico para los estudiantes de la Carrera de Ciencias de la Ingeniería y Aplicadas de la Universidad Técnica de Cotopaxi.

- Antes de iniciar la ejecución del sistema, se debe almacenar las instrucciones del Pic Basic en la memoria eeprom del prototipo de robot.

- Realizar todas las conexiones correctamente del computador al prototipo de robot para evitar malos funcionamientos.

- Proyectos de esta naturaleza rompen los esquemas tradicionales sobre los temas de tesis, debe ser el punto de partida para la creación de nuevas tecnologías que permita el desarrollo y progreso de nuestro país.

BIBLIOGRAFÍA

CONSULTADA

- OLLERO, Aníbal (2001); Robótica Manipuladores y robot móviles, Alfomega, Primera Edición, Madrid-España, Págs. 1-37.
- URQUIZO, Patricio (2001); Como Realizar la tesis o una investigación, Graficas Riobamba, Primera Edición, Quito- Ecuador, Pág. 1-124.
- AKEEL, H.A., RUTLEDGE, G.J. (2000): "Technological Enhancements and Their Effect on Price/Performance Indicators of Industrial Robots", in (IFR) (2000): World Robotics 2000, New York/Geneva, United Nations, pp. XIV XX.
- KINGHT, A.L. (1989): "Robots y maquinaria de producción automática", en OIT (1989): Enciclopedia de salud y seguridad laboral en el trabajo, Madrid, Ministerio de Trabajo y Seguridad Social, Pág. 2143-2147.
- LÓPEZ PELÁEZ, A. (1996): "El trabajo robotizado: perspectivas sobre la producción industrial en la sociedad tecnológica emergente", en Sistema. Revista de Ciencias Sociales, n° 135, Pág. 75-104.
- LÓPEZ PELAÉZ, A. (1997): "Robótica", en Enciclopedia Universal Espasa Calpe. Apéndice 1995-1996, Madrid, Espasa Calpe, Págs. 35.
- LÓPEZ PELÁEZ, A. (1998): "Los procesos de robotización y sus impactos sociales", en Tezanos Tortajada, J.F. y Sánchez Morales, M.R. (1998): Tecnología y Sociedad en el nuevo siglo. Segundo Foro sobre Tendencias Sociales, Madrid, Sistema, Págs. 701-730.

- LÓPEZ PELÁEZ, A. (2000a): Impactos de la robótica y la automatización avanzada en el trabajo. Estudio Delphi, Madrid, Sistema.
- LÓPEZ PELÁEZ, A. (2000b): "Tendencias en Robótica y Automatización Avanzada. ¿Hacia un nuevo modelo de trabajo?", en Tezanos Tortajada, J.E (ed.) (2000): Escenarios del nuevo siglo. Cuarto Foro sobre Tendencias Sociales, Madrid, Sistema, Págs. 171-196.
- López PELÁEZ, A. (2000c): "Prospectiva, Robótica Avanzada y Salud Laboral", en Prevención, Trabajo y Salud. Revista del Instituto Nacional de Seguridad e Higiene en el Trabajo, n° 6, Págs. 14-21.
- LÓPEZ PELÁEZ, A. (2000d): "Towards a new work pattern? Trends of Automation and Robotics Systems in manufacturing and services", in Robotics, (Journal of International Federation of Robotics), n° 40, Pags. 8-10.
- www.jameco.com

CITADA

- ULLOA, Enríquez (2002); Plan Estratégico de Desarrollo 2003-2006, Latacunga-Ecuador, Págs. 7- 24 -51.
- ULLOA, Enríquez (2000); Estatuto Orgánico Sustitutivo, Latacunga –Ecuador, Págs. 10 -11.
- http://www.gio.gov.tw/info/noticia97/2000/14/p3.htm
- http://www.mtas.es/insht/revista/A_24_ST02.htm
- http://www.ugr.es/~jsenso/eunite/robots2.html

ANEXOS

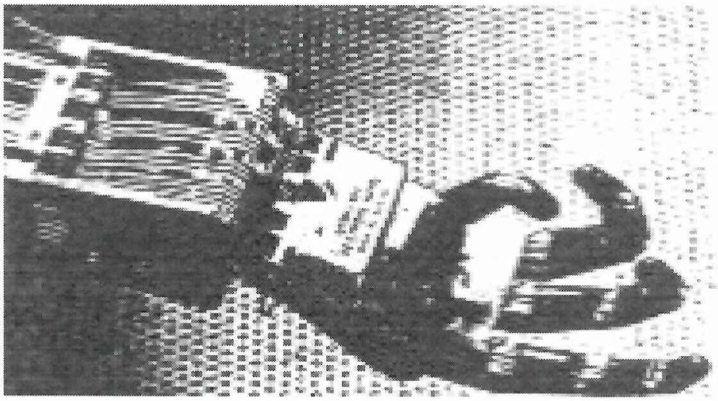


FIGURA. 1.1: MANO ROBÒTICA DESARROLLADA EN LA UNIVERSIDAD DE UTA.

Fuente: Investigador

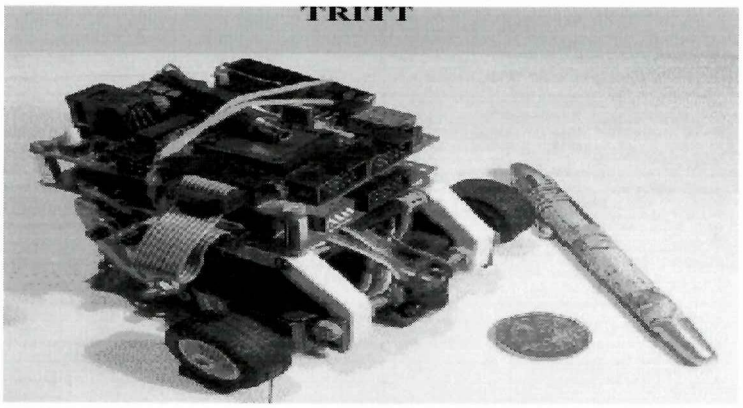


FIGURA. 1.2: ROBOT TRITT CON RUEDAS

Fuente: Investigador

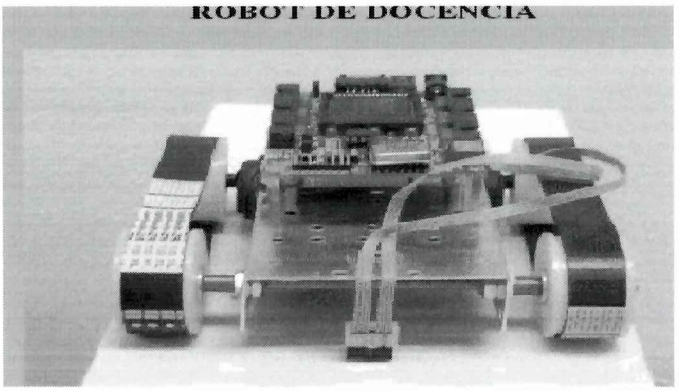


FIGURA. 1.3: ROBOT DE DOCENCIA CON ORUGAS

Fuente: Investigador

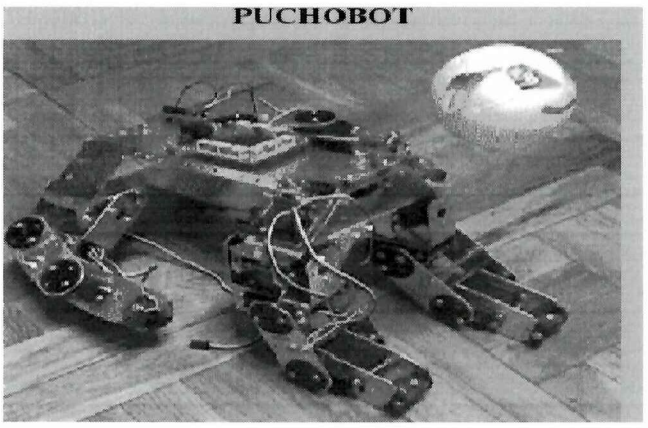


FIGURA. 1.4: ROBOT PUCHOBOT CON PATAS

Fuente: Investigador

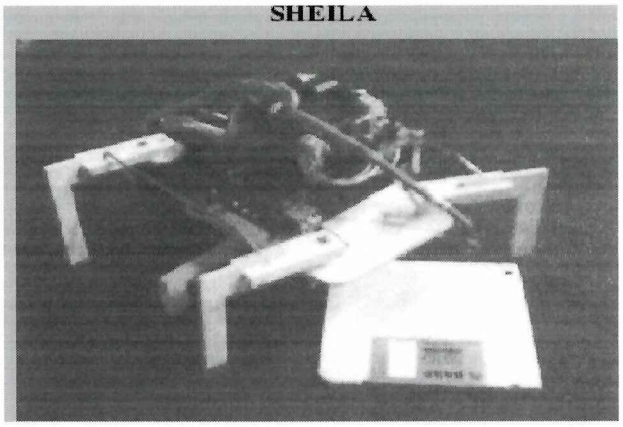


FIGURA. 1.4: ROBOT SHEILA CON PATAS

Fuente: Investigador

FIGURA. 1.5: OTRO TIPO DE ROBOTS

Fuente: Investigador

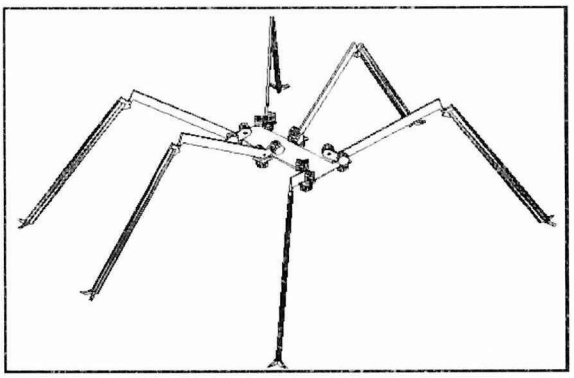


FIGURA. 1.6: ROBOTS CON CUATRO PATAS

Fuente: Investigador

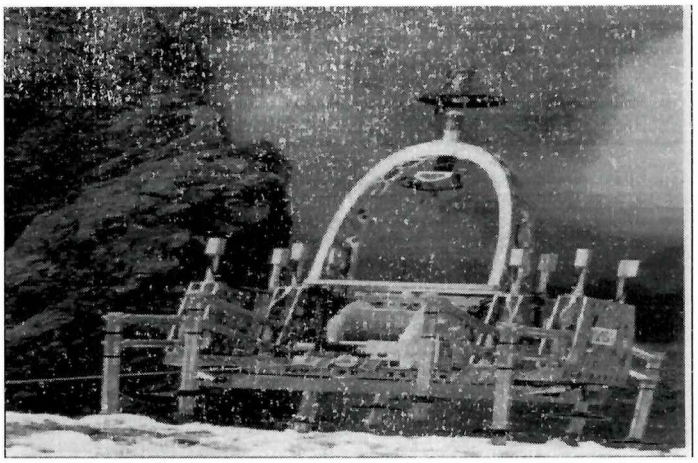


FIGURA. 1.7: ROBOTS CON OCHO PATAS

Fuente: Investigador

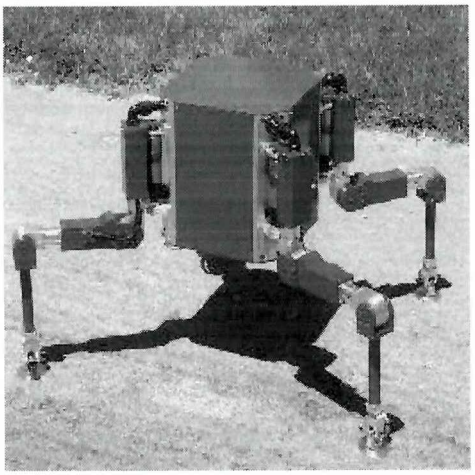


FIGURA. 1.8: ROBOTS CUADRÚPEDOS 4 PATAS

Fuente: Investigador

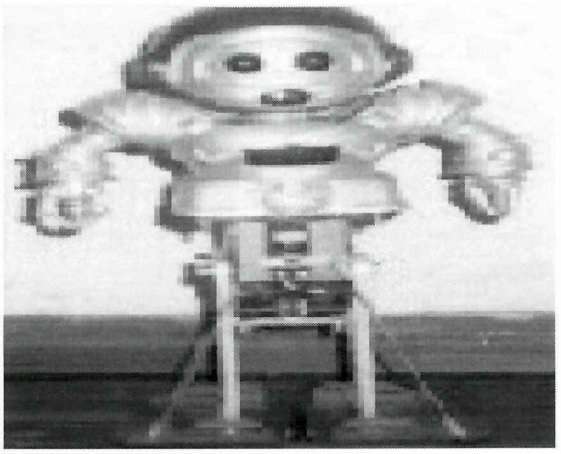


FIGURA. 1.9: ROBOTS BÍPEDOS

Fuente: Investigador

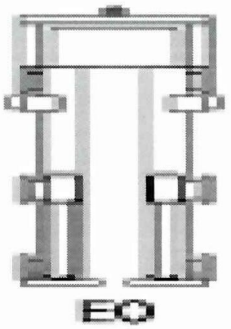


FIGURA. 1.10: ROBOT BÍPEDOS EO

Fuente: Investigador

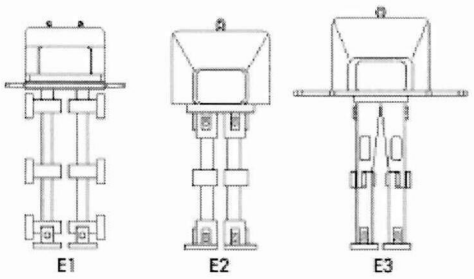


FIGURA. 1.11: VERSIONES DE ROBOT BÍPEDOS

Fuente: Investigador

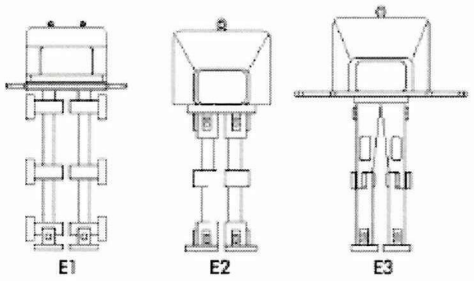


FIGURA. 1.11: VERSIONES DE ROBOT BÍPEDOS

Fuente: Investigador

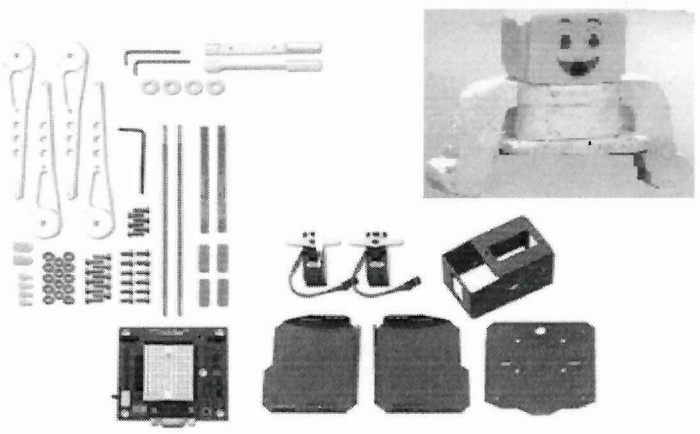


FIGURA. 2.1 ELEMENTOS DEL PROTOTIPO DE ROBOT.

Fuente: Investigador

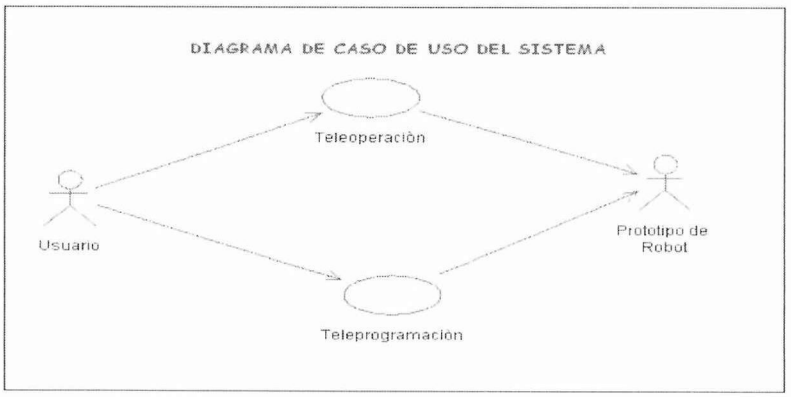


FIGURA No2.2: DIAGRAMA DE CASOS DE USO DE LA APLICACIÓN.
Fuente: Investigador

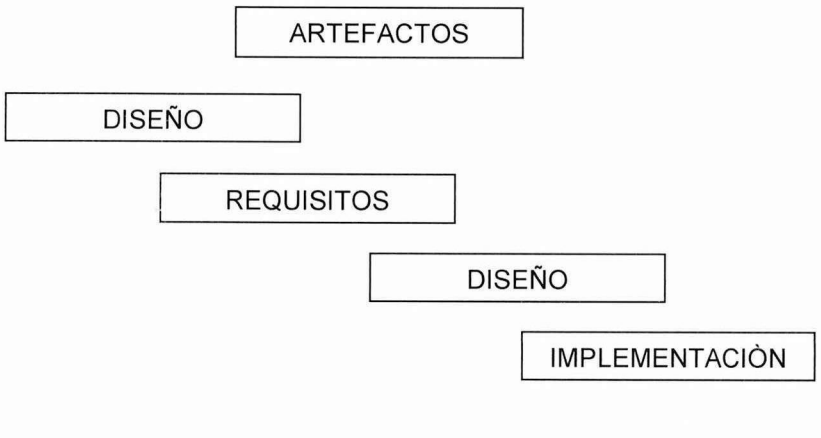


FIGURA No 2.3 ARTEFACTOS DEL CICLO DE DESARROLLO.

Fuente: Investigador

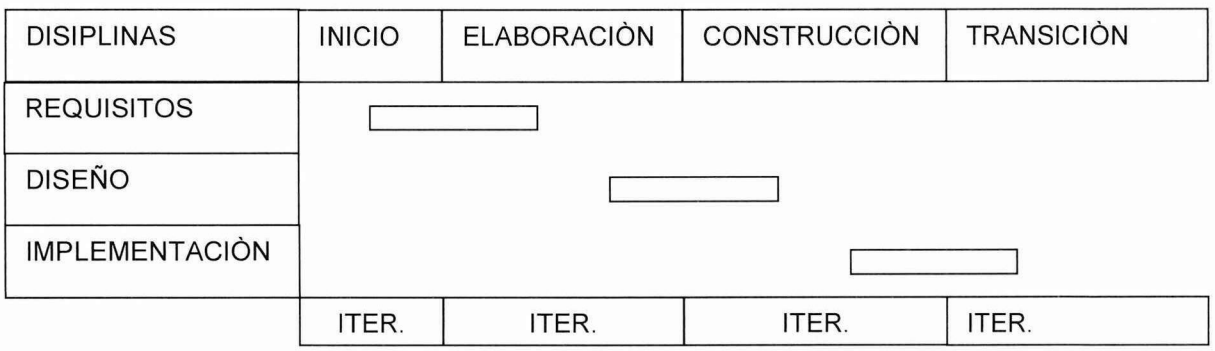


FIGURA No2.4 CICLO DE DESARROLLO DEL SISTEMA.

FUENTE: INVESTIGADOR

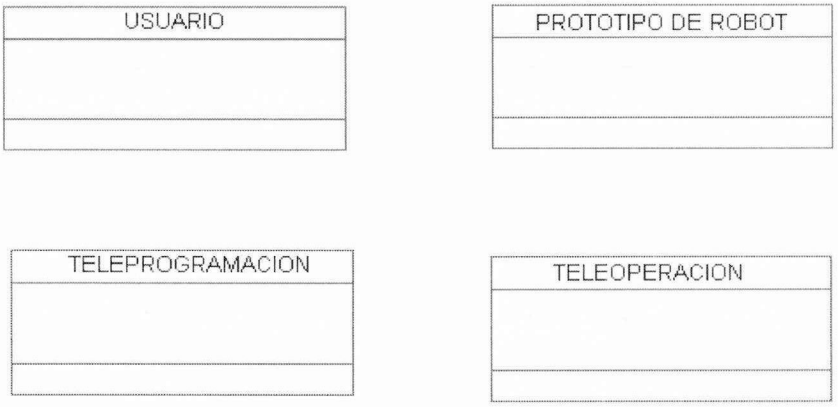


Figura No 2.5 MODELO CONCEPTUAL INICIAL DEL SISTEMA.

Fuente: Investigador

FIGURA No 3.4 :
INSTALACIÓN DEL SERVO DE LA INCLINACIÓN
FUENTE: INVESTIGADOR

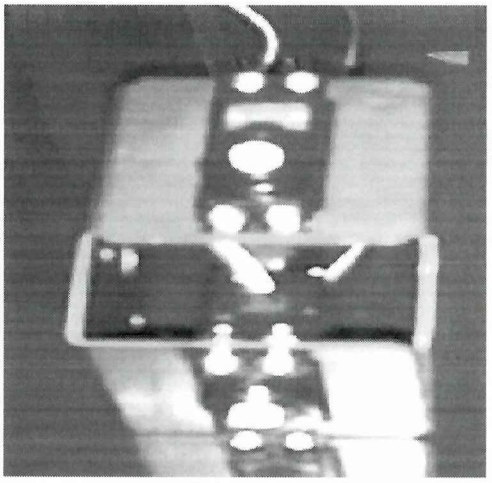


FIGURA No3.5: INSTALACIÓN DEL SERVO DE PASO.
FUENTE: INVESTIGADOR

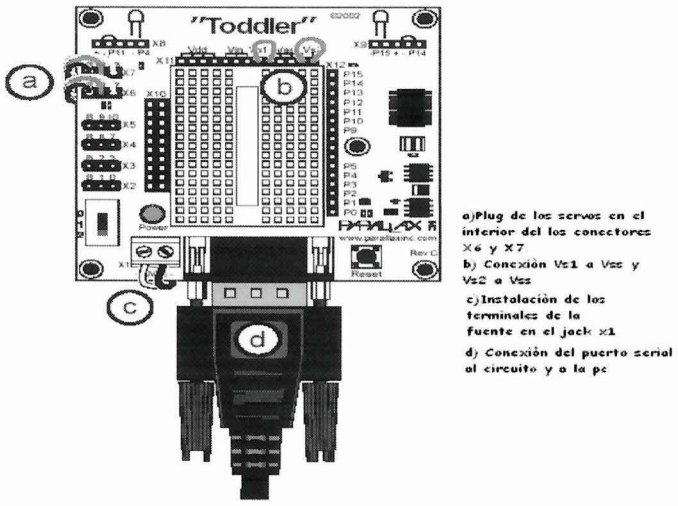


FIGURA No3.6 CENTRAR SERVOS.
FUENTE: INVESTIGADOR

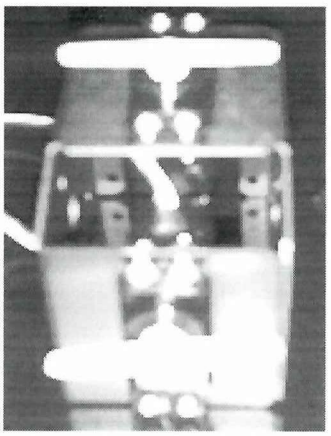


FIGURA N o3.7 INSTALACIÓN DE LOS CUERNOS DE LOS SERVOS.
FUENTE: INVESTIGADOR

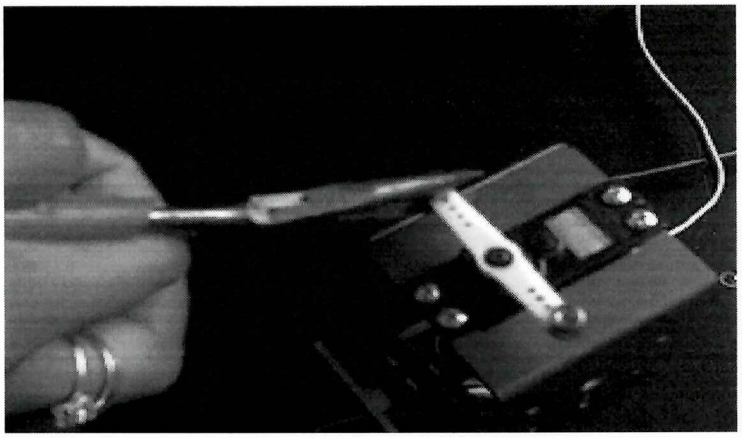


FIGURA No3.8 :
INSTALACIÓN DE LOS GUARDIANES DE ALAMBRE DE LATÓN
FUENTE: INVESTIGADOR

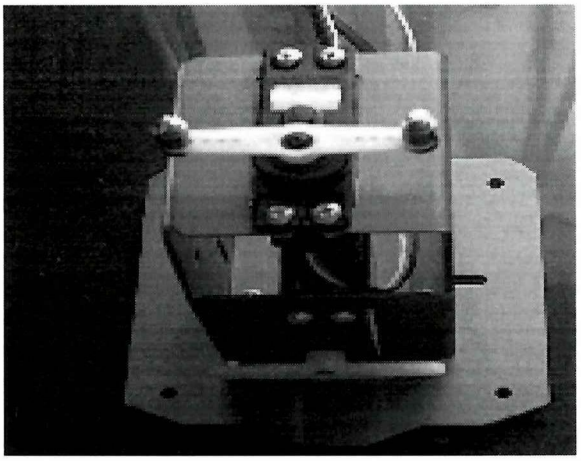


FIGURA No3.9 : INSTALACIÓN DEL PLATO
FUENTE: INVESTIGADOR

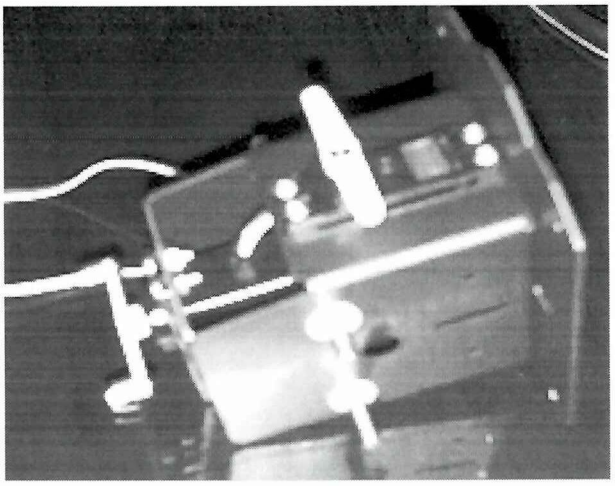


FIGURA No3.10 :
INSTALACIÓN DEL LAS VARAS
FUENTE: INVESTIGADOR

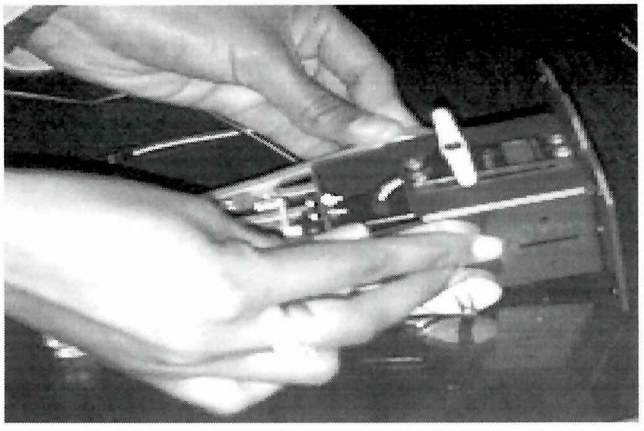


FIGURA No3.11:
COLOCACIÓN DE LAS PIERNAS.
FUENTE: INVESTIGADOR

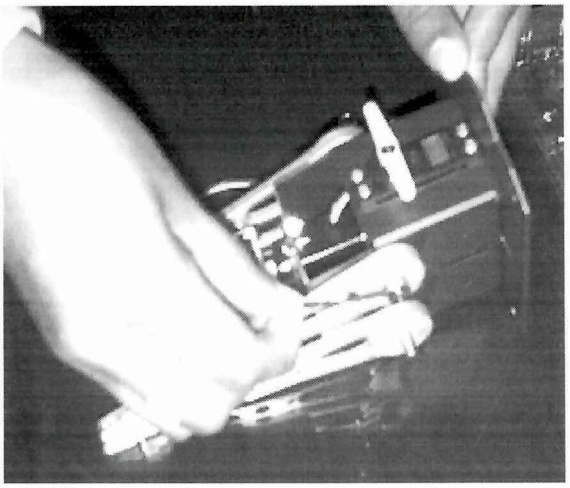


FIGURA No3.12:
ASEGURAMIENTO DE LAS PIERNAS CON LOS COLLARES
FUENTE: INVESTIGADOR

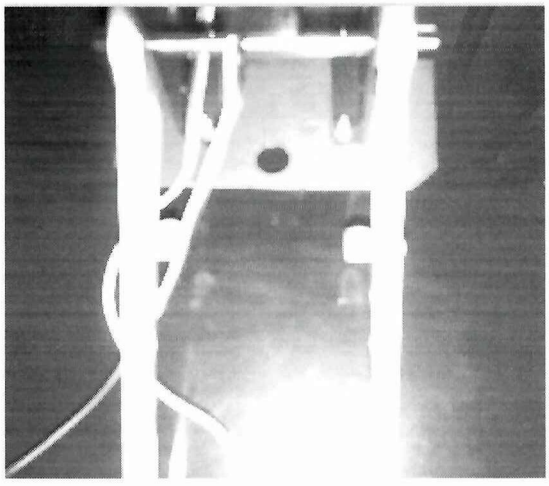


FIGURA No 3.13:
A) ENSAMBLE DE LAS PIERNAS.
FUENTE: INVESTIGADOR

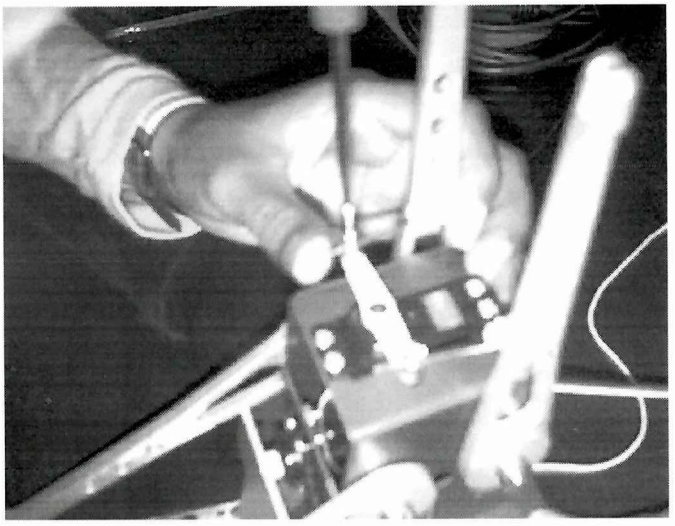


FIGURA No 3.14:
B) ENSAMBLE DE LAS PIERNAS.
FUENTE: INVESTIGADOR

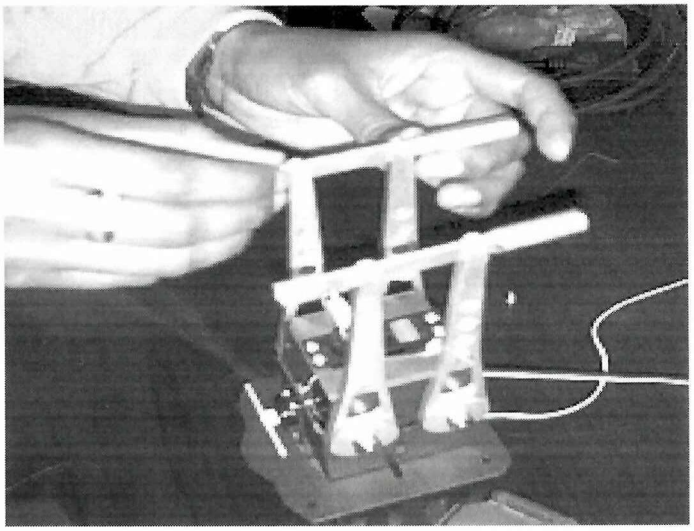


FIGURA No 3.15: COLOCACIÓN DE LOS TOBILLOS.
FUENTE: INVESTIGADOR

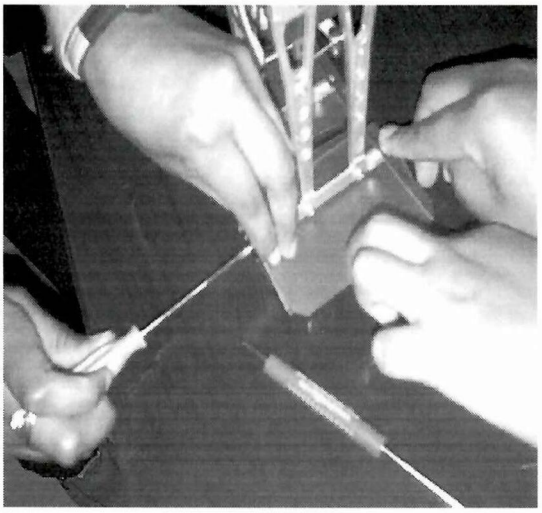


FIGURA No3.16: COLOCACIÓN DE PIES.
FUENTE: INVESTIGADOR

FIGURA No 3.17:
A) INSTALACIONES DE LAS JUNTURAS.
FUENTE: INVESTIGADOR

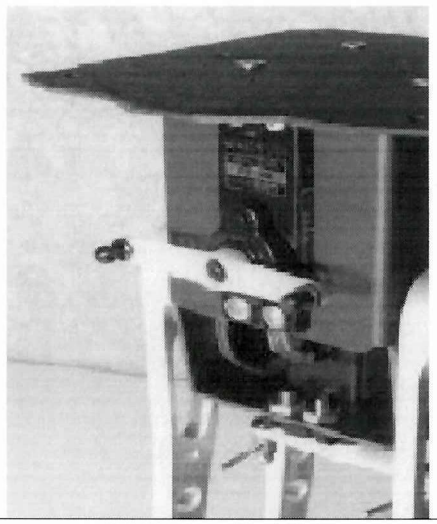


FIGURA No 3.18:
B) INSTALACIONES DE LAS JUNTURAS.
FUENTE: INVESTIGADOR

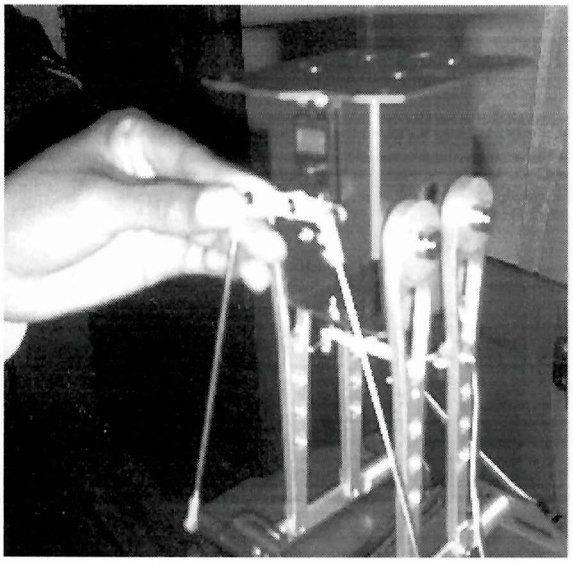


FIGURA NO 3.19: COLOCACIÓN DE VARAS.
FUENTE: INVESTIGADOR

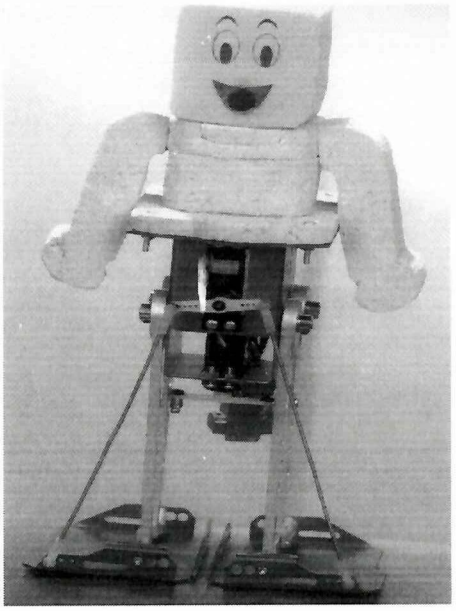
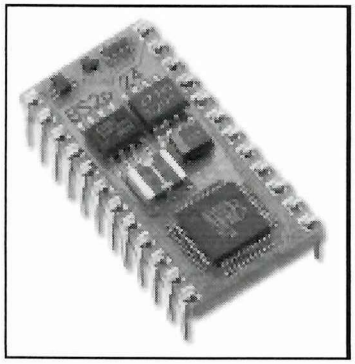


FIGURA No 3.20:
INSTALACIÓN DEL CUERPO ELEMENTAL.
FUENTE: INVESTIGADOR



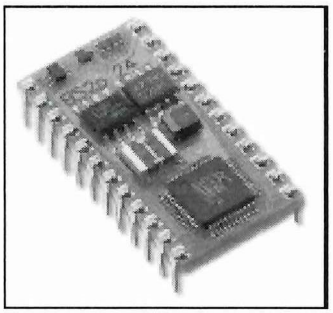
Manual de Programación

BASIC Stamp 2.2.

1. INTRODUCCIÓN AL BASIC STAMP 2

1.1. HISTORIA DEL BASIC STAMP

Es un microcontrolador desarrollado en 1992 por la empresa norteamericana Parallax Inc. Esta conformada por un conjunto de circuitos ensamblados en una misma placa impresa la misma que contiene: un microcontrolador interprete, una memoria EEPROM para almacenamiento de programas, un regulador de 5V, un resonador y un conjunto de terminales de E/S de uso general con niveles de tensión compatibles TTL/CMOS(0-5 V).



El Basic Stamp se presenta en dos modelos, el Basic Stamp 1 y 2, El Basic Stamp se programa utilizando un grupo de directivas, comandos y funciones para la realización de operaciones tanto matemáticas, lógicas y de entrada – salida que conforma un poderoso lenguaje llamado PBASIC.

1.2 EL MICROCONTROLADOR BASIC STAMP 2 (BS2)

El BASIC Stamp II es un pequeño computador que ejecuta programas en lenguaje PBASIC. El BS2-IC tiene 16 pines de (entrada / salida) I/O que pueden ser conectados directamente a dispositivos digitales o de niveles lógicos, tales como botones, diodos leds, altavoces, potenciómetros, y registros de desplazamiento. Además, con unos pocos componentes extras,

estos pines de I/O pueden ser conectados a dispositivos tales como solenoides, relay, servomotores, motores de paso a paso, y otros dispositivos de alta corriente o tensión.

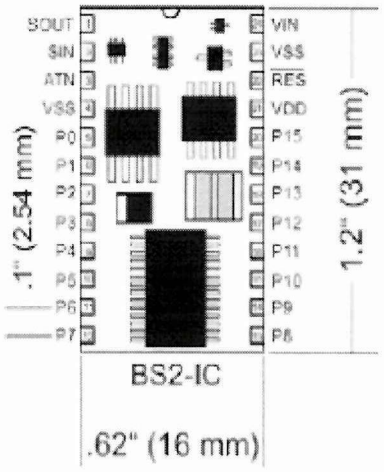


Figura 1: Diagrama esquematico BS2

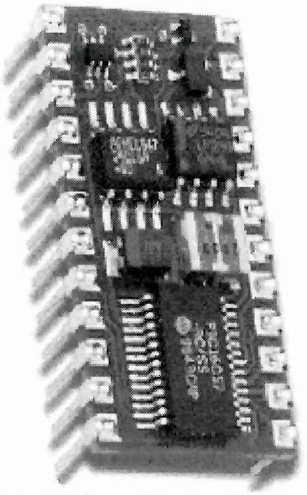


Figura 1.1 : Basic Stamp 2

EQUIPOS NECESARIOS PARA TRABAJAR CON EL BS2

- Un microcontrolador Basic Stamp 2, Ref. #BS2-IC
- Un cable serial RS-232 (no null modem)
- Un Pulsador Momentáneo (N.O.)
- Fuente de alimentación (+5 V - +15 V)
- Una computadora personal PC; S.O. Windows 95/98/NT4/2000
- Programa Editor PBASIC
- Una tablilla de experimentación BreadBoard entre otros componentes.

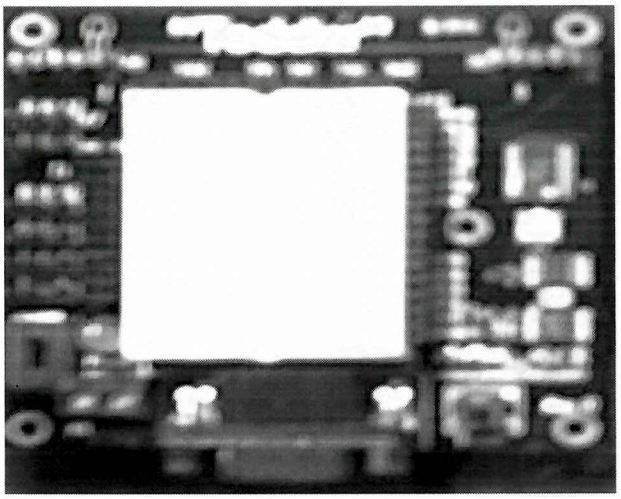


Fig. 1.3 Placa con equipos para utilizar con el Bs2.

1.3 FUNCIONAMIENTO INTERNO DEL BS2

El diseño físico consiste en un regulador de 5+ voltios, un oscilador de 20 MHz, una memoria EEPROM de 2K, un detector de bajo voltaje e chip intérprete PBASIC. Un programa compilado en PBASIC es almacenado en la EEPROM, desde donde el chip intérprete grabado en el microcontrolador lee y escribe las instrucciones.

Este chip intérprete ejecuta una instrucción cada vez, realizando la operación apropiada en los pines de I/O o en la estructura interna del chip intérprete. Debido a que el programa PBASIC es almacenado en una EEPROM, puede ser reprogramada una cantidad cercana a 10 millones de veces.

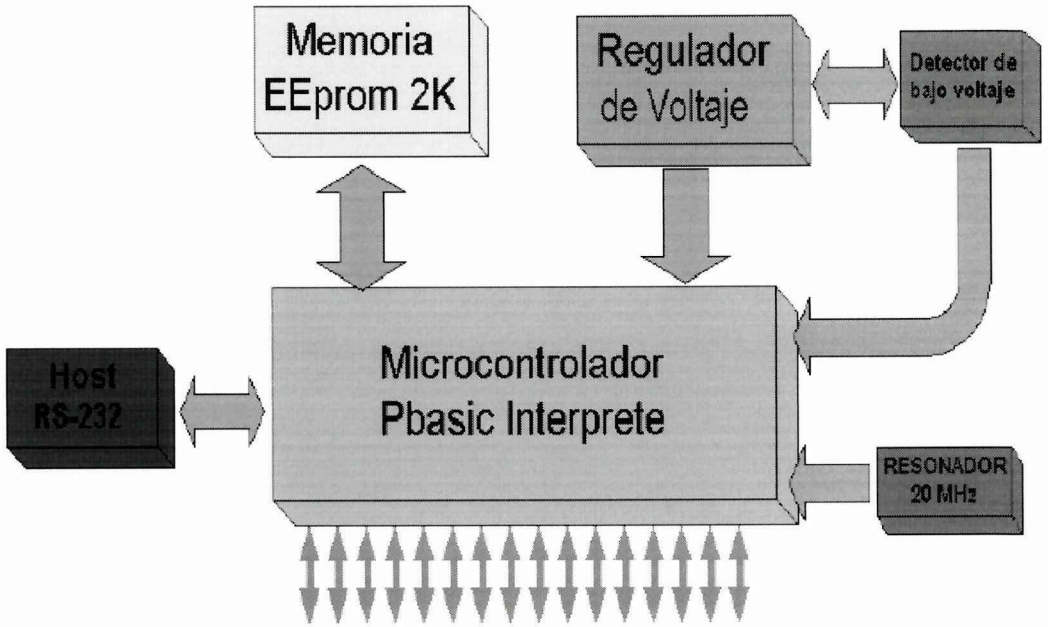


Figura 1.4: Diagrama en Bloque del BS2

El Basic Stamp II es capaz de almacenar entre 500 y 600 instrucciones de alto nivel (PBASIC) y ejecuta un promedio de 4000 instrucciones /segundo. Para programar el BS2-IC, simplemente conéctele un cable serial preparado entre el BS2 y un PC, y ejecute el software editor para crear y descargar su programa, a través del cable serial.

FUNCIONAMIENTO INTERNO DEL BS2.

Hardware del BS2

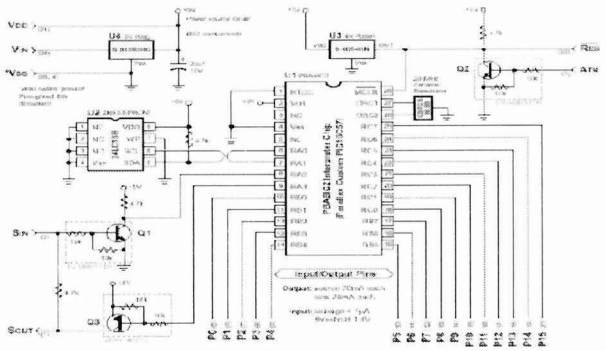


Figura 1.5: Diagrama eléctrico del Basic Stamp 2

El chip intérprete del Basic Stamp II (U1)

El cerebro del BS2 lo constituye un microcontrolador PIC16C57, de la familia de Microchip. U1 esta programado permanentemente de fábrica con un conjunto de instrucciones predefinidas del lenguaje PBASIC. Cuando se programa el BS2, usted le está diciendo a U1 que salve las instrucciones compiladas, llamadas fichas de instrucciones hexadecimales, en la memoria EEPROM (U2). Cuando su programa se ejecuta, U1 extrae las fichas de instrucciones hexadecimales de la memoria (U2), los interpreta como instrucciones PBASIC, y ejecuta las instrucciones equivalentes.

U1 ejecuta su programa interno a una velocidad de 5 millones de instrucciones por segundo. Algunas instrucciones internas entran en una sola instrucción PBASIC2, así que PBASIC2 ejecuta más lentamente aproximadamente 3000 a 4000 instrucciones por segundo. El PIC16C57 tiene 20 pines en total, 16 están destinados a entrada /salida (I/O); 4 están destinados a la comunicación serial RS-232, en el circuito BS2 16 contactos están disponibles para uso general por sus programas, dos de las otros se pueden también utilizar para la comunicación serial asincrónica, los dos restantes se utilizan solamente para interconectar con el EEPROM y no se pueden utilizar.

Los contactos de uso general de I/O, (P0-P15), se pueden interconectar con toda la lógica de +5 voltios moderna, de TTL (lógica del transistor-transistor) con CMOS (semiconductor de óxido metálico).

DESCRIPCIÒN DE LOS PINES DEL BS2

Pin	Nombre	Descripción
1	SOUT	Serial Out: Conectar al puerto serial RX (DB9 pin 2)
2	SIN	Serial In: Conectar al puerto serial TX (DB9 pin 3)
3	ATN	Atención: Conectar al puerto serial DTR (DB9 pin 4)
4	GND	Tierra entre el puerto serial y el BS2
5-20	P0-P15	Puerto de propósitos generales, cada uno puede entregar 25 mA, sin embargo, el total de la corriente no puede exceder los 75 mA utilizando el regulador interno y 100 mA utilizando +5V externo
21	VDD	Voltaje regulado a +5 VDC
22	RES	Reset, Basta con aterrizar y el BS2 reinicialaza
23	GND	Tierra del BS2
24	PWR	Voltaje no regulado entre +5.5 a +15 VDC, si VDD es utilizado VIN no puede ser utilizado

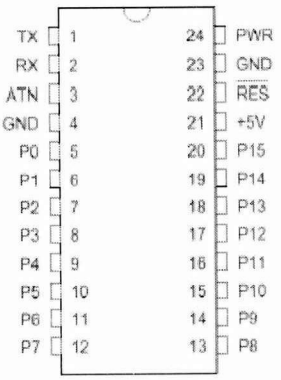


Figura 1.6 : Ubicación de cada PIN

CONEXIÒN TÌPICA PARA SU FUNCIONAMIENTO

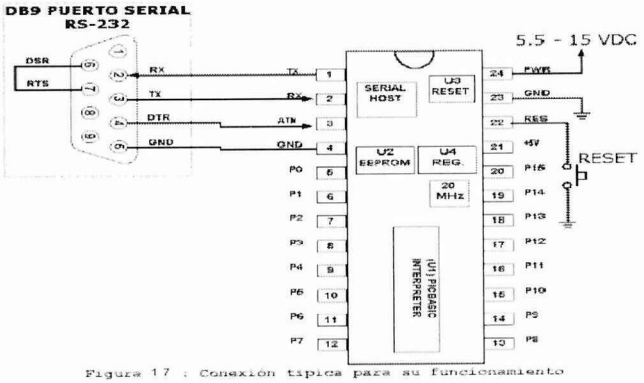


Figura 1.7 : Conexión tipica para su funcionamiento

2. LENGUAJE DE PROGRAMACIÓN PBASIC

PBASIC es un lenguaje de programación basado en un BASIC estructurado orientado a entrada y salida de señales. La utilización de sencillas instrucciones de alto nivel, permite programar los Basic Stamps para controlar cualquier aplicación llevada a cabo por un microcontrolador, las instrucciones de PBASIC permiten controlar las líneas de (entrada /salida),

2.1 PBASIC Editor

El PBASIC editor es el programa donde escribimos el conjunto de instrucciones para el Basic Stamp, es similar en apariencia a cualquier editor de texto del sistema operativo WINDOWS. El editor contiene una serie de herramientas como son identificadores del Basic Stamp, corrector ortográfico de sintaxis, Mapa de memoria y Ventana del depurador.

El editor tiene la capacidad para abrir 16 ventanas simultáneamente. La capacidad de cortar, copiar y pegar se mantiene innata. Su entorno es muy sencillo y usted se familiarizara muy pronto.

Los comandos más importantes son:

F1: Muestra la ayuda en pantalla

Ctrl-O: Abre un archivo

Ctrl-S: Salva un archivo

Ctrl-P: Imprime el archivo actual

F9 o Ctrl-R: Descarga el programa en el BS2

F7 o Ctrl-T: Corrector de Sintaxis

F8 o Ctrl-M: Muestra el mapa de memoria

F6 o Ctrl-I: Muestra él numero de versión de PBASIC

ESC: Cierra la ventana actual.

GRAFICO DEL EDITOR PBASIC

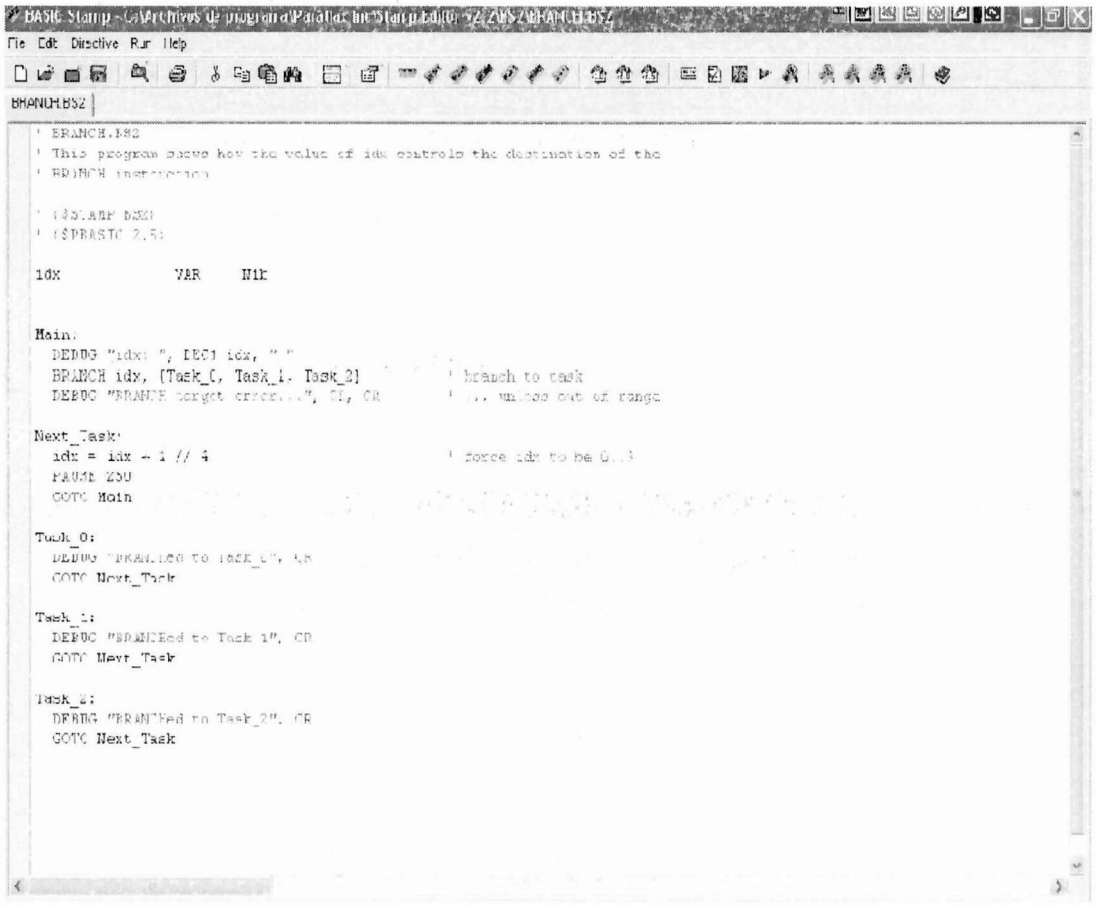


Figura 2: Pantalla de Editor de Pbasic

2.2 PROCEDIMIENTO PARA DESCARGAR EL PROGRAMA AL BS2:

1. Con el BS2 previamente energizado y conectado como lo indica la figura 3.4, cargue el editor PBASIC.

2. Cuando el editor este listo presione [Ctrl-I], si todo esta bien conectado el editor le dará un mensaje de "Found BS2-IC (firmwarev1.0)". Esto indica que va por buen camino.

3. Usted puede digitar su programa o cargar uno previamente del disco.

4. Para asegurar que el código digitado este bien, presione el corrector de sintaxis [Ctrl-T], si existe algún problema lo indicara con un mensaje de error. Si todo marcha bien le indicara un mensaje de "Tokenize Successful"

5. Ahora estará listo para descargar el programa en el BS2, presione [Ctrl-R], y el programa se descargara permanentemente en la EEprom del BS2. En caso de que usted no revise con el corrector de sintaxis, antes del programa descargarse en el BS2, este lo realiza por su cuenta.

6. Apague el Basic Stamp 2, Retire el cable serial del BS2.

7. Encienda el Basic Stamp 2 y la aplicación permanecerán hasta que

2.3 ESTILO DE PROGRAMACIÓN

En la programación cada cual adopta un estilo único, dados los conocimientos básicos de cada instrucción, se plantea el problema el cual debemos solucionar con las funciones disponibles. La combinación de funciones es realmente ilimitada. El mismo problema se puede solucionar de diversas formas, cada cual lo hará con su criterio lógico aprendido.

Código Modelo

Aunque cada quien adopta un estilo propio, es conveniente seguir un patrón de ordenamiento, el siguiente ejemplo nos muestra un código:

Modelo:

```
'----- [Titulo] ----------------------------------------------------------
'Archivo..... Write_01.bs2
'Propósito..... Aprender el uso del comando WRITE
'Autor..... Diego Pulgar
'E-mail..... dpulgar@comser.com.do
'WEB..... www.dpgelectric.com
'Versión..... 1.2
'Fecha de Inicio... 12/ENE/2002
'Fecha de Final... 02/JUN/2002
'Cliente..... Caribean Tecnologies
'----- [Descripción del Programa] ---------------------------------
'----- [I/O Definiciones] ------------------------------------------
'----- [Constantes] ------------------------------------------------
'----- [Variables] -------------------------------------------------
'----- [EEPROM Data] -----------------------------------------------
'----- [Inicializaciones] ------------------------------------------
'----- [Main Code] -------------------------------------------------
'----- [Sub-rutinas] -----------------------------------------------
```

La idea de este estilo es documentar todo el contenido y distinguir todos los procedimientos.

Por lo general en un código PBASIC se debe llevar el siguiente orden:

1. Definir las constantes
2. Definir las variables de Entrada y Salida
3. Definir las variables de Programa
4. inicializar el puerto
5. Direccionar las entradas y salidas
6. Iniciar los circuitos periféricos, si existen
7. Rutina principal (Main)
8. Rutinas Secundarias
9. Sub-Rutinas (Rutinas que se repiten).

3. MEMORIA ORGANIZADA Y VARIABLES.

El Basic Stamp tiene dos tipos de memoria:

- RAM.
- EEPROM.

3.1 BASIC STAMP 1 RAM ORGANIZADA

El BS1 tiene 16 bytes (8 words) de RAM. A continuación se detalla las especificaciones de la RAM:

PALABRAS	BYTE	BIT	NOTAS ESPECIALES
PORT	PINS	PIN0-PIN7	I/O pins, Bit direcciones.
	DIRS	DIR0-DIR7	I/O pins dirección; Bit dirección.
WO	BO	BIT0-BIT7	Bit de direcciones.
	B1	BIT8-BIT15	
W1	B2		
	B3		
W2	B4		
	B5		
W3	B6		
	B7		
W4	B8		
	B9		
W5	B10		
	B11		
W6	B12		Usado por la instrucción Gosub.
	B13		

PORT: Se usa para el control del Pin para I/O.

PINS: Consisten de 8 Bits (PIN0- PIN7) que corresponde a cada ocho I/O de los pines sobre el BS1.

La lectura de los Bytes: (PINS) lee eficazmente la I/O directa de los pines. Devuelve 8 bits colocando 1s o 0s corresponde al estado de alto y bajo. De la respectiva I/O del pin en ese momento.

La escritura de los Bytes :(PINS) Guardan un valor alto o bajo de I/O respectiva en ese momento (Aunque solo en los pines que se esta enviando la salida).

DIRS: Controla la dirección de la I/O de los pines. Consiste de 8 bits (DIR0 – DIR7).Un bit se encuentra en estado alto (1) ya sea una I/O de pines corresponde a una salida de dirección. Un bit se encuentra en estado bajo (0) ya sea una I/O de pines corresponde a una entrada de dirección.

Las words restantes (W0–W6) están disponibles para el uso general. Cada words consiste en dos bytes de direcciones (B0 y B1) incluyendo las direcciones de los bits. Se puede asignar otros nombres.

3.2 BASIC STAMP 2 RAM ORGANIZADA

Los BS2, BS2e, BS2sx y BS2p tienen 32 bytes (16 words) de RAM.

Los primeros seis bytes son reservados para la entrada, salida y control de las I/O de los pins. Permaneciendo 26 bytes disponibles para el uso general como las variables. Hay 16 words, que consisten en dos bytes para cada uno siendo un total de 32 bytes. Todos bits son direcciones individuales a través de los modificadores y los bits dentro de las tres words son también direcciones individuales los nombres son predefinidos como se muestra a continuación:

PALABRAS	BYTE	BIT	NOTAS ESPECIALES
INS	INL	IN0-IN7	Input pins
	INH	IN8-IN15	
OUTS	OUTL	OUT0-OUT7	Output pins
	OUTH	OUT8-OUT15	
DIRS	DIRL	DIR0-DIR7	I/O control de dirección de los pines
	DIRH	DIR8-DIR15	
W0	B0		
	B1		
W1	B2		
	B3		
W2	B4		
	B5		
W3	B6		
	B7		
W4	B8		
	B9		
W5	B10		
	B11		
W6	B12		
	B13		
W7	B14		
	B15		
W8	B16		

	B17		
W9	B18		
	B19		
W10	B20		
	B21		
W11	B22		
	B23		
W12	B24		
	B25		

INS: Siempre muestra el estado de I/O de los pines de ellos mismos, sin tener en cuenta la dirección de cada I/O de los pines (Lectura de Pines). Siempre empareja los estados actuales de las I/O de los pines, si ellos se encuentran en I/O.

INL: Byte Bajo; **INH**: Byte Alto.

DIRS: Determina el estado de un pin si es colocado por el circuito externo (input, 0) o por el estdo de los OUTS (output, 1).

OUTS: Tiene bits que solo aparecen en pines cuando los DIRS de los bits son colocados en Salidas.

La memoria del Basic Stamp es organizada en 16 words de 16 bits cada uno. Las tres primeras words se usan para I/O. Las13 palabras están disponibles para el uso como las variables de uso general. Estas son predefinidas los nombres: W0 a través de W12 y B0 a

través de B25. Donde B0 es el byte bajo de W0; B1 es el byte alto de W0; y así sucesivamente a través de W12 (El byte de B24= bajo, byte de B25= alto).

3.3 DEFINIENDO Y USANDO LAS VARIABLES

Reglas

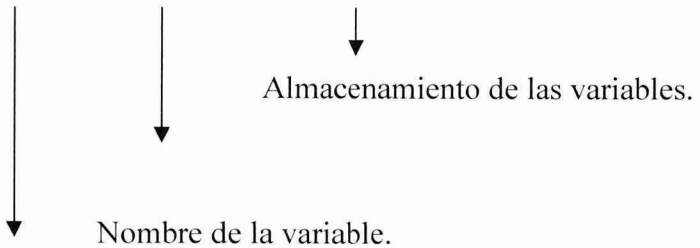
- Puede contener una mezcla de letras, números y caracteres.
- No debe estar igual a las palabras claves de Basic Stamp.
- El tamaño es de 32 caracteres de largo.
- El Basic no distingue entre minúsculas y mayúsculas.

La Sintaxis para las declaraciones de las variables inconstantes son:

BS1

Estas declaraciones inconstantes apuntan a una situación específica en el RAM.

SYMBOL Name = Register Name



Tipo de la variable.

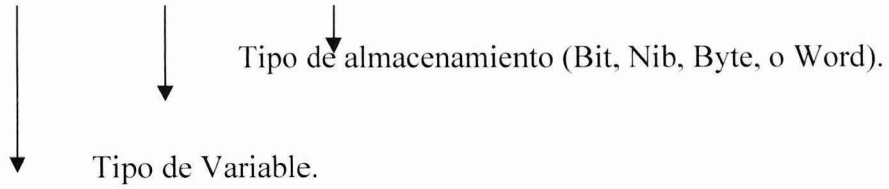
EJEMPLOS:

```
SYMBOL  derecha   = W0          ' valor puede ser de 0 a 65535
SYMBOL  centrado  = B1          ' valor puede ser de 0 a 255
SYMBOL  resultado = B2          ' valor puede ser de 0 a 255
```

BS2, BS2e, BS2sx, BS2p

Se especificamos el tamaño deseado para cada inconstante; el Basic Stamp los colocará en el RAM. Se puede recuperar los volúmenes de estas variables.

Name VAR Var Type



Nombre de la Variable.

EJEMPLOS:

En la familia de BS2 tienen 4 opciones: 1) Bit (1 bit), 2) Nib (nibble; 4 bits), 3) Byte (8 bits), y 4) Word (16 bits).

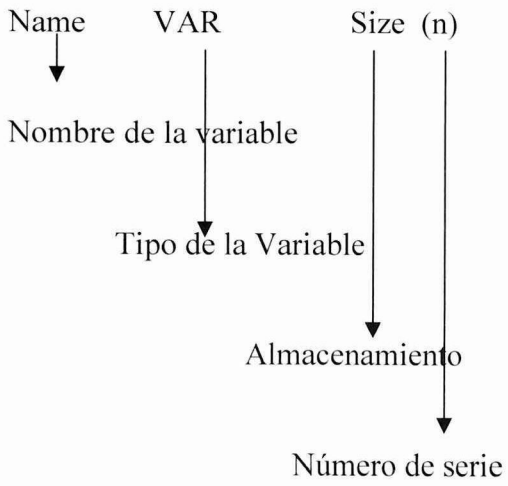
```
ratòn       VAR    Bit          ' El valor puede ser 0 o 1
gato        VAR    Nib          ' El valor puede ser 0 a 15
perro       VAR    Byte         ' El valor puede ser 0 a 255
rinoseronte VAR    Word         ' El valor puede ser 0 a 65535
```

Si usted asigna un valor a una variable que excede su tamaño, los bits del exceso se perderán.

3.3.1 VARIABLE MULTIPART (SERIES)

Es un grupo de variables del mismo tamaño, y compartiendo un solo nombre, se puede definir el número de variables considerado un array.

Sintaxis para BS2:



EJEMPLOS:

Utilizando array:

```
gato  VAR     Byte (10)          ' Crea 10-byte array
```

Una vez una serie se define, usted puede acceder sus elementos por el número. Las salidas numerando a 0 y extremos al n–1. Por ejemplo:

```
gato (3) = 57
DEBUG? gato (3)
```

EJEMPLO:

Utilizando el For:

```
gato    VAR    Byte (10)        ' Define 10-byte array
index   VAR    Nib              ' Define normal nibble variable
Main:
FOR index = 0 TO 9              ' Repite con index= 0, 1, 2...9
   gato (index) = index * 13     ' Escribe el index*13 cada celda del array.
   DEBUG? gato (index)           ' Muestra el contenido de cada array.
NEXT
STOP
```

3.3.2 OTRAS FORMAS PARA DECLARAR LAS VARIABLES

Esta forma se la utiliza mediante el alias donde en la primera declaración el nombre de la variable se le utiliza para el tipo del tamaño de la segunda declaración de la variable.

Ejemplos:

SB1

```
SYMBOL gato    = B0              ' Crea el tamaño de byte de la variable.
SYMBOL tabla   = gato            ' Crea el alias para la variable gato.
```

BS2

```
gato   VAR    Byte               ' Crea el tamaño de Byte de la variable.
tabla  VAR    gato               ' Crea el alias para la variable gato.
```

3.3.3 MODIFICADORES CON VARIABLES

En la familia BS2 el alias puede servir también como una ventana en una porción de otro inconstante. Se usa "modificadores". Aquí el alias se asigna como un modificador que especifica la parte.

```
oso         VAR     Word            ' La variable de 16 bits.
cabeza      VAR     oso.HIGHBYTE    ' Ocho bits altos de oso.
cuerpo      VAR     oso.LOWBYTE     ' Ocho bits bajos de oso.
```

Lista De Modificadores Inconstantes:

Symbol	Definitions
LOWBYTE	Low byte of a word
HIGHBYTE	High byte of a word
BYTE0	Low byte of a word
BYTE1	High byte of a word
LOWNIB	Low nibble of a word or byte
HIGHNIB	High nibble of a word or byte
NIB0	Nibble 0 of a word or byte
NIB1	Nibble 1 of a word or byte
NIB2	Nibble 2 of a word
NIB3	Nibble 3 of a word
LOWBIT	Low bit (LSB) of a word, byte, or nibble
HIGHBIT	High bit (MSB) of a word, byte, or nibble
BIT0	Bit 0 (LSB) of a word, byte, or nibble
BIT1	Bit 1 of a word, byte, or nibble
BIT2	Bit 2 of a word, byte, or nibble
BIT3	Bit 3 of a word, byte, or nibble
BIT4 ... BIT7	Bits 4 through 7 of a word or byte
BIT8 ... BIT15	Bits 8 through 15 of a word

EJEMPLOS:

```
oso     VAR    Word            ' Contiene 16 bits en la variable.
ojos    VAR    oso.BIT9        ' Un bit
```

1) Usted también puede usarlos dentro de las instrucciones del programa:

```
oso     VAR Word                ' A 16-bit variable
ojos    VAR oso.HIGHBYTE        ' Ocho bits altos de la variable oso.
Main:
 oso = 13567
 DEBUG? ojos                    ' Muestra el valor del alias de la variable ojos.
 DEBUG? oso.HIGHBYTE
 STOP
```

2) Utilizando Arrays:

```
lista VAR    Byte (10)              ' Define 10 bytes de array
Main:
 lista (0) = $AB                    'Hex $AB into 0th byte
 DEBUG HEX? lista.LOWNIB (0)        ' Muestra el nibble bajo ($B)
DEBUG HEX? lista.LOWNIB (1)         ' Muestra el nibble alto  ($A)
```

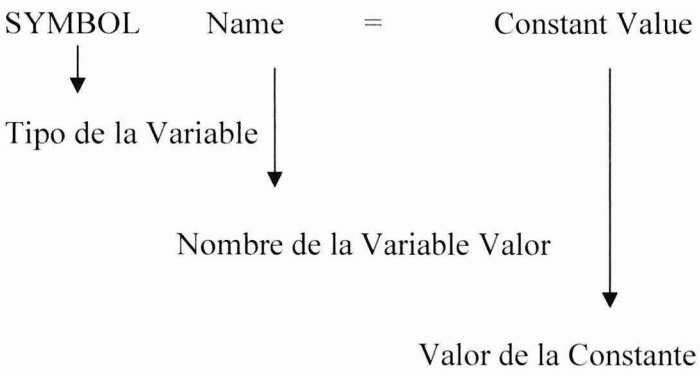
4. ESTRUCTURA EN PROGRAMACIÒN DE BS2

4. 1 CONSTANTES

Son aquellas que cuando el programa está corriendo, nada puede pasar para cambiar esos números. Esto distingue las constantes de variables que pueden cambiar mientras el programa está corriendo. Las constantes pueden ser del sistema decimal, binario, hexadecimal y código ASCII.

Sintaxis:

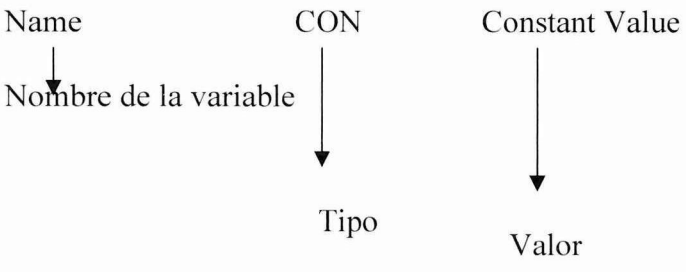
BS1)



EJEMPLO:

SYMBOL ejemplo = 3

BS2)



EJEMPLO:

Gato CON 15

Las constantes pueden ser declaradas y ubicadas en arrays, for y operaciones.

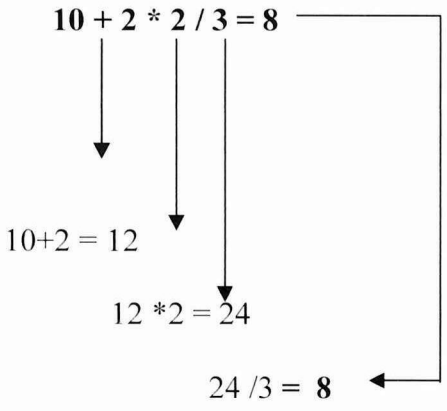
```
Perros   CON    5                ' Número de perros.

Gatos    CON    Perros * 2 - 1   ' Número de gatos.
```

4.2 OPERADORES

Reglas:

1) En BS1 las operaciones son resueltas en el orden que son escritas de izquierda a derecha no son escritas entre paréntesis.

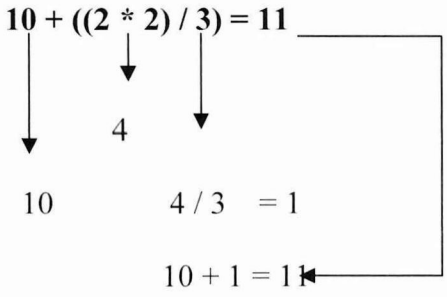
Ejemplos:



2) En una división los resultados siempre son enteros.

Resultado Correcto 500/3 = 166 **Resultado Incorrecta** 500/3 =166,6

3) En BS2 las operaciones son escritas dentro de los paréntesis teniendo en cuenta el paréntesis mas interno hacia fuera.



4.2.1 OPERACIONES BINARIAS

4.2.1.1 LA ADICCIÒN

Sintaxis: *Valor1 + Valor2*

Suma variables o constantes, retornando 16 bits de resultados. Las operaciones solo admiten enteros desde 0 hasta 65535. Si el resultado de la adicción es larga se perderá el bit que se lleva.

Ejemplo:

BS1)

```
SYMBOL  valor1 = W0
SYMBOL  valor2 = W1
Main:
  valor1 = -99
  valor2 = 100
```

```
valor1 = valor1 + valor2          ' Suma los números.
DEBUG valor1                      ' Muestra el resultado 1.
END
```

BS2)

```
Valor1 VAR    Word
Valor2 VAR    Word
Main:
 Valor1 = -1575
 Valor2 = 976
 Valor1 = Valor1 + Valor2          ' Suma los números
 DEBUG SDEC ?  Valor1              'Muestra el resultado (-599)
 END
```

4.2.1.2 LA SUSTRACCIÒN

Sintaxis: *Valor1 - Valor 2*

Resta las variables o constantes, retornando 16 bits de resultados. Las operaciones solo admiten enteros desde 0 hasta 65535. Si el resultado es negativo, se expresará correctamente como un número de 16 bits.

EJEMPLO:

BS1)

```
SYMBOL  valor1  = W0
SYMBOL  valor2  = W1
Main:
```

```
valor1 = 288
valor2 = 200
valor1 = valor1 - valor2        ' Resta el valor2 del valor1
DEBUG valor1                    ' Muestra el resultado (88)
END
```

BS2)

```
valor1  VAR    Word
valor2  VAR    Word
Main:
  valor1 = 7000
  valor2 = 9999
  valor1 = valor1 - valor2              ' Resta el valor2 del valor1
  DEBUG SDEC ? valor1                   ' Muestra el resultado (-2999)
  END
```

4.2.1.3 MULTIPLICACIÒN

Sintaxis: Valor * Valor

Multiplica variables o constante, devuelve 16 bits bajos del resultado. Las operaciones solo admiten enteros desde 0 hasta 65535.

Si el resultado de multiplicación es más grande que 65535, el exceso de los bits se perderán.

La multiplicación de variables será correcta en los dos el número y signo, el resultado está en el rango -32767 a +32767

EJEMPLO:

BS1)

```
SYMBOL  valor1  = W0
SYMBOL  valor2  = W1
Main:
valor1 = 100
valor2 = 20
valor1 = valor1 * valor2              ' Multiplicación de Números.
DEBUG valor1                          ' Muestra el Resultado (2000)
END
```

BS2)

```
valor1  VAR     Word
valor2  VAR     Word
Main:
 valor1 = 100
 valor2 = 20
 valor1 = valor1 * valor2              ' Multiplicación de los Números.
 DEBUG SDEC ? valor1                   ' Muestra los resultados (2000)
 END
```

4.2.1.4 DESPLAZAMIENTO DE CIFRAS (<< Y >>)

Los operadores << y >> desplazan un valor hacia la izquierda ó derecha respectivamente. (n) numero de lugares en la cifra o valor, Los bits desplazados se colocan en 0.

El comando (<<) desplaza la cifra hacia la izquierda (n) numero de lugares el resto es llenado por ceros. Este comando es equivalente a aplicar una multiplicación de 2n donde n es el número de lugar a desplazar.

POR EJEMPLO:

valor CON %100101101111

W1 = valor << 2

DEBUG BIN16 W1, CR 'Muestra el resultado (100101101100)

Corre la Cifra 2 Lugares hacia la izquierda rellena hacia la derecha dos lugares con ceros.

W1 = valor << 4

DEBUG BIN16 W1, CR 'Muestra el resultado (1001011011110000)

Corre la Cifra 4 Lugares hacia la izquierda rellena hacia la derecha cuatro lugares con ceros. Nótese que el resultado de izquierda a derecha se pierde. En otras palabras cuando decimos: 234 << 3. Es equivalente a 234 * 2 3 = 234 * 8.

El comando (>>) desplaza la cifra hacia la derecha (n) numero de lugares el resto es llenado por ceros. Este comando es equivalente a aplicar una división de 2n donde n es el número de lugar a desplazar.

POR EJEMPLO:

Valor CON %100101101111

W1 = valor >> 2

DEBUG BIN16 W1, CR 'Muestra el resultado (00100101101111)

Corre la Cifra 2 Lugares hacia la derecha rellena hacia la izquierda dos lugares con ceros.

W1 = valor >> 4

DEBUG BIN16 W1, CR 'Muestra el resultado (0000100101101111)

Corre la Cifra 4 Lugares hacia la derecha rellena hacia la izquierda cuatro lugares con ceros. Nótese que el resultado de derecha a izquierda se pierde. En otras palabras cuando decimos: 234 >> 3. Es equivalente a 234 / 2 3 = 234 / 8.

4.2.1.5 DIVISIÒN

El operador de la División (/) divide variables y/o constantes, retorna 16 bits de resultado. Use / sólo con los valores positivos; en números negativos el operador / no proporción los resultados deseados.

EJEMPLOS:

BS1)

```
SYMBOL  valor1  = W0
SYMBOL  valor2  = W1
Main:
  valor1 = 200
  valor2 = 2
  valor1 = valor1 / valor2          ' Divide los números.
  DEBUG valor1                      ' Muestra el resultado (100)
  END
```

BS2)

```
valor1 VAR    Word
valor2 VAR    Word
Main:
  valor1 = 200
```

```
valor2 = 2
valor1 = valor1 / valor2                ' Divide los números.
DEBUG DEC? value1                       ' Muestra los resultados (100)
END
```

En el caso de una división negativa se determina en una variable el signo y el dividendo y el divisor se les realiza un valor absoluto, donde se determinara mediante una condición. Todos los valores deben quedar dentro del rango de -32767 a +32767.

EJEMPLO:

BS2)

```
Signo   VAR     Bit             ' Tiene el BIT del Signo.
Valor1 VAR      Word
Valor2  VAR     Word
Main:
  valor1 = 20
  value2 = -200
  signo = valor1.Bit15 ^ valor2.Bit15       ' Determina el resultado del signo.
  valor2 = ABS valor2 / ABS valor1          ' Divide el valor absoluto.
  IF (signo = 1) THEN valor2 = -valor2      'Corrige el signo si es negativo.
  DEBUG SDEC ? valor2                       ' Muestra el resultado(-10)
```

4.2.1.6 MODULO

Sintaxis: Valor1 / / Valor2

El operador del Módulo (/ /) nos indica el residuo de una división.

EJEMPLO

BS1)

```
SYMBOL  valor1  = W0
SYMBOL  valor2  = W1

Main:

  valor1 = 1000

  valor2 = 6

  valor1 = valor1 // valor2              ' Determina el residuo.

  DEBUG valor1                           ' Muestra el residuo (4)

  END
```

BS2)

```
valor1  VAR     Word

valor2  VAR     Word

Main:

  valor1 = 1000

  valor2 = 6

  valor1 = valor1 // valor2              ' Determina el residuo.

  DEBUG DEC ? valor1                     ' Muestra el residuo (4)

  END
```

4.2.1.7 EL MAXIMO

Límites de un Valor especifica el alto.

El operador Máximo (MAX) limita un valor específico de 16-Bit positivos máximos.

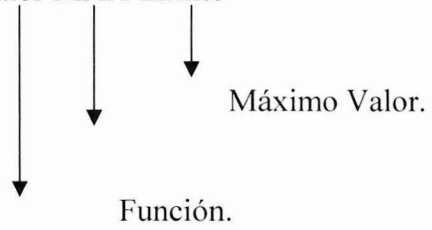
LOGICA

Valor > Limite ⟶ Resultado = Limite.

Valor≤ Limite ⟶ Resultado = Valor.

Sintaxis:

Valor MAX Límite



Máximo Valor.

Función.

Constante o Variable.

Ejemplos:

BS1)

```
SYMBOL  valor1  = W0
SYMBOL  valor2  = W1
Main:
 FOR valor1 = 0 TO 50 STEP 10        ' Camina el Valor 1 de 0 a 50
  valor2 = valor1  MAX  10           ' Use MAX con un limite de  10
  DEBUG  valor2                      ' Muestra el valor máximo
```

```
NEXT
END
```

BS2)

```
valor  VAR    Word
Main:
 FOR valor = 0 TO 50 STEP 10          ' Camina el valor del 0 al 50.
  DEBUG ? valor MAX 10                ' Muestra el valor máximo.
 NEXT
 END
```

4.2.1.8 EL MINIMO

Límites de un Valor especifica el bajo.

El operador Mínimo (MIN) limita un valor específico de 16-Bit positivos mínimos.

LOGICA

Valor < Limite ⟶ Resultado = Limite.

Valor ≥ Limite ⟶ Resultado = Valor.

Sintaxis:

Valor MIN Límite

Máximo Valor.

Función.

Constante o Variable.

EJEMPLOS:

```
value1  VAR     Word
value2  VAR     Word
Main:
 FOR value1 = 100 TO 0 STEP 10        ' Camina el valor 1 del 100 al 0.
  value2 = value1 MIN 50               ' Usa el mínimo
  DEBUG DEC ? value2 MIN 50                ' show "clamped" value
 NEXT
 END
```

4.2.1.9 DIG

Este operador retorna los dígitos decimales específicos de un valor de 16 BIT positivos. Los dígitos son números de 0 a 4.

El cero muestra los dígitos más a la derecha y el 4 a los dígitos mas a la izquierda de 16 BIT es decir en decimal del 0 al 65535.

Sintaxis:

VALOR DIG VALOR

Posición del Digito a extraer

Función

Cantidad

EJEMPLO:

```
value   VAR     Word
idx     VAR     Nib
Main:
```

```
value = 1982
DEBUG ? value DIG 2                    ' Show digit 2 (9)
FOR idx = 0 TO 4
DEBUG ? value DIG idx                  ' Show digits 0 - 4 of 1982
NEXT
END
```

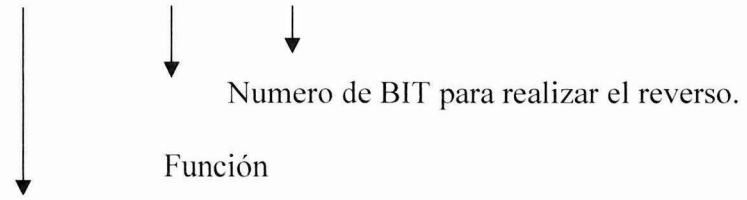
4.2.1.10 REV

El operador inverso retorna una reversa copia de específicos números de BIT de un valor.

Inicia con el BIT de la derecha.

Sintaxis:

BIT REV VALOR




```
DEBUG BIN4? %11001010 REV 4            ' Mirror lower 4 bits (%0101)
```

4.2.1.11 AND

El operador AND (&), devuelve un BIT lógico de dos valores. Cada BIT de los valores está sujeto a la lógica siguiente:

0 & 0 = 0

0 & 1 = 0

1 & 0 = 0

1 & 1 = 1

El resultado que retorne 1s será solo aquellos bits que contengan en ambas entradas 1s.

EJEMPLO:

BS1:

```
SYMBOL  valor1     = B0
SYMBOL  valor2     = B1
SYMBOL  resultado  = B2
Main:
  valor1 = %11111111
  valor2 = %10101010
  resultado = valor1 & valor2
  DEBUG %resultado        ' Muestra el resultado 10101010
  END
```

BS2:

```
DEBUG BIN8? %11111111 & %10101101  ' Muestra el resultado (%10101101)
```

4.2.1.12 OR

El operador OR (I), devuelve un BIT lógico de dos valores. Cada BIT de los valores está sujeto a la lógica siguiente:

0 | 0 = 0

0 | 1 = 1

1 | 0 = 1

1 | 1 = 1

El resultado retornado por OR contendrá 1s, si una entrada o ambas contienen 1s.

EJEMPLOS:

BS1:

```
SYMBOL  valor1  = B0
SYMBOL  valor2  = B1
SYMBOL  result = B2
Main:
  valor1 = %11111111
  valor2 = %10101001
  result = valor1 | valor2
  DEBUG  %result        ' Muestra el resultado %11111111
  END
```

BS2:

```
  DEBUG BIN8? %00001111 | %11111111    ' Muestra el resultado %11111111
```

4.2.1.13 AND LOGICO

El operador AND retorna un valor lógico de dos valores o expresiones. En Basic Stamp el valor cero es False y el valor uno es True. Los valores o las expresiones están sujetos a la lógica siguiente:

F AND F = F

F AND T = F

T AND F = F

T AND T = T

El resultado vuelto por AND será Verdad o Falso.

EJEMPLOS:

BS2:

```
A    VAR    Byte
B    VAR    Byte
Main:
 A = 93
 B = 89
 IF (A > 80) AND (B > 70) THEN
 DEBUG "Se encuentra en el promedio "   ' Cuando la condición es Verdadera. ELSE
DEBUG "No se encuentra en el promedio "    'Cuando la condición es Falsa.
 ENDIF
 END
```

4.2.1.14 OR LOGICO

El operador OR retorna un valor lógico de dos valores o expresiones. En Basic Stamp el valor cero es False y el valor uno es True. Los valores o las expresiones están sujetos a la lógica siguiente:

F OR F = F

F OR T = T

T OR F = T

T OR T = T

El resultado retornado por OR es Verdadero o Falso.

EJEMPLO:

BS2:

```
A    VAR    Byte
B    VAR    Byte
Main:
 A = 93
 B = 89
 IF (A < 80) OR (B < 70) THEN
 DEBUG "Se encuentra en el promedio "   ' Cuando la condición es Verdadera. ELSE
DEBUG "No se encuentra en el promedio "    'Cuando la condición es Falsa.
 ENDIF
 END
```

4.2.1.15 XOR LOGICO

El operador XOR retorna un valor lógico de dos valores o expresiones. En Basic Stamp el valor cero es False y el valor uno es True. Los valores o las expresiones están sujetos a la lógica siguiente:

F XOR F = F

F XOR T = T

T XOR F = T

T XOR T = F

El operador de XOR puede estar confundiendo al principio. Una manera fácil de recordar la lógica es: Si uno o el otro es Verdad, pero no ambos entonces el resultado es Verdad, por otra parte el resultado es Falso. El resultado retornado por XOR será Verdad o Falso.

EJEMPLOS:

BS2:

```
turnL   VAR     Bit
turnR   VAR     Bit
Main:
  turnL = 0
  turnR = 1
  IF (turnL = 1) XOR (turnR = 1) THEN          'El uno o el otro es verdadero.
  DEBUG "Turning - ", "L" + ("R" - "L" * turnR)
  ELSE
```

```
DEBUG "Continue straight."          'Ambos verdaderos o ambos falsos
ENDIF
END
```

4.2.2 OPERADORES DE UNARIO

Son aquellos que trabajan con un solo argumento.

4.2.2.1 ABS (Valor Absoluto)

El operador de Valor Absoluto (ABS) convierte el signo de 16 BIT de un numero a un valor absoluto. Este es un número positivo que representa la diferencia entre ese número y 0.

EJEMPLO: Valor absoluto de -99 es 99. El valor absoluto de 99 también es 99.

BS2:

```
result VAR     Word
Main:
 result = -99
 DEBUG SDEC ? result                ' Muestra el  -99
 DEBUG SDEC ? ABS result            ' Muestra el 99
 END
```

4.2.2.2 NEGACIÒN

Este operador niega un numero de 16 BIT (Convierte a estos en sus 2 complementos).

EJEMPLO:

BS1)

```
SYMBOL  result = W1
Main:
```

```
result = -99                    ' Inicia el resultado.
result = result + 100           ' Suma 100 a este
DEBUG result                    ' Muestra el resultado (1)
END
```

BS2)

```
result  VAR     Word
Main:
  result = -99                  ' Inicia el resultado
  DEBUG SDEC result             ' Muestra el resultado (-99)
  result = result + 100         ' Suma 100 a esto.
  DEBUG SDEC result             ' Muestra el resultado (1)
  END
```

4.2.2.3 INVERTIR BIT ~

El operador Inverso son los complementos de los bits de un número. Cada BIT que contiene 1 se cambia por 0 y cada BIT que contiene 0 se cambia a 1. Este proceso es conocido como el " BITWISE NO" y el complemento de 1.

~0 = 1

~1 = 0

EJEMPLO:

```
result  VAR     Byte
Main:
```

```
result = %11110001                ' Almacena los bit en el resultado.
DEBUG BIN8 ? result               ' Muestra  %11110001
result = ~result                  ' Complemento del  result
DEBUG BIN8 ? result               ' Muestra  %00001110
END
```

4.2.2.4 SQR

SQR devuelve la raíz cuadrada de un valor. Como PBASIC solo trabaja con enteros, el resultado será siempre un entero de 8 bits no mayor que el valor original.

```
DEBUG CLS
FOR W0 = 1 TO 100 STEP 5
W1 = W0
W1 = SQR W1
DEBUG DEC3 W0,"-",DEC3 W1, CR
NEXT
STOP
```

4.2.2.5 DCD

DCD devuelve la potenciación en base 2 entre un rango de exponentes de (0 a 15). En otras palabras:

```
B0 = DCD 4 ´ Muestra B0 como % 00001000    Es equivalente a 24 = 16
B0 = DCD 5 ´ Muestra B0 como % 00010000    Es equivalente a 2 5 = 32
```

Ejemplo:

```
DEBUG CLS
W1 = 0
```

```
FOR B0 = 0 TO 15
W1 = DCD B0
DEBUG DEC2 B0,"-",BIN16 W1,"-",DEC5 W1, CR
PAUSE 10
NEXT
STOP
```

4.3.1 PARA EL CONTROL DEL PROGRAMA

4.3.1.1 BIFURCACIONES

4.3.1.1.1GOTO

Sintaxis:

GOTO Direcciones.

Nombre de cualquier función.

Función

Salto de un punto especifico en el programa por una dirección. Esta es una etique que especifica a donde ir. El uso del goto es ilimitado pero en un buen programa es minimizado.

El comando del goto fabricado por el Basic Stamp ejecuta el código que empieza en la dirección específica. El Basic Stamp lee el Pbasic este codifica de izquierda a derecha. El Goto ordena al Basic Stamp a saltar a la fuerza a otra sección de código.

EJEMPLO:

```
Start:
  GOTO Routine1
```

	BS1	La Familia de BS2

Routine 2:

```
  DEBUG "We're in routine #2", CR
  PAUSE 1000
  GOTO Routine3
Routine1:
  DEBUG "We're in routine #1", CR
  PAUSE 1000
  GOTO Routine2
Routine3:
  DEBUG "We're in routine #3", CR
  PAUSE 1000
  GOTO Routine1
```

4.3.1.1.2 IF THEN

Sintaxis:

IF *Condición* **THEN** *Dirección*

Función:

Evalúa la Condición y, si es verdad, va al punto en el programa marcado por la Dirección.

- **LA CONDICIÓN:** Es una declaración, como "x = 7" eso puede evaluarse como Verdadero o Falso. La Condición puede ser una relación muy simple o muy compleja.
- **LA DIRECCIÓN:** Es una etiqueta que especifica dónde ir en caso de la Condición es verdad.

Operadores de la comparación	=, <>, >, <, >=, <=	=, <>, >, <, >=, <=
Los Operadores de la Lógica condicionales	AND. OR	NOT, AND, OR, XOR
El formato de condición	El Valor de la Comparación inconstante; donde el Valor es una variable o constante	La Comparación del Valor 1 Valor 2; donde Valor 1 y Valor 2 pueden por cualquiera de variable, constante o expresión
Los paréntesis	No Permitido	Permitido

Si esa condición es verdad, va a un punto en el programa especificado por una etiqueta de dirección. La condición que SI... ENTONCES las pruebas son escritas como una mezcla de comparación y operadores de la lógica. Los operadores de la comparación disponibles son:

La comparación Operador Símbolo	La definición
=	El igual
<>	No el Igual
>	Mayor Que
<	Menos de
>=	Mayor Que o Iguala A
<=	Menos de o Iguala A

Las comparaciones siempre son escritas en la forma: la Comparación del Valor1 Valor2. Los valores a ser comparados pueden ser cualquier combinación de variables (cualquier tamaño), constantes, o expresiones. NOTA: En el BS1, las expresiones no se permiten como los argumentos, y el Valor1 (a la izquierda de comparación) debe ser una variable.

EJEMPLO:

BS1)

```
SYMBOL  value   = W1
Main:
  PULSIN 0, 1, value
  DEBUG #value, CR
  IF value < 4000 THEN Main
  DEBUG "Value was greater than 4000!"
  END
```

BS2)

```
value   VAR     Word
Main:
  PULSIN 0, 1, value
  DEBUG DEC value, CR
  IF (value < 4000) THEN Main
  DEBUG "Value was greater than 4000!"
  END
```

4.3.1.1.3 GOSUB

Sintaxis: *GOSUB* *Adress.*

Nombre de la Subrutina.

Nombre de la Función

Este sirve para que se dirija a un Subprograma. Cuando un programa del PBASIC alcanza un GOSUB, el programa ejecuta el principio del código a la etiqueta de dirección especificada. GOSUB también guarda la dirección de la instrucción que se sigue inmediatamente. Cuando el programa encuentra un orden del RETORNO, lo interpreta que va a la instrucción que sigue el más reciente de GOSUB."

	BS1*	BS2 Family
Maximum **GOSUB**s per program	16	256
Maximum nested **GOSUB**s	4	4

EJEMPLO:

Main:

GOSUB Hello

DEBUG "How are you?", CR

END

Hello:

DEBUG "Hello my friend." CR

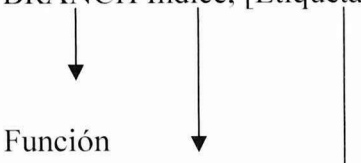
RETURN

Stop

4.3.1.1.4 BRANCH

Sintaxis:

BRANCH Índice, [Etiqueta0, Etiqueta1,... EtiquetaN]



Función

Variable/constante/expresión por lo general tipo BYTE.

Direcciones de referencia

Función

Salta o se dirige a la etiqueta señalada por índice sí esta en el rango.

BS1)

BRANCH value, (Case_0, Case_1, Case_2)

BS2)

BRANCH value, [Case_0, Case_1, Case_2]

Ejemplo:

Ejemplo 1

{$STAMP BS2}

caracter VAR BYTE

Principal:

DEBUG 2,1,1,"Seleccione una Fruta del [0 - 7]", CR

SERIN 16, 16468, [DEC1 caracter]

BRANCH caracter,[Uva, Pera, Manzana, Guineo, Melón, Chinola, Freza, Cereza]

```
DEBUG CLS
DEBUG 2,1,2,"No ha seleccionado correctamente"
PAUSE 1500
DEBUG CLS
GOTO Principal
Uva:
DEBUG 2,2,4,"Ha seleccionado Uva....."
GOTO Principal
Pera:
DEBUG 2,2,4,"Ha seleccionado Pera....."
GOTO Principal
Manzana:
DEBUG 2,2,4,"Ha seleccionado Manzana.."
GOTO Principal
Guineo:
DEBUG 2,2,4,"Ha seleccionado Guineo..."
GOTO Principal
Melón:
DEBUG 2,2,4,"Ha seleccionado Melón...."
GOTO Principal
Chinola:
DEBUG 2,2,4,"Ha seleccionado Chinola.."
GOTO Principal
```

Freza:

DEBUG 2,2,4,"Ha seleccionado Freza...."

GOTO Principal

Cereza:

DEBUG 2,2,4,"Ha seleccionado Cereza..."

GOTO Principal

4.3.1.2 CICLOS REPETITIVOS CONTROLADOS

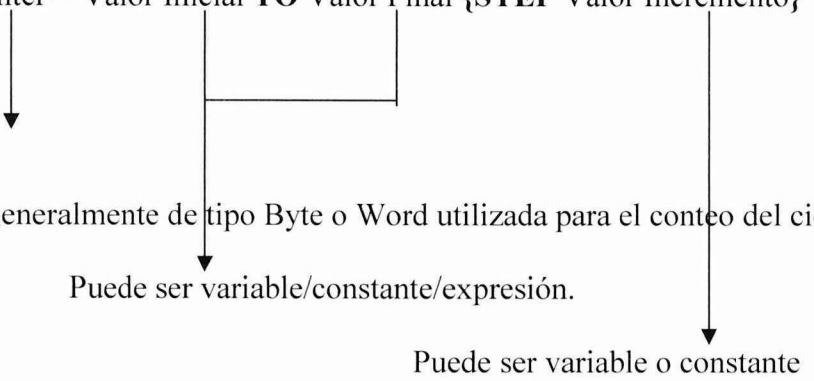
4.3.1.2.1 FOR...NEXT

Sintaxis:

FOR Counter = Valor Inicial **TO** Valor Final **{STEP** Valor Incremento**}**

{Cuerpo}

NEXT



Función: Crea un bucle programado entre un rango de valores iniciales y finales el cuerpo del bucle queda comprendido en el medio de FOR y NEXT, El bucle puede incrementar o decrementar la variable Counter acorde con el valor incremento establecido. Si no se establece un Valor Incremento asume que el incremento será de uno (1). El bucle finaliza cuando la variable Counter llegue al Valor Final establecido.

EJEMPLO:

BS1)

SYMBOL reps = W1

```
Main:
 FOR reps = 0 TO 300
  DEBUG reps
 NEXT
 END
```

BS2)

```
reps   VAR    Byte
Main:
 FOR reps = 0 TO 300
  DEBUG DEC ? reps
 NEXT
 END
```

4.3.1.3 ACCESO DE DATOS A LA EEPROM

4.3.1.3.1 DATA

Sintaxis:

{Etiqueta} DATA Dato tipo {, Dato tipo...}

Opcional nos indica el comienzo de donde empieza un dato.

Constante/ expresión, indica el valor a almacenar.

Función

Almacena datos en la memoria EEprom del BS2 en tiempo de programación.

EJEMPLO:

```
result        VAR    Word
```

Storage:

DATA Word 1125

Main:

READ 0, Word result

DEBUG DEC result

END

4.3.1.3.2 READ

Sintaxis:

READ Localización, Variable

Elementos:

* **Localización** puede ser una variable/constante/expresión de (0-2047) que indica la dirección en la memoria EEprom del BS2, la memoria EEprom del BS2 contiene 2048 direcciones en la que puede almacenar un BYTE por dirección.

* **Variable** es donde se almacena el dato leído. Por lo general es de tipo BYTE.

Limites: Dirección inicial 0; Dirección final 2047

Función:

Lee de la EEprom interna del BS2 un dato tipo BYTE previamente almacenado con la función DATA o con la función WRITE.

EJEMPLO:

```
índice VAR BYTE          'Variable para la localización
caracter VAR BYTE        'Variable para almacenar el contenido
DATA "BS2 es lo máximo"
FOR indice = 0 TO 15     'Dirección de posición 0 a 15 de la EEprom
```

READ indice, carácter 'Lee el contenido de la dirección

DEBUG carácter 'Imprime el contenido leído

NEXT

END

4.3.1.3.3 WRITE

Sintaxis:

WRITE Localización, Dato Tipo

Elementos:

• **Localización** puede ser una variable/constante/expresión de (0- 2047) que indica la dirección en la memoria EEprom del BS2, la memoria EEprom del BS2 contiene 2048 direcciones en la que puede almacenar un BYTE por dirección. Por lo general es de tipo WORD.

• **DatoTipo** puede ser una variable/constante/expresión específica el valor a almacenar. Por lo general es de tipo BYTE.

Limites: Dirección inicial 0; Dirección final 2047.

Función:

Puede escribe datos en la memoria EEprom en tiempo de ejecución.

EJEMPLO:

```
value   VAR     Word
Main:
  value = 1125
  WRITE 0, value.LowByte
  WRITE 1, value.HighByte
```

END

4.3.1.4 BÚSQUEDA DE DATOS Y TABLA DE DATOS

4.3.1.4.1 LOOKUP

Sintaxis:

LOOKUP Índice, [Valor0, Valor1,... ValorN], Variable.

Elementos:

• **Índice** puede ser una variable/constante/expresión de (0-255) indica la posición del valor en la lista a recuperar.

• **Valores** pueden ser una variable/constante/expresión de (0-65535) y son los valores de la lista.

• **Variable** es una donde se transfiere el valor indicado por índice en la lista de valores. Si el índice excede el número de elementos la variable permanece sin cambios.

Función

Recupera valores de una tabla de datos no mayor de 256 elementos, índice contiene la posición del valor a buscar en la lista, cuando índice recupera el valor lo transfiere a variable. Si el índice excede el número de elementos la variable permanece sin cambios.

Limites

Numero máximo de Valores	256
Valor inicial de índice	0
Si Indice es mayor que el numero de valores	La Variable permanece sin cambios

EJEMPLO 1

índice VAR BYTE

Resultado VAR BYTE

índice = 5

Resultado = 255

LOOKUP índice,[63,6,91,79,102,109,125,7,127,111],Resultado

DEBUG "El valor correspondiente es:", DEC3 Resultado

En este ejemplo, DEBUG imprime 109, el cual es el valor que se encuentra en la posición No. 5, de la lista de valores. Es posible listar caracteres ASCII, unas de las maneras de enviar información por pantallas de cristal líquido (LCD). Es con LOOKUP.

4.3.1.4.2 LOOKDOWN

Sintaxis:

LOOKDOWN Target, {Comparación} [Valor0, Valor1... ValorN], Variable

Elementos:

• **Target** puede ser una variable/constante/expresión de (0-65535) para ser comparada con los valores de la lista.

• **Comparación** es opcional, utiliza los operadores lógicos de IF...THEN, cuando se omite, se asume que la comparación es de (=) igualdad.

• **Valores** pueden ser una variable/constante/expresión de (0-65535) y son los que se compararan con Target.

• **Variable** es una variable por general tipo (byte), y es donde se almacena el número de posición de la lista en caso de que el target concuerde con uno de los valores de la lista. En caso de que no concuerden variable permanece intacta.

Función

Busca en una lista de valores el valor Target y si coinciden, almacena su posición física en Variable. Si el valor es el primero de la lista, Variable = 0. Si es el segundo, Variable = 1 y así, sucesivamente. Si no se encuentra, no se toma ninguna acción y Variable permanece sin cambios.

Limites

Numero máximo de Valores	256
Valor inicial de índice	0
Si el valor no esta en la lista	La Variable permanece sin cambios

Ejemplo:

Target VAR BYTE

Resultado VAR BYTE

Target = "n"

Resultado = 255

LOOKDOWN Target, ["Republica Dominicana"], Resultado

DEBUG "El valor igual a Target esta ubicado en la Pos.: ", DEC Resultado

END

4.3.1.4.3 RANDOM

Sintaxis:

RANDOM Variable

Elementos:

• **Variable** usualmente es de tipo WORD donde los bits internos son revueltos para generar un número al azar.

Función

Genera un número aleatorio o al azar en un rango de (0 a 65535).

Ejemplo:

```
ALEAT VAR WORD      'Define una variable tipo WORD
LOOP:               'Etiqueta de referencia LOOP
RANDOM ALEAT        'Genera un numero aleatorio y lo almacena en ALEAT
DEBUG DEC5 ALEAT, CR    'Imprime el numero ALEAT
PAUSE 250           'Espera una Pausa de 250 milisegundos
GOTO LOOP           'Vuelve a LOOP
```

4.3.1.5 SEÑALES DIGITALES

4.3.1.5.1 INPUT

Sintaxis:

INPUT Pin

Función

Declara el Pin especificado, como modo de entrada.

Elementos:

• **Pin** puede ser una expresión variable o constante de (0-15) del puerto de Entrada / salida del BS2. Este Pin se ajusta como modo de entrada.

Ejemplo:

```
INPUT 8                 'Declara P8 como entrada.
Repetir_Lectura:
PAUSE 250               ' Pausa de 250 mili-segundos.
IF IN8=0 THEN Repetir_Lectura  ' Lee el estado de p8
```

DEBUG "Fin de Lectura" ' Imprime Fin de Lectura"

END ' Fin del Programa.

4.3.1.5.2 OUTPUT

Sintaxis:

OUTPUT Pin

Función

Declara el Pin especificado, como modo de salida.

Elementos:

• **Pin** puede ser una variable/expresión/constante de (0-15) del puerto de Entrada / salida del BS2. Este Pin se ajusta como modo de salida automaticamente.

Ejemplo:

OUTPUT 4 ' Declara al Pin 4 como Salida

LOOP:

OUT4=1 ' Estable un 1 lógico en el Pin 4

PAUSE 500 ' Espera ½ segundo

OUT4=0 ' Estable un 1 lógico en el Pin 4

PAUSE 500 ' Espera ½ segundo

GOTO LOOP ' Se dirige a la etiqueta LOOP

4.3.1.5.3 REVERSE

Sintaxis:

REVERSE Pin

Función

Invierte la dirección de Entrada/Salida de un pin especificado.

Elementos:

• **Pin** puede ser una variable/constante/expresión de (0-15) del puerto de Entrada/salida del BS2. Este Pin se direcciona opuestamente a su estado lógico anterior.

Ejemplo:

```
' Program: REVERSE.BS2
'{$STAMP BS2}          ' STAMP directive
OUT0 = 0              ' Pone en low el Pin 0.
Again:
PAUSE 500             ' Espera (1/2 segundo).
REVERSE 0             ' Invierte la dirección del Pin 0.
GOTO Again            ' Repite para siempre.
```

4.3.1.5.4 HIGH

Sintaxis:

HIGH Pin

Función

Asigna un 1 lógico al Pin especificado, esta salida es compatible con la familia lógica TTL.

Elementos:

• **Pin** puede ser una variable/constante/expresión de (0-15) del puerto de Entrada / salida del BS2. Este Pin se direcciona como salida automáticamente.

Ejemplo:

```
HIGH 7                          ' Ajusta el Pin7 a 5+ Voltios (Enciende el Led)
DEBUG "Comando HIGH",CR ' Imprime "Comando HIGH"
PAUSE 1500                      ' Espera 1.5 Segundos
```

```
LOW 7                    ' Ajusta el pin7 a 0+ Voltios (Apaga el Led)
DEBUG "Comando LOW"      ' Imprime "Comando LOW"
PAUSE 1500               ' Espera 1.5 Segundos
END                      ' Fin del Programa
```

El comando HIGH 7, es equivalente a:

```
OUT7=1 ' Ajusta la Salida 7 a 5+ Voltio.
DIR7=1 ' Direcciona el Pin 7 como salida
```

La ventaja del comando HIGH es que direcciona el pin automáticamente como salida.

4.3.1.5.5 LOW

Sintaxis:

LOW Pin

Función

Asigna un 0 lógico al Pin especificado, esta salida es compatible con la familia lógica TTL.

Elementos:

• **Pin** puede ser una variable/constante/expresión de (0-15) del puerto de Entrada / salida del BS2. Este Pin se direcciona como salida automáticamente.

Ejemplo:

```
LOOP:
LOW 0                    'Ajusta el pin7 a 0+ Voltios (Apaga el Led)
PAUSE 500                'Pausa de ½ Segundo
HIGH 0                   'Ajusta el Pin7 a 5+ Voltios (Enciende el Led)
PAUSE 500                'Pausa de ½ Segundo
GOTO LOOP                'Retorna al Inicio
```

El comando LOW 7, es equivalente a:

OUT0=0 ‘ Ajusta la Salida 7 a 0 Voltios.

DIR0=1 ‘ Direcciona el Pin 7 como salida

La ventaja del comando LOW es que direcciona el pin automáticamente como salida. Y es más rapido.

4.3.1.5.6 TOGGLE

Sintaxis:

TOGGLE Pin

Función

Invierte el estado de salida de Pin especificado.

Elementos:

• **Pin** puede ser una variable/constante/expresión de (0-15) del puerto de Entrada/Salida del BS2. Este Pin se direcciona como salida automáticamente.

Ejemplo 1

' {$STAMP BS2} ' STAMP directive

Again:

PAUSE 500 ' Espera (1/2 segundo)

TOGGLE 0 ' Invierte la dirección del Pin 0

GOTO Again ' Repite para siempre

4.3.1.5.7 PULSIN

Sintaxis:

PULSIN Pin, Estado, Variable

Función

Mide el ancho de un pulso por el pin y el estado especificado y almacena el resultado en variable.

Elementos:

• **Pin** puede ser variable/constante/expresion de (0-15) que especifica el pin a utilizar. Este pin debe ser ajustado como entrada antes de utilizarse.

• **Estado** puede ser una variable/constante/expresion de (0-1) que especifica donde comenzara la medición del pulso, si con la transición de (0-1) el estado es (1) o con la transición de (1-0) el estado es (0).

• **Variable** es donde se almacena la duración del pulso medido en unidades de 2 µS. Por lo general es de tipo WORD.

Limites

Pulso Mínimo 2 µS

Pulso Máximo 131.07 ms

Ejemplo :

```
pulse_pin CON 0
pulse_estado CON 1
pulse_variable VAR WORD
DEBUG CLS
Repetir:
PULSIN pulse_pin, pulse_estado, pulse_variable
DEBUG HOME, DEC5 pulse_variable
PAUSE 500
GOTO Repetir
```

Para calcular el tiempo real del pulso:

Tr = Variable * 2 µs ' Tiempo Real del pulso en micro-segundos.

4.3.1.5.8 PULSOUT

Sintaxis:

PULSOUT Pin, Periodo

Función

Genera un pulso por el Pin y Periodo especificado.

Elementos:

• **Pin** puede ser variable/constante/expresion de (0-15) que especifica el pin a utilizar. Este pin debe ser ajustado como salida antes de utilizarse.

• **Periodo** puede ser variable/constante/expresion de (0-65535) que especifica la duración del pulso.

Limites

Pulso Mínimo 2 µS

Pulso Máximo 131.07 ms

Ejemplo:

```
LOW 15                  'Declara el Pin 15 como salida y un estado lógico bajo
PULSOUT 15, 50000       'Genera un pulso de 100 mili-segundos por el Pin 15
STOP                    'Detiene el programa
```

4.3.1.5.9 BUTTON

Sintaxis:

BUTTON Pin, Downstate, Delay, Rate, Workspace, Targetstate, Etiqueta.

Función

La instrucción BUTTON es utilizada para leer el estado de pulsadores momentáneos (Push-Botón), su funcionamiento es similar al de un teclado de una computadora. Incluye retrasos antes de tomar una acción, auto repeticiones y no-auto repeticiones. Cuando un Push-Botón es accionado bajo el dominio la instrucción BUTTON este se dirige a la dirección o etiqueta señalada.

Elementos:

• **Pin** puede ser variable/constante/expresión de (0-15), especifica el pin a utilizar. Este pin se declara como entrada automáticamente.

• **Downstate** puede ser variable o constante (0-1), especifica el estado lógico cuando se presiona el botón. Ver grafico.

• **Delay** puede ser variable o constante (0-255)y especifica cuánto tiempo el botón debe ser presionado antes comenzar la autorepeticiones. Delay se mide en los ciclos de la rutina de BUTTON. Delay tiene dos configuraciones especiales: 0 y 255. Si Delay es 0, no realiza ningún antirebotes o auto-repeticiones. Si Delay es
255, el botón realiza antirebotes, pero no auto-repeticiones.

• **Rate** puede ser variable o constante (0-255)y especifica el número de ciclos entre las auto-repeticiones. El Rate se expresa en los ciclos de la rutina de button.

• **Workspace** es una variable de trabajo que la función BUTTON, necesita para operar, Debe ser inicializada antes de utilizarla.

• **Targetstate** puede ser variable o constante (0-1) y especifica cual estado debe tomar el botón para que ocurra una acción a la dirección señalada. (0 = no presionado, 1 = presionado).

• **Etiqueta** es la dirección de referencia que específica a donde apuntara el programa si el botón es presionado.

Ejemplo:

```
btn_t VAR BYTE
pin0 CON 0 ' Conectar como esta en el diagrama
downstate CON 0
delay CON 255
rate CON 250
targetstate CON 1
Loop_Main:
BUTTON pin0, downstate, delay, rate, btn_t, targetstate, No_Press
DEBUG "*"
No_Press:
GOTO Loop_Main 'Retorna a Loop_Main
```

Luego cambie los parametros: delay, rate y targetstate.

4.3.1.5.10 COUNT

Sintaxis:

COUNT Pin, Periodo, Variable

Función

Cuenta él número de ciclos de una señal (0-1-0 o 1-0-1) en el pin especificado durante el período especificado en milisegundos y almacena él numero de ciclos en una variable.

Elementos:

• **Pin** puede ser variable/constante/expresión de (0-15), especifica el pin a utilizar para la entrada de señal. Este pin se declara como entrada automáticamente.

• **Período** puede ser variable/constante/expresión de (1-65535) especifica el tiempo en milisegundos durante los cuales contar.

• **Variable** donde se almacenaran los números de ciclos (generalmente es de tipo Word) o 16 Bits.

Limites

Ancho del pulso Minimo 8 µS

Máxima Frecuencia 120,000 Hz

Ejemplo

```
F_in CON 0                          ' Entrada de la señal por P0 del BS2
freq VAR WORD                       ' Variable
Main:
COUNT F_in, 10, freq                ' Cuenta él numero de ciclos en 10 mseg.
DEBUG CLS, "No. de Ciclos: ", DEC5 freq     ' Muestra 4 ciclos
GOTO Main                           ' Ir a Main
END                                 ' Fin del programa
```

4.3.1.5.11 XOUT

Sintaxis:

XOUT Mpin,Zpin,[House\Commando{Ciclos}{,House\Commando{\Ciclos}...}]

Función

Envía un código X-10 por la red eléctrica de 120AC/60Hz. A través de un dispositivo que sirve como interface entre la línea AC y el microcontrolador.

Elementos:

• **Mpin** puede ser una variable/constante/expresión de (0-15) del puerto de entrada / salida del BS2 para enviar la señal (modulada) X-10, a través del dispositivo de interface. Este pin es ajustado como salida automáticamente.

• **Zpin** puede ser una variable/constante/expresión de (0-15) del puerto de entrada / salida del BS2 que recibe la señal (cruce por cero) X-10, a través del dispositivo de interface. Este pin es ajustado como entrada automáticamente.

• **House** puede ser una variable/constante/expresión de (0-15) que especifica el código de la unidad representada por los valores de (0-15) los cuales representan las letras de la (A a la P) el los dispositivos.

• **Comando** puede ser una variable/constante/expresión de (0-30) que especifica el comando a enviar. Los valores de (0-15) corresponde con el código de la unidad de (1-16) y del (17-30) son códigos de control los cuales se describen en la tabla de códigos de control.

• **Ciclos** es opcional puede ser una variable/constante/expresión de (1-255) especifica él numero de veces que se transmitirá el código X-10. Si no se especifica el transmite dos veces por defecto.

Ejemplo:

```
Zpin    PIN    0              ' Zpin is P0
Mpin    PIN    1              ' Mpin is P1
HouseA  CON    0              ' House code A = 0
Unit1   CON    0              ' Unit code 1 = 0
Main:
 XOUT Mpin, Zpin, [HouseA\Unit1]      ' Get unit 1's attention..
```

```
XOUT Mpin, Zpin, [HouseA\UnitOn]     ' ..and tell it to turn on
END
```

4.3.1.6 COMUNICACIÓN ASINCRÓNICA

4.3.1.6.1 SERIN

Sintaxis:

SERIN Rpin{\Fpin},Baudmode,{Plabel,}{Timeout,Tlabel,}[InputData]

Función

Recibe uno ó más datos en el Pin especificado en formato estándar asincrónico RS-232.

Elementos:

• **Rpin** puede ser variable/constante/expresión de (0-16) que especifica el pin a utilizar. rpin es colocado como entrada en forma automática. Si se especifica rpin a 16, quiere decir que se utilizara el puerto de programación. Es decir, el que utiliza el DEBUG.

• **Fpin** puede ser variable/constante/expresión de (0-15) es opcional. Se utiliza si se quiere establecer una comunicación con control de datos, muy importante para comunicación de microcontroladores entre sí.

• **Baudmode** puede ser variable/constante/expresión de (0-65535) especifica la velocidad de transmisión y configuración.

• **Plabel** es un parámetro opcional indica en caso de que ocurra un error, en realidad Plabel es una etiqueta de referencia. Si ocurre un error saltara al nombre de la etiqueta. Este argumento solo puede utilizarse si el Baudmode es 7 bits, y paridad par.

• **Timeout** es un parámetro opcional, puede ser variable/constante/expresión de (0-65535) le indica a SERIN que si en el tiempo establecido por Timeout en milisegundos no arriban los datos, entonces salta a Tlabel.

• **Tlabel** es un parámetro opcional, que funciona con Timeout, Tlabel es una etiqueta de referencia, que indica que los datos no arribaron, en el tiempo establecido por Timeout.

• **InputData** es una lista de variables que serán recibidas a través del puerto serial RS-232. puede ser una variable o un arreglo de variables en forma matricial. De formato de texto, decimal, binario o hexadecimal

Ejemplo:

```
sData   VAR     Byte
Main:
  SERIN 1, 16780, [sData]                ' baudmode set for BS2
  END
```

4.3.1.6.2 SEROUT

Sintaxis:

SEROUT Tpin{\Fpin},Baudmode,{Timeout,Tlabel,}[OutData]

Función

Transmite uno ó más datos en el Pin especificado en formato estándar asincrónico RS-232.

Elementos:

• **Rpin** puede ser variable/constante/expresión de (0-16) que especifica el pin a utilizar. tpin es colocado como salida en forma automática. Si se especifica rpin a 16, quiere decir que se utilizara el puerto de programación. Es decir, el que utiliza el DEBUG.

• **Fpin** puede ser variable/constante/expresión de (0-15) es opcional. Se utiliza si se quiere establecer una comunicación con control de datos, muy importante para comunicación de microcontroladores entre sí.

• **Baudmode** puede ser variable/constante/expresión de (0-65535) especifica la velocidad de transmisión y configuración.

• **Timeout** es un parámetro opcional, puede ser variable/constante/expresión de (0-65535) le indica a SERIN que si en el tiempo establecido por Timeout en milisegundos no arriban los datos, entonces salta a Tlabel.

• **Tlabel** es un parámetro opcional, que funciona con Timeout, Tlabel es una etiqueta de referencia, que indica que los datos no arribaron, en el tiempo establecido por Timeout.

• **OutData** es una lista de variables que serán enviados a través del puerto serial RS-232. puede ser una variable o un arreglo de variables en forma matricial.

Ejemplo 1

'Envía por el puerto de programación el mensaje "Hola Mundo"

SEROUT 16, 16468,["Hola Mundo"]

4.3.1.7 COMUNICACIÓN SINCRÓNICA

4.3.1.7.1 SHIFTIN

Sintaxis:

SHIFTIN Dpin, Cpin, Modo,[Variable{\bits}{, Variable {\bits}...}]

Función

Recibe uno ó más datos en el Pin especificado en formato estándar sincrónico SPI.

Elementos:

• **Dpin** puede ser variable/constante/expresión de (0-15) que especifica el pin a utilizar. Dpin es colocado como entrada en forma automática. Dpin significa Data pin.

• **Cpin** puede ser variable/constante/expresión de (0-15) que especifica el pin a utilizar. Cpin es colocado en modo de salida automáticamente. Cpin significa Clock pin.

• **Modo** puede ser variable/constante/expresión de (0-3), o uno de 4 símbolos predefinidos. Esto le indican el orden de arribar los datos.

• **Variable** es una variable en la cual se almacenan los datos recibidos.

• **Bits** es opcional especifica cuantos bits deben arribar de (1-16).Si no se especifica arribaran 8 Bits.

Ejemplo:

Resultado VAR BYTE

SHIFTIN 0,1, MSBPRE,[Resultado]

' Data es pin 0

' Clock es pin 1

' El modo de Formato es MSBPRE

' Recibe 8 BITS por la variable [Resultado]

4.3.1.7.2 SHIFTOUT

Sintaxis:

SHIFTOUT Dpin, Cpin, Modo,[Data{\bits}{,Data{\bits}...}]

Función

Envía uno ó más datos en el Pin especificado en formato estándar sincrónico SPI.

Elementos:

• **Dpin** puede ser variable/constante/expresión de (0-15) que especifica el pin a utilizar. Dpin es colocado como salida en forma automática. Dpin significa Data pin.

• **Cpin** puede ser variable/constante/expresión de (0-15) que especifica el pin a utilizar. Cpin es colocado en modo de entrada automáticamente. Cpin significa Clock pin.

• **Modo** es un valor de (0-1) predefinidos e indican el orden de enviar los datos.

• **Data** es una variable/constante/expresión que contiene el Dato a ser enviado.

• **Bits** es opcional especifica cuantos bits debe enviar de (1-16).Si no se especifica envia 8 Bits.

Ejemplo:

D_pin CON 0 ' Data es pin 0

C_pin CON 1 ' Clock es pin 1

SHIFTOUT D_pin, C_pin, MSBFIRST,[250]

' El modo de Formato es MSBFIRST

' Envía el numero 250 de forma sincrónica

4.3.1.8 SEÑALES ANÁLOGAS

4.3.1.8.1 PWM

Sintaxis:

PWM Pin, Duty, Ciclo

Función

Convierte la señal digital en análoga por modulación de pulsos.

Elementos:

• **Pin** puede ser variable/constante/expresion de (0-15) que especifica el pin a utilizar. Este pin debe ser ajustado como salida antes de utilizarse.

• **Duty** puede ser variable/constante/expresion de (0-255) que especifica la variación de voltaje análogo de (0-5V).

• **Ciclo** puede ser variable/constante/expresion de (0-255) que especifica la duración de la señal PWM.

Limites

Unidad en Ciclo 1 ms

Formula para calcular el voltaje V = (Duty/255) * 5 Voltios

Tiempo de carga Ciclos C = 4*R*C

Ejemplo:

Repetir:

PWM 0, 100,40

PAUSE 1000

GOTO Repetir

4.3.1.8.2 RCTIME

Sintaxis:

RCTIME Pin, Estado, Variable

Función

Mide el tiempo en el cual se carga o descarga un capacitor conformado por un circuito (RC) Resistor y capacitor, el que se mida la carga o descarga del capacitor dependerá del parámetro Estado el cual puede ser (0-1).

Elementos:

• **Pin** puede ser variable/constante/expresión de (0-15) que especifica el pin a utilizar. Cuando finaliza la función este Pin se convierte en entrada.

• **Estado** puede ser una expresión/variable/constante de (0-1) que especifica si la medición será realizada durante la carga o descarga del capacitor a medir.

• **Variable** es donde se almacena la duración del tiempo medido en unidades de 2 μS. Por lo general es de tipo WORD.

Limites

Medición Mínima 2 µS

Medición Máxima 131.07 ms

Ejemplo:

```
Resultado VAR WORD          'Se define una variable tipo WORD
HIGH 15                     'Se descarga el capacitor polarizándolo (+5)
PAUSE 1                     'Espera un milisegundo para asegurar la descarga
RCTIME 15,1, resultado      'Inicia la función RCTIME por el pin 15
DEBUG DEC5 resultado        'Imprime el resultado del tiempo
```

4.3.1.9 FUNCIONES DE TIEMPO

4.3.1.9.1 PAUSE

Sintaxis:

PAUSE Periodo

Función

Detiene el programa momentáneamente por el periodo especificado.

Elementos:

• **Periodo** puede ser una expresión/variable/constante de (0-65535) especifica la duración de la pausa en milisegundos.

Limites

Pausa Mínima 1 ms

Pausa Máxima 65535 ms (65.535 Segundos)

Ejemplo:

```
Flash: ' Etiqueta de referencia
LOW 0 ' Estado lógico 0
```

PAUSE 500 ' Espera ½ segundo

HIGH 0 ' Estado lógico 1

PAUSE 500 ' Espera ½ segundo

GOTO Flash ' Volver a Flash

Los retrasos de tiempo producidos por PAUSE están basados en la precisión del cristal de 20MHz del BS2, con una precisión de ±1 un por ciento.

4.3.1.10 FUNCIONES DE SONIDO

4.3.1.10.1 DTMFOUT

Sintaxis:

DTMFOUT Pin, { OnTime ,OffTime,} [Tone {,Tone...}]

Función

Genera un tono doble de multi-frecuencia mejor conocido como: dualtone, multifrecuency (DTMF), es el codigo de marcado de los teléfonos de teclas.

Elementos:

• **Pin** puede ser variable/constante/expresión (0-15), especifica el pin para la salida de la señal. Este pin se declara temporalmente como salida durante la generación del (DTMF). Después de la generación del tono, el pin se deja en modo de entrada, incluso si era previamente una salida.

• **OnTime** es opcional puede ser variable/constante/expresión (0-65535), especifica la duración del tono en milisegundos. Por defecto este valor es de 200 ms.

• **OffTime** es opcional puede ser variable/constante/expresión (0-65535), especifica la duración de pausa de silencio en milisegundos entre cada tono, si se especifica más de un tono. Por defecto este valor es de 50 ms.

• **Tone** puede ser variable/constante/expresión (0-15), especifica los tonos DTMF a generar. Los tonos del 0 al 11 corresponden al teclado estándar del teléfono, mientras del 12 al 15 es la cuarta columna utilizado por equipos de teléfonos de prueba y radio aficionados. Corresponden a las teclas extendidas (A –D).

EJEMPLO:

```
Main:
  FOR phone = 0 TO 2
    ' retrieve address
    LOOKUP phone, [Parallax, ParallaxFax, Information], eeLoc
    GOSUB Dial_Number
    PAUSE 2000
  NEXT
  END
Dial_Number:
  DO
    READ eeLoc, eeByte              ' Retrieve byte from EEPROM
    eeLoc = eeLoc + 1               ' point to next pair of digits
    FOR hiLo = 0 TO 1               ' Dial upper and lower digits
      IF (dtDig = $F) THEN EXIT     ' Hex $F is end-of-number flag
      DTMFOUT Spkr,                 ' dial digit
            150 */ TmAdj, 25, [dtDig] ' 150 ms on, 25 ms off
      eeByte = eeByte << 4          ' Shift in next digit
    NEXT
```

```
LOOP UNTIL (dtDig = $F)
RETURN
```

4.3.1.10.2 FREQOUT

Sintaxis:

FREQOUT, Pin, Periodo, freq1{, freq2 }

Función

Generan uno o dos tonos de señales senosoidal durante un periodo especificado.

Elementos:

• **Pin** puede ser variable/constante/expresión de (0-15), especifica el pin para la salida de la señal. Este pin se declara como salida.

• **Periodo** puede ser variable/constante/expresión de (0-65535) especifica la permanencia del tono a generar. La unidad del periodo es de un 1 milisegundo.

• **Freq1** puede ser variable/constante/expresión de (0-32767) especifica la frecuencia en hertz del primer tono.

• **Freq2** puede ser variable/constante/expresión exactamente igual que Freq1. Cuando se especifican dos frecuencias, lo que se obtiene es la mezcla de los dos tonos especificados. Freq1 y Freq2 se rigen por el mismo Periodo.

Limites

Unidad en Periodo 1 ms

Unidad en Frecuencia 1 Hz

Rango de frecuencia 0 a 32767 Hz

Ejemplo:

```
PAUSE 1000                    ' Espera un segundo
```

```
i VAR BYTE                    ' Declara una variable tipo Byte
f VAR WORD                    ' Declara una variable tipo Word
C CON 523                     ' Nota Musical C
D CON 587                     ' Nota Musical D
E CON 659                     ' Nota Musical E
G CON 784                     ' Nota Musical G
R CON 8                       ' Silencio
FOR i=0 TO 28          ' Ciclo controlado para reproducir las 29 notas musicales.
LOOKUP i,[E,D,C,D,E,E,E,R,D,D,D,R,E,G,G,R,E,D,C,D,E,E,E,E,D,D,E,D,C],f
FREQOUT 15, 225,f,(f-8) MAX 32768
NEXT
END                           ' Fin del programa
```

4.3.1.11 CONTROL DE ENERGIA

4.3.1.11.1 SLEEP

Sintaxis:

SLEEP Segundos

Función

Pone el BS2 en modo de bajo consumo de energía por él numero de segundos especificados.

Elementos:

• **Segundos** puede ser una variable/constante/expresión de (1-65535) especifica la duración en el que el BS2 estará en modo descanso,la unidad de un periodo en realidad para el BS2 hay que multiplicarla por 2.3 segundos.

Limites

Consumo de corriente en modo Normal 8 mA

Consumo de corriente en modo NAP 40 uA

Ejemplo:

DEBUG CLS 'Limpia la pantalla

HIGH 0 'Enciende el LED

Duerme: 'Etiqueta de Referencia

DEBUG "Voy a Dormir", CR

SLEEP 10 'Duerme por 23 Segundos

DEBUG "Estoy Despierto", CR

PAUSE 500 'Espera ½ segundo

GOTO Duerme 'Retorna a Duerme

4.3.1.11.2 NAP

Sintaxis:

NAP Periodo

Función:

El BS2 entra en modo de descanso por un periodo especificado. El consumo de energía se reduce como indica la tabla. Asumiendo que no hay cargas conectadas.

Elementos:

• **Periodo** puede ser una variable/constante/expresión de (0-7) especifica la duración en el que el BS2 estará en modo descanso, según la siguiente expresión: la Duración es (2^Periodo)*18ms.

Limites

Consumo de corriente en modo Normal 8 mA

Consumo de corriente en modo NAP 40 Ua

Ejemplo:

```
Setup:
  LOW 0                    ' turn LED on
Snooze:
  NAP 4                    ' nap for 288 ms
  GOTO Snooze
  END
```

4.3.1.11.3 END

Sintaxis:

End

Función

Finaliza el programa, poniendo al BS2 en modo de bajo consumo indefinidamente. El comando END se utiliza más bien como parámetro de referencia final de un programa. Es opcional y es raramente utilizado.

Limites

Consumo normal en operación 8 mA

Consumo después de END 40 µA

4.3.1.12 DEPURACIÓN DE PROGRAMA

4.3.1.12.1 DEBUG

Sintaxis:

DEBUG Outputdata {, Outputdata...}

Función

Visualiza variables y mensajes por la pantalla de la PC en combinación con el editor del BS2. Este comando es utilizado para visualizar textos y números en varios formatos.

Elementos:

• **OutputData** salida de datos pueden ser variable/constante/expresión, del rango comprendido entre (0–65535) especifica la salida de datos. La salida de datos puede estar en caracteres ASCII (Texto entre comillas " ", y caracteres de control), los números decimales (0–65535), los números hexadecimales ($0000-$FFFF), y los números binarios (%0000000000000000-%1111111111111111). La data numérica puede ser modificada con formatos como se explicara mas adelante.

Ejemplo:

DEBUG "Aprender BS2 es muy fácil!" 'Mensaje de prueba.

END

Después de que usted corra este programa con (Ctrl + R), el editor del BS2 abrirá una ventana que se llama "Debug Terminal" y usted podrá visualizar:

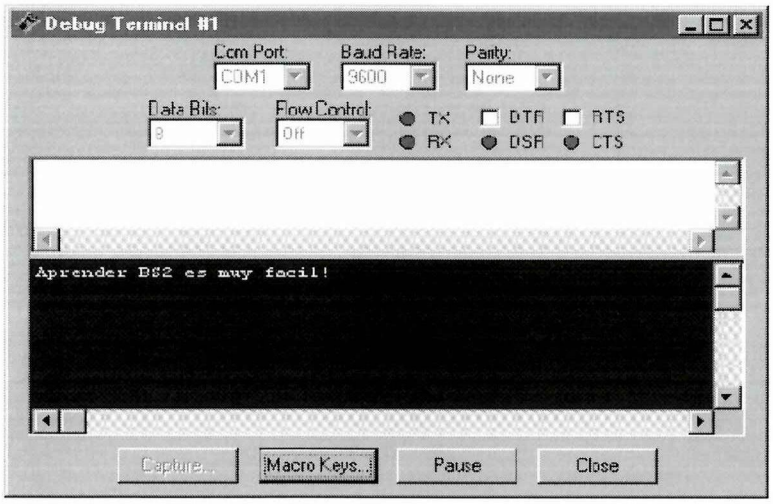


Figura: Pantalla tipica de la función DEBUG.

Presionando cualquier tecla con excepción de la barra espaciador se puede eliminar la ventana del Debug Terminal. Realmente el BS2 guarda el mensaje en su memoria EEPROM, para volver a ejecutar el DEBUG presione (CTRL + D) aparecerá nuevamente la ventana del Debug Terminal luego inicialice el BS2 presionando el botón de Reset y el mensaje aparecerá nuevamente.

Se pueden enviar múltiples datos y comandos con una solo una instrucción DEBUG separándolos por comas (,).

UNIVERSIDAD TÈCNICA DE COTOPAXI

CARRERA DE CIENCIAS DE LA INGENIERÍA Y APLICADAS

ESPECIALIDAD DE SISTEMAS COMPUTACIONALES E INFORMÀTICA

PROYECTO DE TESIS

TEMA:

"EL DESARROLLO DE UN PROTOTIPO DE ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR ASISTIDO POR UN COMPUTADOR PARA LA UNIVERSIDAD TÉCNICA DE COTOPAXI".

PROMOCIÓN:

Sexta.

POSTULANTES:

Herrera Calderón Alexandra de los Ángeles.

Uribe Monga Mariela Cristina.

DIRECTOR DE TESIS:

Ing. Silva Marco Polo.

Latacunga-Ecuador

2005

PROLOGO

La robótica es una de las especialidades de la ingeniería más complejas que existen hoy en día, esta ha involucrado tres principios en áreas diferentes, pero que son vitales para gestionar proyectos que permiten diseñar y construir robots de éxito, estas son la Informática, la Electrónica, y el Diseño Estructural.

Uno de los principales campos en que la robótica brinda apoyo es en la educación, nace como una disciplina aleada con la informática educativa que tiene como finalidad extender las habilidades intelectuales de los estudiantes apoyando la calidad de su aprendizaje.

La realización del proyecto posee un alto grado de complejidad, será necesario contar con laboratorios de electrónica e informática en cambio para su demostración solo se requerirá de maquinas básicas, que están al alcance de toda Institución, como es el caso de una computadora.

1. -SELECCIÓN Y DELIMITACION DEL TEMA

Los robots a nivel mundial han tenido un desarrollo considerable en instituciones y compañías. Por lo que ha permitido ayudar cada vez más a los humanos en la fabricación de nuevos productos, el mantenimiento de las infraestructuras y el cuidado de hogares y empresas; esto ha permitido optimizar la utilización de los recursos y agilizar procesos productivos, informativos, e industriales entre otros.

A pesar de que las instituciones y compañías no cuentan con los suficientes recursos no pueden estar fuera del desarrollo de la ciencia y la tecnología. En nuestro País nos han permitido acceder a mucha información que es la base para el desarrollo de esta y muchas áreas no explotadas, esto facilita desarrollo científico, donde permite que los procesos informativos se realicen en forma más eficiente y rápida.

Como se ha podido detectar a través de un diagnostico desarrollado en la Universidad Técnica de Cotopaxi, se puede determinar que la misma no se encuentra involucrada en aplicaciones científicas y tecnológicas debido a la escasa motivación por parte de estudiantes que laboran en dicha institución. La creación de un robot incentivara a todas las personas que se encuentran involucradas en cada una de las carreras técnicas, a que no limiten sus conocimientos y que permitan desarrollar su propia tecnología.

Ante lo expuesto anteriormente el grupo investigador se propone, **"EL DESARROLLO DE UN PROTOTIPO DE ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR ASISTIDO POR UN COMPUTADOR PARA LA UNIVERSIDAD TECNICA DE**

COTOPAXI ". La misma que se realizara bajo el método científico, permitiendo incrementar el incentivo en los estudiantes; para el desarrollo y aporte tanto científico como tecnológico.

El Desarrollo del prototipo de robot se realizara mediante partes electrónicas lo que se refiere a sus extremidades inferiores y en sus extremidades superiores se le acoplara un cuerpo elemental. El mismo que será asistido por un computador.

2. PLANTEAMIENTO DEL PROBLEMA

Por el momento la Universidad Técnica de Cotopaxi no se ha involucrado en aplicaciones de robótica, las cuales ha perjudicado el desarrollo científico y tecnológico de la institución, esta limitación de aprendizaje no permite que los estudiantes desarrollen todas sus habilidades adquiridas en los diferentes años de estudios, existiendo un analfabetismo tecnológico científico al cual esta siendo sometidos los estudiantes.

La escasa motivación de los estudiantes de la Universidad Técnica de Cotopaxi, en el desarrollo de proyectos tecnológicos y científicos ha tenido una influencia considerable, limitándoles a desarrollar trilladas aplicaciones que ningún aporte real hacen el desarrollo tecnológico de la Universidad.

La no presencia de tecnología en las diferentes aulas de clases en la Universidad Técnica de Cotopaxi, no permite que los estudiantes tengan ambientes de aprendizaje interdisciplinarias, limitando sus habilidades para estructurar investigaciones y resolver problemas concretos, lo que dificulta su interacción con otras tecnologías y áreas del conocimiento.

Las bases teóricas para el desarrollo de este tipo de proyectos se imparten en la especialidad, estas son la Inteligencia Artificial y la Electrónica, pero nunca se las ha llevado a la práctica. Es ahora cuando se tiene la oportunidad de realizar una tesis y momento de cristalizar un proyecto ambicioso que se ha convertido en una prioridad.

3.-JUSTIFICACIÓN

La presente investigación tecnológica científica será el complemento para que la Universidad Técnica de Cotopaxi, se involucre en aplicaciones de Robótica contribuyendo al desarrollo científico y tecnológico en dicha institución, y por ende, ayudando a que el aprendizaje obtenido en los diferentes años de estudios sean puestos en práctica, y con ello alcanzar un porcentaje más bajo de analfabetismo tecnológico científico en los estudiantes.

Muchos de los avances tecnológicos se han producido en el área de la electrónica que unidas a las computadoras a revolucionado nuestra vida, y nos han incentivado a investigar, como es el caso del desarrollo de este tipo de proyectos, para de esta manera contribuir el avance tecnológico y científico de esta institución.

Con este proyecto, estamos brindando a las nuevas generaciones la oportunidad de despertar el interés por la investigación científica y la exploración de campos a fines a la carrera de sistemas, como es la creación de robots controlados por computadoras que tengan la capacidad de hablar y caminar. Esto permitirá explotar diferentes campos principalmente de la industria. Ayudando a que los estudiantes desarrollen tecnología propia, por ende permitiéndoles así la explotación de sus conocimientos, y su perfil profesional.

Con esta investigación contribuimos a que la Universidad Técnica de Cotopaxi, inserte docentes con un espíritu innovador, para el desarrollo de proyectos tecnológicos y científicos, teniendo una influencia considerable en los estudiantes, permitiéndoles desarrollar proyectos investigativos y por ende desarrollar su propia tecnología, contribuyendo al posicionamiento de nuestra alma mater.

Por lo tanto la factibilidad de llevar a cabo este proyecto es una realidad que permitirá aportar al avance investigativo científico y tecnológico de la Universidad Técnica de Cotopaxi. Además con la integración de tópicos de Informática y Electrónica en las asignaturas, se lograra reutilizar las tendencias tecnológicas y naturales del ser humano de hacer cosas positivas como construir y desarrollar maquinas con tecnología de punta.

4.- OBJETIVOS

4.1 OBJETIVO GENERAL

- Desarrollar un prototipo de robot con capacidad de hablar y caminar asistido por un computador, que permita introducir aplicaciones de robótica en la Universidad Técnica de Cotopaxi.

4.2 OBJETIVOS ESPECIFICOS

- Determinar los requerimientos materiales, estructurales, electrónicos, y de software que nos permitan diseñar un prototipo de robot.

- Diseñar un prototipo de robot con capacidad de hablar y caminar controlado por un computador.

- Probar el funcionamiento del prototipo de Robot que pueda caminar y hablar asistido por un computador.

- Demostrar que las Ciencias de la Informática y la Electrónica están íntimamente ligadas al punto de que el software y el hardware no pueden funcionar si no se complementan.

5. MARCO TEORICO

5.1. ANTECEDENTES

Se cuenta con mucha información bibliográfica en lo que respecta al área de investigación del proyecto ya que es una disciplina que esta en pleno auge en este siglo. Permitiéndonos obtener con facilidad la información requerida para la investigación.

Para la realización de este proyecto se contara con ayuda del personal docente de la universidad y personal externo la cual facilitara el trabajo investigativo científico contribuyendo al posicionamiento de nuestra alma mater.

Este proyecto se lo realizara para incentivar a que los estudiantes puedan desarrollar investigación científica y tecnológica. Permitiéndoles poner en práctica todos sus conocimientos adquiridos. Dándoles la oportunidad a las nuevas generaciones que desarrollen investigación.

Por lo tanto la factibilidad de llevar a cabo este proyecto es una realidad que permitirá reducir el analfabetismo tecnológico y contribuir al desarrollo científico investigativo de la Universidad Técnica de Cotopaxi.

5.2 BASES TEORICAS

PROTOTIPO

Según LÓPEZ PELÁEZ, (2001), "Es un modelo a escala o facsímil de lo real, pero no tan funcional para que equivalga a un producto final, ya que no lleva a cabo la totalidad de las funciones necesarias del sistema final, proporcionando una retroalimentación temprana por parte de los usuarios acerca del sistema "

Desde el punto de vista de WEICHBRODT, B. (2000), "Prototipo se usa para obtener los requerimientos del usuario, su principal propósito es obtener y validar los requerimientos esenciales, manteniendo abiertas, las opciones de implementación. Esto implica que se deben tomar los comentarios de los usuarios, pero también se debe volver a los objetivos para no perder la atención."

De lo citado se puede decir que el prototipo son los primeros equipos realizados en los laboratorios de desarrollo. Suelen estar realizados en plan artesanal y contienen diversas modificaciones que los diseñadores van incluyendo según avanzan en el equipo.

Este se puede referir a cualquier tipo de máquina en pruebas, o un objeto diseñado para una demostración de cualquier tipo.

ROBOT

Según WEICHBRODT, B. (2000), "Es un dispositivo generalmente mecánico, que desempeña tareas automáticamente, ya sea de acuerdo a supervisión humana directa, a través de un programa predefinido o siguiendo un conjunto de reglas generales, utilizando técnicas de inteligencia artificial. Generalmente estas tareas reemplazan, asemejan o extienden el trabajo humano, como ensamble en manufactura, manipulación de objetos pesados o peligrosos, trabajo en el espacio, etc."

Desde el punto de vista de LÓPEZ PELÁEZ, (2001), "Un robot es un manipulador multifuncional, reprogramable diseñado para mover materiales, componentes, herramientas o dispositivos especializados a través de movimientos programados variables para la consecución de una variedad de tareas."

De lo citado se puede decir que un Robot es una máquina controlada por ordenador y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Donde las tareas que realizan los hombres son remplazadas por una maquina facilitando su trabajo especialmente en lugares peligrosos.

PROTOTIPO DE ROBOT

Según KINGHT, A.L. (1989), "Es un modelo de un robot, pero no tan funcional, ya que no lleva a cabo la totalidad de las funciones necesarias del proceso final, no es completamente seguro en un cien porciento".

Desde el punto de vista de OLLERO, Aníbal (2001); "Un prototipo de Robot es un ejemplar original cuyo dispositivo es generalmente mecánico, que desempeña tareas automáticas o primer molde de un robot".

De lo citado se puede decir que un Prototipo de Robot es una máquina controlada por ordenador y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno. Donde estos son desarrollados en los laboratorios Suelen estar realizados en plan artesanal y contienen diversas modificaciones que los diseñadores van incluyendo según avanzan en el equipo

ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR

Según URQUIZO, Patricio (2001); "Un dispositivo mecánico, que desempeña tareas automáticamente permitiéndole hablar y caminar, ya sea de acuerdo a supervisión humana directa, a través de un programa predefinido o siguiendo un conjunto de reglas generales, utilizando técnicas de inteligencia artificial."

Desde el punto de vista LÓPEZ PELÁEZ, A. (2000a), "Un robot con capacidad de hablar y caminar es un manipulador multifuncional, reprogramable diseñado para mover materiales a través de movimientos programados ya sea mediante la utilización de ruedas o patas mecánicas para la consecución de una variedad de tareas. En su capacidad para hablar los robot son utilizados para seguir ordenes de los humanos facilitándoles a realizar trabajos peligrosos".

De lo citado anteriormente se puede decir que un robot con capacidad de hablar y caminar es un dispositivo que puede trasladarse de un lugar a otro realizando actividades que el hombre no puede realizar teniendo en cuenta que es capaz de hablar y tomar sus propias decisiones.

5.3 DICCIONARIO DE TERMINOS

AUTÓMATA.- es una máquina que siempre repite el mismo proceso. Por ejemplo, el controlador de semáforos.

AMBIENTE VIRTUAL.- Es un ambiente sintético en donde se establece una relación entre el usuario y el sistema que compone este espacio.

CINEMÁTICA.- la cinemática del robot trata de cómo se mueve el mismo, dado que la dirección adopta tal el ángulo y que cada rueda gira tantas veces.

CONTROLADOR DE POSICIÓN.- es el dispositivo que se encarga de regular el movimiento de los elementos del brazo, y de todo tipo de acciones, cálculos y procesos de información. La complejidad del control varía con los parámetros que se manejan, existiendo varias categorías de controlador.

ELEMENTOS MOTRICES O ACTUADORES.- (motores neumáticos, mecánicos o eléctricos que suministran la fuerza de entrada para el movimiento de los ejecutores).

ENGRANAJES.- Son empleados para trasmitir un esfuerzo, estos son usados para cambiar la velocidad y la dirección o el sentido de rotación, o la intensidad en el momento de la fuerza. El engranaje pequeño aumenta la fuerza, mientras que el engranaje grande reduce la fuerza.

FUENTES DE MOVIMIENTO.- Las fuentes de movimiento son las que le otorgan movimiento al robot. Una de las más utilizadas es el motor eléctrico. Un motor es un dispositivo que convierte la energía eléctrica en energía mecánica rotacional que se utiliza para darle movimiento a ruedas y otros medios de locomoción.

GRADOS DE LIBERTAD.- Los grados de libertad son los parámetros que se precisan para determinar la posición y la orientación del elemento Terminal del manipulador. También se pueden definir, como los posibles movimientos básicos (giratorios y de desplazamiento) independientes.

HIDRÁULICA.- Es una aplicación de la mecánica de fluidos en ingeniería, para construir dispositivos que funcionan con líquidos, por lo general agua o aceite. La hidráulica resuelve problemas como el flujo de fluidos por conductos o canales abiertos y el diseño de presas de embalse, bombas y turbinas.

INTERFASE.- es el medio de comunicación entre el computador y la máquina o los sensores.

INTERACTIVIDAD.- Es la "acción entre" dos agentes, acción recíproca, es un continuo intercambio de respuestas entre agentes emisores y receptores. El intercambio de información entre computadora y usuario a través de prótesis o extensiones del ser humano, tal como los lentes VR, trajes llenos de sensores, mouse, entre otros.

INTELIGENCIA ARTIFICIAL.- Es un termino general que abarca un amplio rango de ideas referidas a computadoras aproximándose o simulando la inteligencia humana. Central a este concepto resulta la noción de adaptabilidad, a la solución de problemas en lugar de, solamente, limitarse a seguir los pasos establecidos por su programador.

MÁQUINA.- Es un aparato transformador de energía. Por ejemplo, un taladro transforma E eléctrica en E mecánica y E calórico.

MANIPULADOR.- Es el ejecutor final para agarrar o coger objetos.

MEDIOS DE LOCOMOCIÓN.- Los medios de locomoción son sistemas que permiten al robot desplazarse de un sitio a otro si éste debe hacerlo. El más utilizado y simple es el de las ruedas y le siguen en importancia las piernas y las orugas.

MEDIOS DE AGARRE.- Algunos robots deben sostener o manipular algunos objetos y para ello emplean dispositivos denominados de manera general medios de agarre.

MICROPROCESADOR.- Unidad de proceso y corazón del ordenador que funciona como un agente de tráfico ya que controla la circulación de información dentro del ordenador, así como las transformaciones de los datos que se soliciten.

NO-LINEALIDAD.-Es cuando el acceso a la información almacenada puede realizarse desde cualquier punto de la misma, sin necesidad de recorrerla desde el principio para encontrar algún segmento, el desarrollo del hipertexto ha permitido este tipo de interacción.

RENDER.- Es el proceso mediante el cual la computadora calcula las vistas de un objeto tridimensional que será visualizado en la pantalla.

RESOLUCIÓN DEL MANDO.- Es el incremento más pequeño de movimiento en que el robot puede dividir su volumen de trabajo. La resolución espacial depende de dos factores: los sistemas que controlan la resolución y los robots las inexactitudes mecánicas.

ROBOT.- Es un sistema compuesto por un computador, una interfase electrónica y una máquina provista eventualmente de sensores. «Un robot es un manipulador multifuncional reprogramable diseñado para mover materiales, componentes, herramientas o dispositivos especializados a través de movimientos programados variables para la consecución de una variedad de tareas.»

ROBÓTICA.- es la rama de la Inteligencia Artificial que se ocupa de las máquinas inteligentes. Disciplina que se ocupa de cuanto concierne al diseño y construcción de robots.

SENSORES.- Permiten al robot a manejarse con cierta inteligencia al interactuar con el medio. Son componentes que detectan o perciben ciertos fenómenos o situaciones.

TECNOLOGÍA.- Es un conjunto ordenado de instrumentos, conocimientos, procedimientos y métodos aplicados en las distintas ramas industriales.

TECNOLOGÍA FIJA.- No esta cambiando continuamente (siderúrgica, refinerías de petróleo, cemento y petroquímica).

TECNOLOGÍA FLEXIBLE.- Tiene varias y diferentes formalidades ejemplos:

Industria alimenticia, automotriz, medicamentos, etc.

6. ESQUEMA DE CONTENIDOS

CAPÌTULO I

CONOCIMIENTOS DE LA ROBÓTICA

1.1. ROBÓTICA

1.2. ESQUEMA GENERAL DEL SISTEMA ROBOT

1.3. ROBOT MANIPULADORES

1.4. ROBOTS MÓVILES

1.5. ROBOT AUTÓNOMOS Y TELERROBÓTICA

1.6. MORFOLOGÍA DE LOS ROBOTS

1.6.1. ESTRUCTURA DE ROBOTS MANIPULADORES

1.6.2. NUEVAS ESTRUCTURAS PARA ROBOTS MANIPULADORES

1.6.3. ROBOTS MÓVILES

CAPÌTULO II

DETERMINACIÓN DE LOS REQUERIMIENTOS

2.1. INTRODUCCIÓN A LOS REQUERIMIENTOS

2.2.INTRODUCCIÓN DE LOS CASOS DE USO

2.3. MODELO DE CASOS DE USO DE LA APLICACIÓN

2.4.INICIO DEL CICLO DE DESARROLLO

2.5. MODELO CONCEPTUAL DE LA APLICACIÓN

2.6. DICCIONARIO DE DATOS

2.7. ESPECIFICACIONES ADICIONALES

2.8. ANÁLISIS DE CIRCUITOS

CAPÌTULO III

FASE DE DISEÑO

3.1. INTRODUCCIÓN AL MODELO DE ANÁLISIS Y DISEÑO

3.1.1.DESCRIPCIÓN DE LOS CASOS DE USO

3.1.2. DIAGRAMAS DE CLASES DE DISEÑO

3.1.3.DISEÑO DEL PROTOTIPO DE ROBOT

3.2. MODELO DE DATOS

3.3.ALGUNOS ASPECTOS DEL DISEÑO DEL SISTEMA

3.4. MODELO DE DESPLIEGUE

3.5.CASO DE PRUEBAS

CONCLUSIÒNES Y RECOMENDACIÒNES

CONCLUSIÒNES

RECOMENDACIÒNES

BIBLIOGRAFÌA

CONSULTADA

CITADA

ANEXOS

7. METODOS Y TECNICAS DE INVESTIGACION

7.1 DEFINICION DEL TIPO DE INVETIGACIÒN

Por la naturaleza de nuestra investigación los métodos más apropiados para la misma será el método analítico, inductivo, deductivo, científico y experimental.

7.2 TECNICAS DE RECOLECCION DE INFORMACIÒN

La recolección de información se basará en las técnicas descritas en la investigación científica entre las cuales podemos enumerar fuentes de información, la entrevista y la observación.

8. DISEÑO DE LA INVESTIGACION

El diseño de la investigación se basara en el alcancé de los objetivos planteados por el grupo investigador, por cuanto el tema de tesis con lleva el diseño y la construcción de una maquinaria tecnológica basada en ciencias exactas como es la matemática, la física, la mecánica, la electrónica y la Informática.

9. RECURSOS

9.1 HUMANOS

- Director de Tesis.

  Ing. Marco Polo Silva.

- Asesora Interna

  Ing. Enma Campaña.

- Investigadoras.

  Alexandra Herrera.

  Mariela Uribe.

DESCRIPCIÒN	CANTIDAD	N. HORAS	PREC/HORA	TOTAL
Director de Tesis	1	50	5	250,00
Asesora de Tesis	1	40	5	200,00
Egd. Sistemas	2	200	4	800,00
TOTAL		290		**$1250,00**

9.2 TECNICOS

DESCRIPCION	CANTIDAD	PREC/HORA	TOTAL
Internet	100	1,00	100,00
Uso de computadora	50	0,20	10,00
Partes del Robot			1000,00
TOTAL			**$ 1110,00**

9.3 MATERIALES

DESCRIPCION	CANTIDAD	PREC/HORA	TOTAL
Esferos	2	0,50	1,00
Lápices	2	0,35	0,70
Hojas de Papel Boon	1000	0,02	20,00
Disquete	3	0,50	1,50
CD'S RW	5	1,20	6,00
Cartuchos De Impresión (Recargada)	4	5,00	5,00
Copias	2000	0.03	60,00
Encuadernación	4	1,00	4,00
Movilización			200,00
Imprevistos			20,00
TOTAL			**318,20**

9.4 PRESUPUESTO

DESCRIPCION	TOTALES
Recursos Humanos	1250,00
Recursos Técnicos	1110,00
Materiales	318,20
SUBTOTAL DE PRESUPUESTO	**2678,20**
2% INLACIÒN	53,564
TOTAL	**2731,764**

NOTA: Los gastos de la investigación y desarrollo del proyecto serán cubiertos por los investigadores. Las piezas serán exportadas las mismas que facilitaran el desarrollo del proyecto.

10. CRONOGRAMA

Ver la Figura.

11. BIBLIOGRAFIA

11.1 BÀSICA

- LÓPEZ PELÁEZ, A. (2001), "Robótica", en Enciclopedia Universal Espasa Calpe, Apéndice 1999-2000, Madrid, Espasa Calpe, 40 Págs.
- LÓPEZ PELÁEZ, A. y KRux, M. (2000), "Social Impacts of Robotics and Advanced Automation towards the Year 2010", The IPTS Report, (edited by The Institute for Prospective Technological Studies, European Commission), n° 48, Pag. 34-40.
- TEZANOS, J.F., DÍAZ, J.A., SÁNCHEZ MORALES, M.R. y LÓPEZ, A. (1997): Tendencias científico-tecnológicas en España. Estudio Delphi 1997, Madrid, Sistema.
- WARMBOLD, J. (2000): "Robot palletizes vitamins", in (IFR) (2000): World Robotics 2000, New York/Geneva, United Nations, Pàgs. 273-275.
- WEICHBRODT, B. (2000): "Industrial Development in Robotics", in IFR (2000): World Robotics 2000, New York/Geneva, United Nations, Pàgs. IX-X.
- http://www.noticiasdot.com/publicaciones/2004/0304/1203/noticias120304/noticias120304-2.htm.
- http://www.wowwee.com/robosapien/manual.html.
- http://www.wowwee.com/robosapien/robo1/robomain.html
- http://www.robotstore.com/robosapien.asp?afid=4-122.

11.2 CONSULTADA

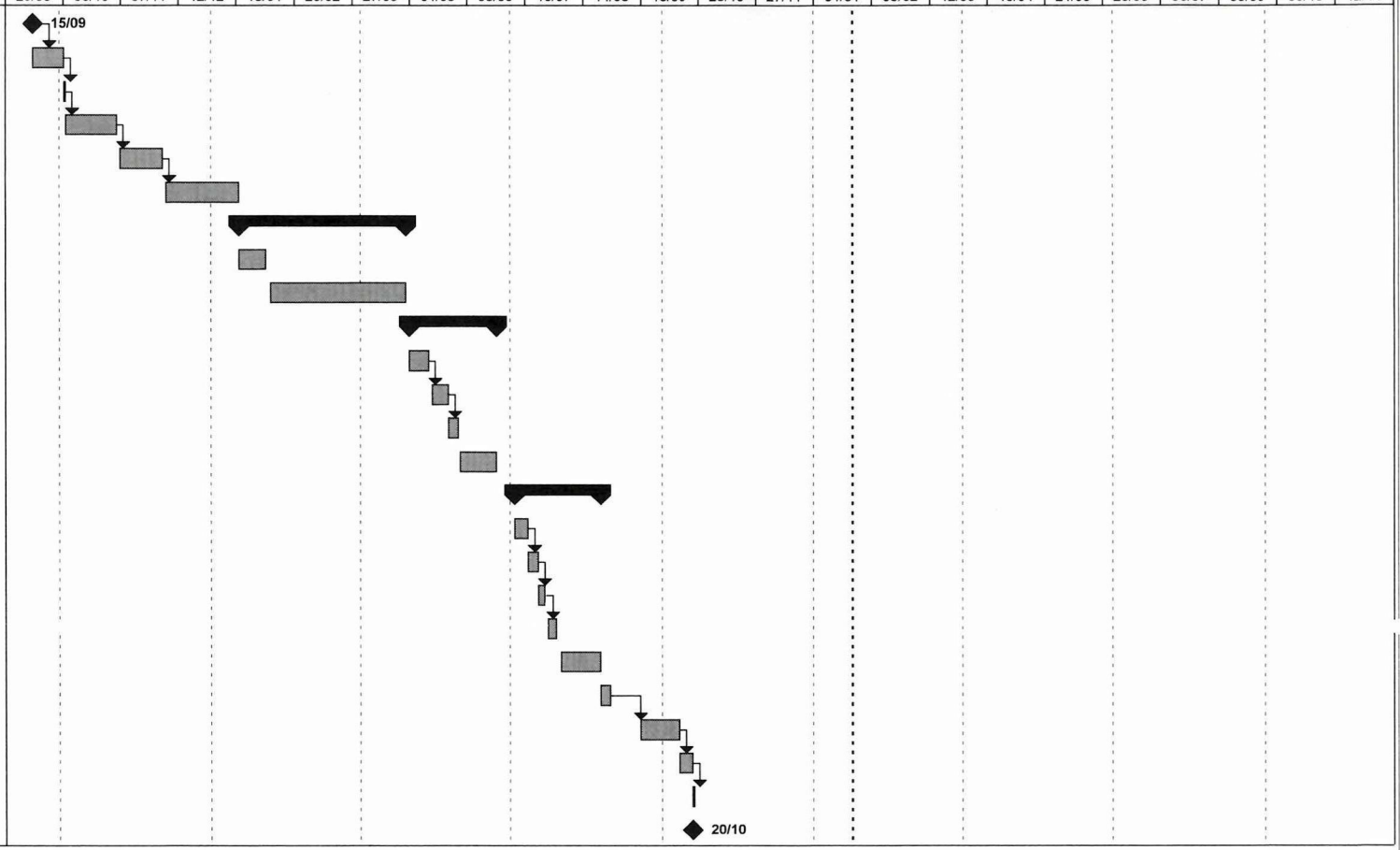
- OLLERO, Aníbal (2001); Robótica Manipuladores y robot móviles, Alfomega, Primera Edición, Madrid-España, Págs. 1-37.
- URQUIZO, Patricio (2001); Como Realizar la tesis o una investigación, Graficas Riobamba, Primera Edición, Quito- Ecuador, Pág. 1-124.
- AKEEL, H.A., RUTLEDGE, G.J. (2000): "Technological Enhancements and Their Effect on Price/Performance Indicators of Industrial Robots", in (IFR) (2000): World Robotics 2000, New York/Geneva, United Nations, pp. XIV XX.
- KINGHT, A.L. (1989): "Robots y maquinaria de producción automática", en OIT (1989): Enciclopedia de salud y seguridad laboral en el trabajo, Madrid, Ministerio de Trabajo y Seguridad Social, Pág. 2143-2147.
- LÓPEZ PELÁEZ, A. (1996): "El trabajo robotizado: perspectivas sobre la producción industrial en la sociedad tecnológica emergente", en Sistema. Revista de Ciencias Sociales, n° 135, Pág. 75-104.
- LÓPEZ PELAÉZ, A. (1997): "Robótica", en Enciclopedia Universal Espasa Calpe. Apéndice 1995-1996, Madrid, Espasa Calpe, Págs. 35.
- LÓPEZ PELÁEZ, A. (1998): "Los procesos de robotización y sus impactos sociales", en Tezanos Tortajada, J.F. y Sánchez Morales, M.R. (1998): Tecnología y Sociedad en el nuevo siglo. Segundo Foro sobre Tendencias Sociales, Madrid, Sistema, Págs. 701-730.
- LÓPEZ PELÁEZ, A. (2000a): Impactos de la robótica y la automatización avanzada en el trabajo. Estudio Delphi, Madrid, Sistema.

- LÓPEZ PELÁEZ, A. (2000b): "Tendencias en Robótica y Automatización Avanzada. ¿Hacia un nuevo modelo de trabajo?", en Tezanos Tortajada, J.E (ed.) (2000): Escenarios del nuevo siglo. Cuarto Foro sobre Tendencias Sociales, Madrid, Sistema, Págs. 171-196.
- López PELÁEZ, A. (2000c): "Prospectiva, Robótica Avanzada y Salud Laboral", en Prevención, Trabajo y Salud. Revista del Instituto Nacional de Seguridad e Higiene en el Trabajo, n° 6, Págs. 14-21.
- LÓPEZ PELÁEZ, A. (2000d): "Towards a new work pattern? Trends of Automation and Robotics Systems in manufacturing and services", in Robotics, (Journal of International Federation of Robotics), n° 40, Pags. 8-10.
- www.jameco.com

11.3 CITADA

- ULLOA, Enríquez (2002); Plan Estratégico de Desarrollo 2003-2006, Latacunga-Ecuador, Págs. 7- 24 -51.
- ULLOA, Enríquez (2000); Estatuto Orgánico Sustitutivo, Latacunga –Ecuador, Págs. 10 -11.
- http://www.gio.gov.tw/info/noticia97/2000/14/p3.htm
- http://www.mtas.es/insht/revista/A_24_ST02.htm
- http://www.ugr.es/~jsenso/eunite/robots2.html
- http://www.quintadimension.com/
- http://www.monografias.com/trabajos/cibernetica/cibernetica.html.

Nombre de tarea	Duración	Comienzo	Fin
Inicio del proyecto	0 días	ju 15/09/05	ju 15/09/05
Diseño del anteproyecto	13 días	ju 15/09/05	lu 03/10/05
Presentaciòn del anteproyecto	1 día	ma 04/10/05	ma 04/10/05
Aprobaciòn del anteproyecto	23 días	mi 05/10/05	vi 04/11/05
Defensa del anteproyecto	20 días	lu 07/11/05	vi 02/12/05
Inicio de la tesis	32 días	lu 05/12/05	ma 17/01/06
Conocimientos de la Robotica	**73 días**	**mi 18/01/06**	**vi 28/04/06**
Robotica	12 días	mi 18/01/06	ju 02/02/06
Morfologia de la Robotica	60 días	lu 06/02/06	vi 28/04/06
Analisis	**39 días**	**lu 01/05/06**	**ju 22/06/06**
Modelo de Casos de Uso de la Aplicación.	10 días	lu 01/05/06	vi 12/05/06
Modelo de Objetos de la Aplicación.	8 días	lu 15/05/06	mi 24/05/06
Glosario.	4 días	ju 25/05/06	ma 30/05/06
Modelo de Casos de Uso.	16 días	ju 01/06/06	ju 22/06/06
Diseño	**38 días**	**ma 04/07/06**	**ju 24/08/06**
Modelo de Análisis y Diseño.	6 días	ma 04/07/06	ma 11/07/06
Modelo de Datos.	4 días	mi 12/07/06	lu 17/07/06
Modelo de Implementación.	4 días	ma 18/07/06	sá 22/07/06
Modelo de Despliegue.	5 días	lu 24/07/06	vi 28/07/06
Casos de Prueba.	18 días	ma 01/08/06	ju 24/08/06
Conclusiones y Recomendaciones	4 días	vi 25/08/06	mi 30/08/06
Predefensa de la Tesis	18 días	lu 18/09/06	mi 11/10/06
Correciiones de la Predefensa	6 días	ju 12/10/06	ju 19/10/06
Defensa de Tesis	1 día	vi 20/10/06	vi 20/10/06
fin	0 días	vi 20/10/06	vi 20/10/06



PROYECTO:

" DESARROLLO DE UN PROTOTIPO DE ROBOT CON CAPACIDAD DE HABLAR Y CAMINAR ASISTIDO POR UN COMPUTADOR PARA LA UNIVERSIDAD TECNICA DE COTOPAXI"