SEDE GUAYAQUIL

FACULTAD DE INGENIERÍAS

CARRERA: INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del Título de:
Ingeniero Electrónico

TEMA:

Diseño y construcción de dos robots con aplicaciones de Movimientos y Transformación sobre una misma estructura

AUTORES:

Erick Rea Escalante

Elson Guizamano Araujo

Geovanny García Flor

TUTOR:

Ing. Luis Córdova

GUAYAQUIL – ECUADOR

2014

DECLARATORIA DE RESPONSABILIDAD

Bajo juramento declaramos que los conceptos desarrollados, los procedimientos descritos y las conclusiones del presente trabajo, son exclusiva responsabilidad de los autores, y, que hemos consultado las referencias bibliográficas citadas en este documento.

Además, declaramos que la propiedad intelectual es de la Universidad Politécnica Salesiana Sede Guayaquil según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Guayaquil, Septiembre del 2014

(f)_____
ERICK REA ESCALANTE

(f)_____
ELSON GUIZAMANO ARAUJO

(f)_____
GEOVANNY GARCÍA FLOR

DEDICATORIA

Dedico el presente trabajo a mis padres Eloy Rea y Lenny Escalante, por el apoyo incondicional brindado durante todos estos años de estudios y por los consejos y enseñanzas que me han dado durante toda mi vida, ya que gracias a esos consejos y a ese apoyo incondicional me han convertido en un hombre de bien y estaré agradecido toda mi vida por ello.

A mis hermanos por ser mi referencia para seguir adelante tanto académicamente como profesionalmente.

A mis amigos, que de una u otra manera me brindaron su apoyo para no decaer y seguir adelante.

Pero en especial dedico este trabajo a mi esposa, Martine Moaerts, siendo ella la que me alentaba día a día durante el transcurso de esta dura etapa, y que a pesar de la distancia que nos separaba, me supo comprender y pudo entender que la culminación de mi carrera comprendía un sacrificio mutuo que nos mantendría alejados, pero que a la vez significaba mucho en mi vida, además, por estar siempre al lado mío incondicionalmente para decirme con tanto amor “tu puedes hacerlo”.

Te dedico este trabajo también por darme la bendición de poder ser padre, y saber que mi hijo está creciendo sano y fuerte junto a ti. Siendo eso para mí como un motor que me impulsaba día a día con tanta energía para lograr tan duro objetivo y poder estar junto a ustedes, ya que ustedes son mi razón para seguir adelante porque los extraño, así que por eso y muchas cosas más les dedico este trabajo.

Erick Rea Escalante

AGRADECIMIENTOS

A Dios por derramar sus bendiciones sobre mí y llenarme de sus fuerzas para vencer todos los obstáculos desde el principio de la vida.

A mi Padre, por ser ejemplo de arduo trabajo y tenaz lucha en la vida este logro lo quiero compartir contigo, por ser mi papá y por creer en mí, eres mi modelo a seguir.

A mi Madre, esa mujer que supo ser padre y madre, por todo el esfuerzo, sacrificio, cariño, comprensión, apoyo incondicional, la confianza en cada momento de mi vida y un montón de cosas que nuestras madres son capaces de hacer para ver a su hijo feliz, realizado en la vida. Te Amo Mamá.

A mis hermanos por estar siempre presentes, acompañándome para poderme realizarme.

A Teresa quien me apoyo y alentó para continuar, cuando parecía que me iba a rendir.

A mi familia quienes por ellos soy lo que soy.

A mis compañeros de tesis gracias totales.

“La dicha de la vida consiste en tener siempre algo que hacer, alguien a quien amar y alguna cosa que esperar”. Thomas Chalmers

Elson Guizamano Araujo

Mi mayor agradecimiento a Dios quien ha sabido guiar cada uno de mis pasos, me ha dado la fuerza, sabiduría, fe y valor para afrontar cada uno de los retos que se han presentado durante toda mi carrera profesional.

Quiero agradecer profundamente a mi madre y abuela por haberme dado todo el apoyo posible, no sólo en mis estudios universitarios, sino en cada meta y sueño que me eh propuesto cumplir, por haberme permitido tener el privilegio de seguir una carrera profesional, seguramente no podré pagar todo lo que han hecho por mí, pero trataré de hacer todo lo posible para retribuir sus esfuerzos. A ellas agradezco de todo corazón, ya que con su ejemplo han sabido inculcarme todos los valores necesarios para ser un hombre de bien.

A la Sra. Martha Núñez Espinoza por haberme ayudado en lo que más pudo y ser parte principal en el transcurso de mi carrera profesional.

Al Ingeniero Luis Córdova, Director del proyecto de titulación por sus grandiosos consejos y su valioso tiempo entregado al proyecto.

Al Ingeniero Víctor Huilcapi, Director de Carrera de Ingeniería Electrónica
Por su apoyo y tiempo entregado en la elaboración del proyecto de titulación.

Geovanny García Flor

Agradecemos también a nuestros profesores en general que compartieron sus experiencias y enseñanzas durante todos estos años de estudio, a nuestro director de tesis Ing. Luis Córdova por su asesoría técnica durante el desarrollo del trabajo físico y a nuestro director de carrera Ing. Víctor Huilcapi por su dedicación y compromiso de ayuda durante el desarrollo del libro. También agradecemos a todas esas personas que no fueron nombradas pero que de una u otra manera contribuyeron para la culminación de este trabajo de investigación, y en especial a "Talleres Cedeño" por ayudarnos en la asesoría mecánica y prestarnos sus máquinas de taller para el desarrollo del trabajo físico.

Los autores

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ABSTRACT

AÑO	ALUMNOS	DIRECTOR DE TESIS	TEMA DE TESIS
2014	• ERICK LENIN REA ESCALANTE • ELSON JONATHAN GUIZAMANO ARAUJO • GEOVANNY XAVIER GARCÍA FLOR	ING. LUIS CÓRDOVA RIVADENEIRA	**DISEÑO Y CONSTRUCCIÓN DE DOS ROBOTS CON APLICACIONES DE MOVIMIENTOS Y TRANSFORMACIÓN SOBRE UNA MISMA ESTRUCTURA**

La presente tesis: **"DISEÑO Y CONSTRUCCIÓN DE DOS ROBOTS CON APLICACIONES DE MOVIMIENTOS Y TRANSFORMACIÓN SOBRE UNA MISMA ESTRUCTURA",** se basa en el diseño y construcción de dos estructuras electromecánicas diferentes capaces de cambiar su estado original por otro secundario utilizando servomotores como elementos motrices, los mismos que estarán equipados con sensores para la adquisición de datos de variables físicas como la temperatura y la distancia. Estos robots pueden ser comandados de dos formas distintas mediante un switch incorporado en cada robot que permite seleccionar el tipo de modo de operación, siendo la primera el modo manual, que permite comandar a cada robot de forma teleoperada mediante un control remoto inalámbrico a una distancia no mayor a los 25mts. El segundo modo es el modo automático o modo demo, que le permite a cada robot dar una demostración automática de todas las funciones que puede desarrollar. Los robots tienen incorporado en su interior algoritmos de programación de alto nivel capaces de permitirles realizar todas las funciones anteriormente mencionadas.

Los beneficiarios directos, los estudiantes de los ciclos superiores de la carrera de Ingeniería en Electrónica y el club de robótica.

PALABRAS CLAVES

Diseño, construcción, elementos motrices, estructuras mecánicas, sensores, adquisición de datos, inalámbrico, comunicación serial, teleoperado, algoritmos.

ABSTRACT

YEAR	STUDENTS	THESIS DIRECTOR	THESIS TOPIC
2014	• ERICK LENIN REA ESCALANTE • ELSON JONATHAN GUIZAMANO ARAUJO • GEOVANNY XAVIER GARCÍA FLOR	ING. LUIS CÓRDOVA RIVADENEIRA	**DESIGN AND CONSTRUCTION OF TWO ROBOTS WITH THE APPLICATIONS: MOVEMENT AND TRANSFORMATION IN THE SAME STRUCTURE**

This thesis: **"DESIGN AND CONSTRUCTION OF TWO ROBOTS WITH THE APPLICATIONS: MOVEMENT AND TRANSFORMATION IN THE SAME STRUCTURE"**, is based on the design and construction of two different electromechanical structures that can be able of changing its original status in a secondary one using elements like the servomotors, the robots are also equipped with sensors for acquiring data from physical variables such as temperature and distance. These robots can be commanded in two different ways using a built-in switch that allows each robot to select the type of operating mode. The first being the manual mode, which allows using a wireless remote control to operate each robot at a distance of no more than 25mts. The second one is the automatic mode or demo mode, which allows each robot gives automatic demonstration of all functions that can be developed. The robots have incorporated inside a big level of programming in algorithms. With those algorithms, the robots are able to realize all function mentioned.

The direct beneficiaries are the students in the upper levels of Engineering in Electronics and the Robot Club.

KEY WORDS

Design, construction, motor elements, mechanical structures, sensors, data acquisition, wireless, serial communication, tele-operated, algorithms.

INTRODUCCIÓN

Dicho proyecto trata sobre el diseño y construcción de dos robots con aplicaciones de Movimientos y Transformación sobre una misma estructura, los mismos que servirán para realizar prácticas en las materias de Microcontroladores II, Robótica y Teoría del Diseño que son parte del área de Automatización y Control Industrial de la carrera de Ingeniería en Electrónica.

El objetivo principal de este proyecto es fortalecer, por medio de la práctica y de la interacción, los conocimientos teóricos adquiridos por parte de los estudiantes de los ciclos superiores (8vo, 9no,10mo) en las materias anteriormente citadas, ya que estos dos robots están listos con programas pregrabados tanto para una demostración de sus capacidades o, para el control de los mismos mediante un control remoto inalámbrico, pero, lo más importante, es que tienen el algoritmo de programación libre para realizar cualquier tipo de modificación que sea necesaria ó, para experimentar con algoritmos realizados por los estudiantes.

En el **Capítulo 1** se detallan los hechos preliminares tales como el planteamiento del problema, metodologías aplicadas, técnicas utilizadas, beneficiarios de la propuesta y el impacto del proyecto para los estudiantes.

En el **Capítulo 2** se realiza el marco teórico sobre los tópicos más importantes del proyecto, donde se detallan los elementos utilizados que forman parte del proyecto, teniendo así, un material de consulta bien detallado complementario al trabajo realizado.

En el **Capítulo 3** se encuentran todos los parámetros y procedimientos que se utilizaron y siguieron para desarrollar el proyecto, detallando todos los circuitos electrónicos utilizados, las simulaciones de los circuitos, los diseños y diagramas de las PCB, la realización de las estructuras con sus materiales complementarios y el algoritmo principal de control de cada robot y del control remoto.

CAPÍTULO 1: EL PROBLEMA

1. EL PROBLEMA

1.1 Planteamiento del Problema

La Universidad Politécnica Salesiana con sede en la ciudad de Guayaquil (UPS-Gye), con aproximadamente 15 años de presencia, es una institución de educación superior que se encuentra ubicada en las calles Chambers y Calle 5 de Junio. Teniendo como una de sus ofertas educativas las carreras de Ingenierías, siendo la carrera de Ingeniería en Electrónica con Mención en Sistemas Industriales una de la más antigua, donde dentro de sus perfiles profesionales se destaca el énfasis en automatización y control industrial, siendo ésta un área de desarrollo que exige una actualización constante de su conocimiento, tanto para los estudiantes de la carrera como para los docentes de la misma.

En el área de automatización y control industrial se encuentra la Robótica, siendo ésta una importante área en donde los estudiantes de los ciclos superiores 8vo, 9no y 10mo no han tenido la oportunidad de realizar suficiente práctica, de esta manera en la presente tesis se pretende fortalecer esta importante área para beneficio de docentes y estudiantes.

En vista de esta necesidad, se ha planteado desarrollar dos robots con aplicaciones de transformación llamados “Transformer” para satisfacer las necesidades tanto de los profesores como la de los estudiantes de ciclos superiores que cursen la materia de robótica. Así, los estudiantes, y el club de robótica, podrán contar con una herramienta sólida y didáctica para complementar el aprendizaje adquirido en clases.

1.2 Delimitación del problema

El tema de investigación se enfoca al desarrollo de dos robots con características de transformación didácticos para complementar el aprendizaje práctico de los estudiantes de Ingeniería Electrónica, los mismos que cambiarán su forma estructural inicial a otra secundaria (Transformar), es decir, pasar de un carro 4x4 no tripulado de baja velocidad a un robot del tipo "humanoide" de 22 grados de libertad, y el segundo de un carro 6x6 de baja velocidad a un robot del tipo "arácnido" de 25 grados de libertad, cuyos movimientos no serán tan veloces.

Los robots incorporarán en su interior tres tipos de sensores para interactuar con variables físicas, las cuales son temperatura, distancia y luminosidad.

El control del robot se lo realizará de dos formas, la primera por medio de un control remoto inalámbrico, cuyo mando será por medio de una pantalla táctil monocromática de 64x128 mpx (Mega Pixeles), el mismo que cubrirá una distancia no mayor de los 25 metros en línea de vista, y la segunda de forma autónoma, es decir con rutinas de movimientos pre grabadas suficiente para controlar los movimientos del mismo.

Constará además de una memoria de voz, con voces pregrabadas de aproximadamente 8 segundos de duración, que servirá para indicar comandos en ejecución como inicio y final de la transformación.

Los robots tendrán como fuente de energía a dos baterías tipo Ni-mh de 6V y una batería alcalina de 9V, por lo tanto no necesitan de ningún tipo de fuente de poder externa, evitando el uso de cables que se encuentren fuera de la estructura de cada robot.

La electrónica del robot (robot + control remoto) se la desarrollará con elementos SMD (Dispositivos de Montaje Superficial) en la mayor parte posible por cuestiones de diseño.

Para la estructura del robot se utilizarán materiales livianos pero resistentes como el aluminio de 1.5mm y el acrílico de 3mm y 6mm, y se utilizarán servomotores de fuerza para su “transformación” y movimiento.

Se pretende diseñar los robots con las dimensiones aproximadas de 20cm de alto y 15cm de ancho.

Estos robots servirán como material didáctico de los profesores de la UPS - Gye para las materias Microcontroladores II, Robótica y Teoría del Diseño.

1.3 Objetivos

1.3.1 Objetivo General

- Diseñar y construir dos robots con características estructurales de transformación sobre una misma plataforma para realizar prácticas y fortalecer el proceso de enseñanza - aprendizaje en el área de robótica de los estudiantes de la carrera de Ingeniería en Electrónica de la UPS-Gye.

1.3.2 Objetivos Específicos

- Diseñar y construir dos tarjetas controladoras a base de microcontroladores PIC capaces de comunicarse con módulos inalámbricos Xbee y una pantalla grafica GLCD con panel táctil para cumplir la función de control remoto para cada robot.

- Desarrollar dos tarjetas principales o interfaces de control a base de Microcontrolador PIC capaces de controlar 22 y 25 servomotores respectivamente, y que permitan la conexión con dispositivos externos como las memorias de voz, las luces led, y que además se puedan comunicar vía serial con el modulo de transmisión y recepción inalámbrica Xbee de cada robot.

- Diseñar y construir dos tarjetas para control a base de microcontroladores PIC encargadas de gestionar la lectura de datos de los sensores externos y

que posea salida serial para comunicación entre microcontroladores PIC de cada tarjeta.

- Desarrollar dos estructuras dinámicas a base de aluminio y acrílico capaces de poder adquirir dos formas físicas diferentes para cada tipo de robot (carro 4x4 – humanoide y carro 6x6 - arácnido) en una misma estructura.

- Diseñar y construir dos tarjetas capaces de gravar voz de un humano y reproducirla dentro de los comandos de ejecución de cada robot.

- Desarrollar algoritmos de programación que le permitan al robot realizar las aplicaciones de transformación de modo 4x4 a humanoide y caminar en ese modo, de humanoide a 4x4 y realizar las tareas de movimiento delantero, movimiento trasero, giros de 90 grados hacia la derecha e izquierda.

- Desarrollar algoritmos de programación que le permitan al robot realizar las aplicaciones de transformación de modo 6x6 a hexápodo y caminar en ese modo, de hexápodo a 6x6 y realizar las tareas de movimiento delantero, movimiento trasero, giros de 90 grados hacia la derecha e izquierda.

1.4 Justificación

En el área de Robótica, existen pocas herramientas didácticas que ayuden a complementar y relacionar el aprendizaje teórico adquirido en las aulas de clases por parte de los estudiantes de los ciclos superiores con la práctica a nivel profesional, es por ésta razón que se diseñará una herramienta muy didáctica que le ayudará a los estudiantes a realizar sus prácticas y poner a prueba sus conocimientos adquiridos en el área de Robótica.

Se decidió por la realización de este proyecto, debido a la necesidad que se notaba por parte de los compañeros de la carrera, que querían ver en un robot práctico todos sus conocimientos teóricos adquiridos durante sus años de estudio en las

materias de electrónica, tales como, microcontroladores II, teoría del diseño y robótica, en un robot con tal dificultad.

En la materia de *Microcontroladores II*, se pueden utilizar estos robots para diseñar nuevas rutinas de programación, realizar prácticas con movimientos angulares para servomotores, poner a pruebas módulos de comunicación inalámbrica, sensores de temperatura, distancia, luminosidad e incluso poder adaptar otros nuevos.

En la materia *Teoría del Diseño*, se pueden utilizar estos robots como referencias para el diseño de PCB electrónicos de doble cara y todo lo que implica su desarrollo, como distancias entre líneas de datos, ubicación de fuentes de poder, ubicación de la electrónica de control, e incluso, para referencia entre acoples de partes mecánicas con partes electrónicas.

En la materia de *Robótica*, se pueden utilizar estos robots como ejemplos de robots de categoría "transformer", considerando que la UPS-Gye carece de robots de este tipo, además, se los puede utilizar para darles nuevas aplicaciones didácticas o probar nuevas rutinas de movimientos para la creación de nuevos robots en esta categoría.

Adicional a las aplicaciones didácticas que estos robots tienen, se los puede considerar para concursos de robótica, como es el famoso CER (Concurso Ecuatoriano de Robótica), donde los miembros actuales del club de robótica de la UPS-Gye ya podrán contar con dos nuevos robots y participar en la categoría "Robot transformer" o "Categoría Libre" de dicho concurso.

1.5 Hipótesis

Será posible la construcción de una estructura dinámica capaz de poder optar dos formas diferentes manteniendo la motricidad en cada una de sus formas y poder ser manipuladas por el mismo controlador de manera inalámbrica.

1.6 Variables e Indicadores

Variable dependiente. – Desde la propuesta

- Diseñar un robot de categoría “Transformer” capaz de cambiar su aspecto físico inicial por uno secundario, sin perder su motricidad para la materia de Robótica.

Variable Independiente. – Desde el problema

- Complementar y Fortalecer del aprendizaje de los estudiantes de los ciclos superiores de la carrera de Ingeniería en Electrónica con mención en Sistemas Industriales de la UPS-Gye en las materias de Microcontroladores II, Teoría del Diseño y Robótica.

1.7 Metodología

1.7.1 Métodos

Para el desarrollo de la investigación se utilizarán los siguientes métodos:

1.7.1.1 Método Experimental

Se utilizará el método experimental para desarrollar diseños preliminares de varias tarjetas para el control principal de los servomotores y del control remoto hasta determinar el mejor diseño que nos permita controlar todos los elementos electrónicos que conforman a los robots.

También se utilizará éste método experimental en el desarrollo de las estructuras de cada robot, hasta llegar a determinar a la mejor estructura que cumpla con los requisitos necesarios para satisfacer el objetivo final que era el de desarrollar una estructura que le permita al robot transformarse.

1.7.2 Técnicas

Las técnicas que se aplicarán durante la investigación son las siguientes:

1.7.2.1 Técnica Documental

Para la recolección de información necesaria como hojas de datos y teorías para estudiar el funcionamiento de cada elemento que forma parte de nuestro sistema

final o robot y luego verificar su funcionamiento en la práctica. Además, el desarrollo del marco teórico fue en base a ésta técnica mencionada.

1.7.2.2 Técnica Observación

Para la visualización de hechos, fenómenos, objetos, acciones que ayuden al desarrollo de la investigación.

1.7.2.3 Técnica Experimental

Para la comprobación de los nuevos conocimientos adquiridos con la técnica de documentación.

1.7.2.4Técnica de Test

Para la verificación de los resultados de la experimentación realizada.

1.7.2.5 Técnica de Diseño

Para la implementación de los conocimientos adquiridos durante todo el proceso de investigación en el desarrollo del proyecto.

1.7.3 Instrumentos de información y recolección de datos

Para el proyecto se utilizarán varios elementos que colaboraron con el desarrollo de la investigación.

Los instrumentos involucrados a la recolección de datos estadísticos fueron:

- Encuestas (Véase ANEXO 1).

Los instrumentos involucrados en la recolección de datos técnicos fueron:

- Hojas de datos de cada elemento utilizado.
- Análisis de contenido documental físico y virtual.

1.8 Población y Muestra

Para efectos de la relevancia de la investigación, se realizó un censo a estudiantes de octavo a décimo semestre de la carrera de Ingeniería en Electrónica, teniendo los siguientes resultados:

Descripción	Cantidad
Estudiantes de la materia MICROCONTROLADORES II	25
Estudiantes de la materia TEORÍA DEL DISEÑO	15
Estudiantes de la materia ROBÓTICA	17
Club de ROBÓTICA	10
TOTAL	**67**

Tabla 1: Población de estudiantes de la carrera de Ingeniería Electrónica

Fuente: Los autores

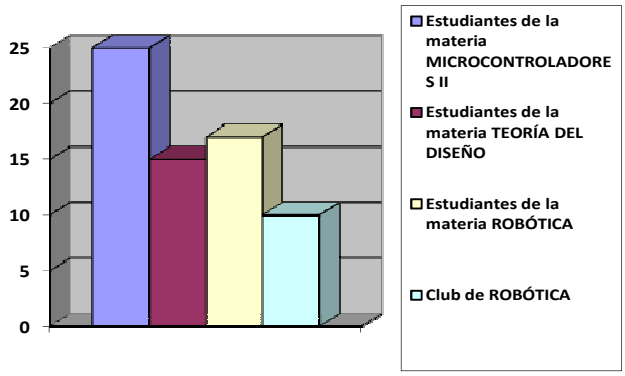


Figura 1.1: Gráfico estadístico de la población específica de la investigación

Fuente: Los autores

1.9 Descripción de la propuesta

Los robots transformer serán controlados mediante Microcontroladores PIC de la familia 16F. Básicamente cada robot constará de dos módulos conformados por un PIC y sus dispositivos externos, el primer módulo será para la comunicación entre la pantalla táctil y el módulo de transmisión y recepción inalámbrica (control remoto), y el segundo módulo será para la ejecución de los comandos que se recepten del primer módulo, logrando así poder controlar los servos para sus movimientos, la memoria de voz para los comandos de voz y los servomotores de fuerza para su transformación.

El control remoto para cada uno de los robots estará conformado por un microcontrolador PIC de la familia 16F, además constará de un transmisor bidireccional de largo alcance llamado Xbee o módulo de transmisión y recepción inalámbrica.

El Microcontrolador PIC del control remoto servirá para manejar la pantalla gráfica táctil monocromática, la misma que nos indicará los estados del robot mediante mensajes como “inicialización”, “modo de operación”, entre otros, además, tendrá en cada menú y submenú cuadros de comando que al pulsarlos enviarán la información necesaria hacia el robot para su ejecución.
La comunicación entre el Microcontrolador PIC de la tarjeta de control principal y el de la tarjeta de comunicación de los sensores externos será de forma serial. La comunicación entre el microcontrolador PIC de la tarjeta principal con el módulo de transmisión y recepción inalámbrica Xbee también será de forma serial, recibiendo éste último las instrucciones de ejecución desde el control remoto, y, una vez recibido los comandos, transmitirá las confirmaciones o mensajes que envíe el robot hacia el control.

Este módulo de la tarjeta de control principal también constará de una memoria de voz para la ejecución de los comandos de voz pregrabados.

Este módulo principal tendrá la función de controlar a todos los servomotores para realizar las rutinas de movimientos indicados en el control remoto, ya sea en la

etapa inicial (como un carro 4x4 o un carro 6x6), o en la etapa final (transformación hacia un humanoide o hexápodo).

El estado inicial del primer robot será la de un carro 4x4 (debido a que posee tracción en las cuatro ruedas) de poca velocidad, y cada rueda estará montada sobre un servo motor.

En el estado inicial del primer robot podrá realizar los siguientes movimientos: aceleración en línea recta hacia adelante, aceleración en lineal recta hacia atrás, giros de 90 grados hacia la derecha y hacia la izquierda.

Al iniciar la “transformación”, nuestro robot pasara de ser un carro 4x4 a ser un robot “humanoide” con 22 grados de libertad, ya que tendrá movimientos en los hombros (1 grados por cada hombro, son dos en total), brazos (1 grados por cada brazo, son dos brazos en total), codos (1 grados por cada codo, son dos codos en total), piernas (4 grados por cada pierna, son dos piernas en total), base para ruedas superiores (1 grados por cada base, son dos bases en total), cuello (1 grado), cabeza (1 grado) y ruedas (1 grado por cada rueda, 4 ruedas en total).

Además, el robot podrá realizar la aplicación de caminar hacia adelante en forma de “Humanoide”.

El segundo robot tendrá como estado inicial la forma de un carro 6x6 (ya que tiene tracción en las seis ruedas) de baja velocidad.

En el estado inicial, el segundo robot podrá realizar los siguientes movimientos: aceleración en línea recta hacia adelante, aceleración lineal recta hacia atrás, giros de 90 grados hacia la derecha y hacia la izquierda.

Al iniciar la “transformación”, el segundo robot pasará de ser un carro 6x6 a ser un robot del tipo “Hexápodo” con 25 grados de libertad, ya que tendrá movimientos en las seis extremidades que lo conforman (3 grados de libertas por cada extremidad), en su “arma de ataque” (porque posee la forma de un tanque de guerra) tendrá simplemente un grado de libertad y en sus 6 ruedas (1 grado de libertad por cada rueda).

Además podrá realizar la función de caminar hacia adelante en forma de "Hexápodo".

Para culminar, cada robot tendrá incorporada una memoria de voz, la cual le permitirá "hablar" frases o palabras pregrabadas cuando inicia y termina la transformación correspondiente.

1.10 Beneficiarios

Los estudiantes de los ciclos 8vo, 9no y 10mo de la carrera de Ingeniería en Electrónica de la UPS-Gye y el club de robótica.

1.11 Impacto

Los robots construidos ayudarán a mejorar y aumentar el interés de los estudiantes de la carrera de Ingeniería en Electrónica hacia la robótica, ya que de una forma práctica y didáctica podrán palpar lo que ellos son capaces de desarrollar. Con este proyecto se pretende también aumentar más el interés hacia la elaboración de este tipo de robots para desarrollar aplicaciones que solucionen problemas de la vida cotidiana.

CAPÍTULO 2: MARCO TEÓRICO

2. MARCO TEÓRICO

2.1 Antecedentes

En la actualidad la enseñanza teórica no es suficiente para los estudiantes de ingeniería electrónica en las materias de microcontroladores I-II y robótica, ya que buscan complementar de una u otra manera lo aprendido en clases con la práctica, logrando así experimentar de forma real lo aprendido durante sus años de estudios.

Existen varias formas de hacerlo, pero la ventaja en la robótica, es que encierra todos los conocimientos de electrónica en general, desde el uso adecuado de las resistencias, hasta la programación avanzada de algoritmos de control para desarrollar un sistema o robot estable.

En investigaciones anteriores no se desarrollaron robots que ayuden a facilitar el aprendizaje a estudiantes de la carrera de ingeniería en electrónica.

2.2 Introducción a la robótica

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de maquinas capaces de desempeñar tareas realizadas por el ser humano o que requieren del uso de inteligencia. Las ciencias y tecnologías de las que deriva podrían ser: los autómatas programables, las maquinas de estados, la mecánica o la informática.

La historia de la robótica ha estado unida a la construcción de “artefactos”, que trataban de materializar el deseo humano de crear seres semejantes a nosotros, que nos descargue el trabajo.

El ingeniero Español Leonardo Torres Quevedo acuno el término “automática” en relación con la teoría de la automatización de tareas tradicionalmente asociadas a los humanos.

2.3 Robot

Un robot, es un agente artificial mecánico o virtual. Es una maquina usada para realizar un trabajo automáticamente y que es controlada por una computadora.

Si bien la palabra robot puede utilizarse para agentes físicos y agentes virtuales de software, estos últimos son llamados “bots” para diferenciarlos de los otros.

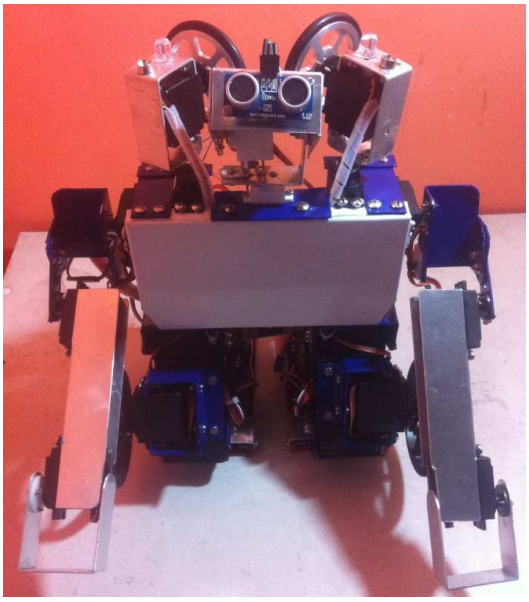


Figura 2.1: Robot Transformer UPS

Fuente: Los autores

Además un robot está conformado por piezas motrices que lo ayudan a desplazarse según sea el tipo del robot. Estas piezas motrices generalmente son: motores DC, motores AC, servomotores, entre otros.

2.4 Tipos de robots

A grandes rasgos se puede hablar de los siguientes: [1]

- Androides
- Móviles

[1] Tipos de robots. (s.f.). tiposde.org. http://www.tiposde.org/general/460-tipos-de-robots/

- Industriales
- Médicos
- Tele-operados

2.4.1 Androides

Según el documento Tipos de robots. (s.f.). tiposde.org. http://www.tiposde.org/general/460-tipos-de-robots/, nos dice “estos artilugios se parecen y actúan como si fueran seres humanos. Este tipo de robots no existen en la realidad, por lo menos por el momento, sino que son elementos ficcionales”.

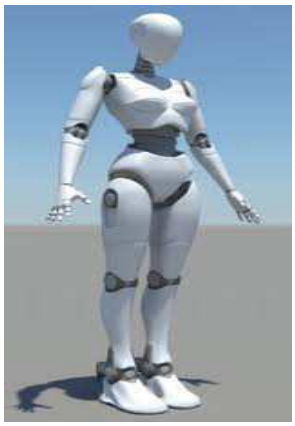


Figura 2.2: Robot Androide

Fuente: http://www.tiposde.org/general/460-tipos-de-robots/

2.4.2 Móviles

Según el documento Tipos de robots. (s.f.). tiposde.org. http://www.tiposde.org/general/460-tipos-de-robots/, nos dice “Estos robots cuentan con orugas, ruedas o patas que les permiten desplazarse de acuerdo a la programación a la que fueron sometidos. Estos robots cuentan con sistemas de sensores, que son los que captan la información que dichos robots elaboran. […]”.

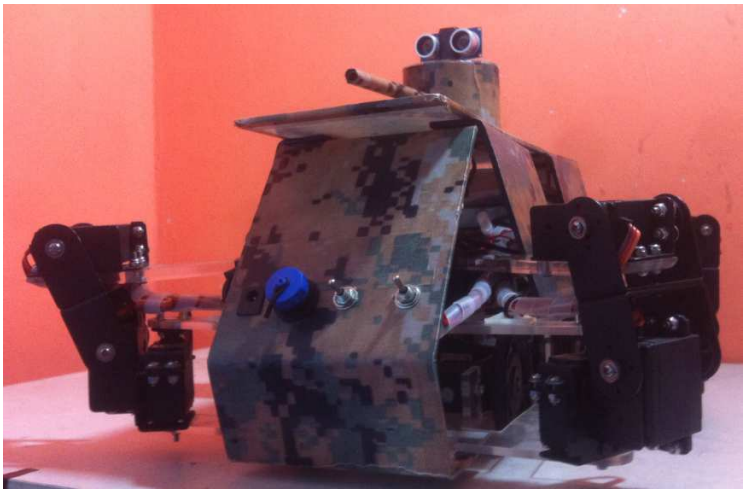


Figura 2.3: Robot Móviles

Fuente: Los autores

2.4.3 Industriales

Según Antonio, B, Luis, P, Carlos, B, Rafael. (1997). Fundamentos de Robótica. Madrid: McGraw-Hill, nos dice que "La definición más comúnmente aceptada posiblemente sea la de la *Asociación de Industrias Robóticas* (RIA)" según la cual dice "Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas".

Figura 2.4: Robot Industrial

Fuente: http://www.tiposde.org/general/460-tipos-de-robots/

2.4.4 Médicos

De acuerdo a, Tipos de robots. (s.f.). tiposde.org. http://www.tiposde.org/general/460-tipos-de-robots/, nos dice “Bajo esta categoría se incluyen básicamente las prótesis para disminuidos físicos. Estas cuentan con sistemas de mando y se adaptan fácilmente al cuerpo. Estos robots lo que hacen es suplantar a aquellos órganos o extremidades, […]”.

Figura 2.5: Robot Médicos

Fuente: http://www.tiposde.org/general/460-tipos-de-robots/

2.4.5 Teleoperados:

Según Antonio, B, Luis, P, Carlos, B, Rafael. (1997). Fundamentos de Robótica. Madrid: McGraw-Hill, nos dice que “Los robots teleoperados son definidos por la NASA como dispositivos robóticos con brazos manipuladores y sensores y cierto grado de movilidad, controlados remotamente por un operador humano de manera directa o a través de un ordenador”.

Figura 2.6: Robot Teleoperados

Fuente: http://www.tiposde.org/general/460-tipos-de-robots/

2.5 Características Generales de los robots

Según el documento, La Robótica. (Noviembre 15, 2010). http://robotcamixo.blogspot.com/2010/11/caracteristicas-generales-de-los-robots.html, nos dice "Los robots pueden ser de diferentes diseños al igual que programas, todo depende de la función que vayan a realizar.[…]" podemos decir que las características generales de los robots se las puede conocer, de las cuales se pueden nombrar las siguientes:

- Precisión de movimiento.
- Grado de libertad a la que fue diseñado y elaborado.
- Capacidad de carga según sea el tipo del robot.
- Tipo de movimiento según sus coordenadas (X-Y-Z)
- El nivel de programación y tipo de autonomía que posee.

2.6 Ventajas y desventajas de la robótica

2.6.1 Ventajas [2]

- Las ventajas de la sustitución humana por el robot son innatamente, inmensas e infinitas. Ya que sin la ayuda de esta el ser humano no hubiera sido capaz de evolucionar hasta el punto que hemos llegado ahora.
- Gracias a la robótica el ser humano ha podido dedicar su tiempo a mejora la calidad de vida al aplicarla constantemente y sustituyéndose a sí mismo en labores repetitivas y agotadoras.

[2] Ventajas y Desventajas de la Robótica. (s.f.). http://robotica-ares.blogspot.com/

- Estos robots permiten a los investigadores a entender algunas funciones imposibles de desentrañar directamente atreves de la experimentación animal.

- Son manipuladores multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas.

- Se utiliza cualquier dispositivo mecánico capaz de reproducir los movimientos humanos para la manipulación de objetos.

- Los programas educacionales utilizan la simulación de control de robots como un medio de enseñanza.

- Podemos mencionar a los multi robots como aquellos capaces de adaptarse a diversos campos de trabajo.

2.6.2 Desventajas [2]

- El robot esta específicamente diseñado para sustituir la labor humana y de esta manera mejorarla o simplemente igualar su capacidad. Aunque si es muy cierto que la robótica puede crear más empleos, también puede quitarlos.

- Actualmente se puede observar que el desempleo por la sustitución humana por los robots ha sido mayor que el empleo que esta ha creado. Esto se debe mayormente a que muchas generaciones pasadas no pudieron competir ante el potencial de los robot, por esto mimo fueron sustituidos y expulsados de esta área de trabajo especifico.

- Estas computadoras pensantes solo se consideran como una base de una especie de robots inteligentes capaces de crear copias de ellos mismo.

[2] Ventajas y Desventajas de la Robótica. (s.f.). http://robotica-ares.blogspot.com/

- El costo de un robot permanece constante con baja reducción.
- Requieren gran capital al instalarle que se deprecia con los anos y el recurso humano que cuesta el tiempo trabajando.
- Un robot se puede justificar económicamente.
- El recurso humano puede sentirse amenazado por el desempleo.

2.7 El microcontrolador PIC

Los microcontroladores fabricados por la empresa Microchip Technology Inc., son comúnmente llamados PIC- *Peripheral Interface Controller* (controlador de interfaz periférico), aunque su nombre verdadero es PICmicro.

La empresa Microchip actualmente es una de las empresas con mayor venta de microcontroladores a nivel mundial, debido a las ventajas que sus dispositivos tienen en comparación a sus competencias, por ejemplo, son más económicos, poseen una gran variedad de modelos por cada familia. Existen en la actualidad una gran variedad de software para programarlos con distintos lenguajes de programación, considerando que los microcontroladores solamente reconocen el lenguaje máquina, el mismo que está compuesto de ceros (0`s) y unos (1`s). Se los debe programar en lenguaje humano, es decir, un lenguaje que sea comprensivo lógicamente para las personas, y que mediante otro software se lo pueda convertir a lenguaje máquina para que el microcontrolador pueda realizar las tareas asignadas.

Dentro de los lenguajes de programación se puede señalar desde el más básico como el lenguaje en Basic, que es de alto nivel ya que es similar al lenguaje humano, hasta el más complejo que se programa en código binario, pero de bajo nivel.

[2] Ventajas y Desventajas de la Robótica. (s.f.). http://robotica-ares.blogspot.com/

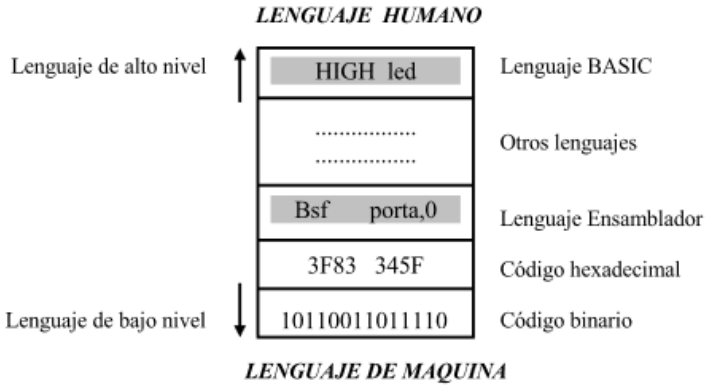


Figura 2.7: Niveles de lenguaje de programación

Fuente: REYES, Carlos, Microcontroladores *PIC Programación en Basic*,3re edición,rispergraf,Quito,2008,Volumen 1

Además de eso, los microcontroladores PIC gracias al potencial de su empresa desarrolladora que es reconocida a nivel mundial y a las ventajas que éstos ofrecen, hoy en día son los microcontroladores más comunes que se utilizan para el desarrollo de aplicaciones de robótica en las deferentes Universidades del país.

Los microcontroladores PIC están constituidos en su interior por tres unidades funcionales de una computadora, las cuales son: unidad central de procesamiento (CPU), memoria y periféricos de entrada y salida.

2.7.1 Microcontrolador PIC16F877A

Uno de los microcontroladores PIC más utilizados en proyectos que demanden el uso de varios periféricos ya sean de entrada o de salida, es el PIC16F877A, ya que posee todo lo necesario en hardware para poder expandir cualquier tipo de proyecto que amerite utilizarlo.

2.7.1.1 Características del Microcontrolador PIC16F877A

El microcontrolador PIC16F877A posee las siguientes características en su CPU:

- Memoria de programa tipo FLASH de 8192 palabras o “Words”
- Memoria de datos tipo EEPROM de 256 bytes

- Memoria RAM de 368 bytes
- Procesador de arquitectura RISC de solo 35 instrucciones
- Velocidad de operación máxima de 20Mhz
- Soporta hasta 14 fuentes de interrupción
- Posee opción de protección de código programado (encriptado)
- Utiliza la función WDT -*Watchdog Timer* (Perro Guardián) para lograr un modo de operación más seguro

Además posee un total de 40 pines en encapsulado tipo DIP divididos en:

- 33 pines configurables divididos en 5 puertos llamados A-B-C-D-E
- 4 pines de alimentación (2 para VDD y 2 para VSS o Gnd)
- 2 pines para el oscilador externo
- 1 pin de reset o MCLR

En sus periféricos posee las siguientes características:

- 8 convertidores análogo – digital A/D de 10bits que se los puede utilizar a través del puerto A y E
- Timer o contador de 8bits con pre escala de 8 bits
- Timer de 16bits
- Puertos comparadores / PWM (multiplexados a 16bits de resolución)
- Puerto SPI (Puerto serial sincrónico para conexión de periféricos esclavos)
- USART (Universal Synchronous Asynchronous Receiver Transmiter) para comunicación serial con paquetes de 8 o 9bits
- Puerto paralelo esclavo de 8 bits

En la figura 2.8 se muestra la distribución de pines de cada puerto dividida de la siguiente manera:

- El puerto A trabaja a 6 bits
- El puerto B trabaja a 8 bits
- El puerto C trabaja a 8 bits
- El puerto D trabaja a 8 bits

- El puerto E trabaja a 3 bits

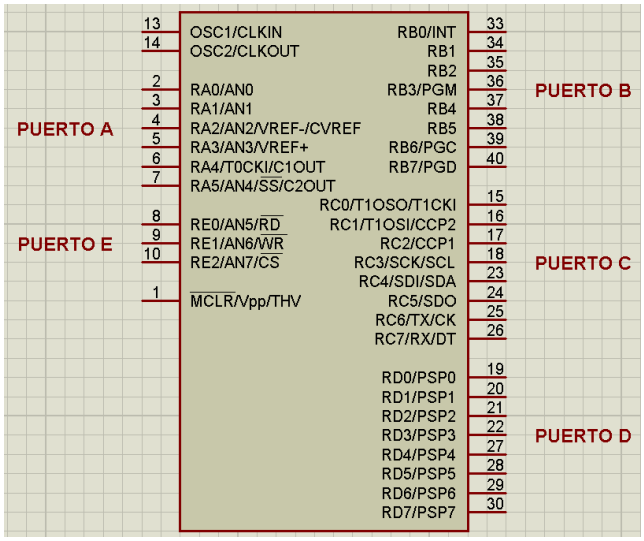


Figura 2.8: Distribución de puertos del PIC16F877A

Fuente: Los autores

Como se puede observar, en la distribución de pines del microcontrolador, se aprecia la simbología de las funciones especiales que por hardware el microcontrolador nos puede brindar, siendo éstas funciones de gran utilidad e importancia al momento de diseñar los diversos proyectos, por ejemplo el pin número 25 que corresponde al puerto C6, aparte de poder ser configurado como puerto de entrada o salida digital, proporciona además una salida de transmisión TX para comunicación con dispositivos vía hardware.

Nombre pin	Pin	Descripción
RA0/AN0	2	E/S/Digital o Entrada análoga 0.
RA1/AN1	3	E/S/Digital o Entrada análoga 1.
RA2/AN2 Vref-	4	E/S/Digital o Entrada análoga 2.
RA3/AN3/Vref+	5	E/S/Digital o Entrada análoga 3.
RA4/TOCKI	6	Bit 4 del puerto A (E/S bidireccional). También se usa como entrada de reloj al temporizador/contador TMR0. Salida de colector abierto.
RA5/AN0	7	E/S Digital o Entrada análoga 4. También se usa como el puerto serial síncrono.
RB0/INT	33	Bit 0 del puerto B (E/S bidireccional). Buffer E/S: TTL/ST. También se usa como entrada de interrupción externa (INT).
RB1	34	Bit 1 del puerto B (E/S bidireccional). Buffer E/S: TTL
RB2	35	Bit 2 del puerto B (E/S bidireccional). Buffer E/S: TTL
RB3/PGM	36	Bit 3 del puerto B (E/S bidireccional). Buffer E/S: TTL (Programación en bajo voltaje)
RB4	37	Bit 4 del puerto B (E/S bidireccional). Buffer E/S: TTL. Interrupción por cambio de pin
RB5	38	Bit 1 del puerto B (E/S bidireccional). Buffer E/S: TTL. Interrupción por cambio de pin
RB6/PGC	39	Bit 1 del puerto B (E/S bidireccional). Buffer E/S: TTL. Interrupción por cambio de pin. Entrada de reloj para programación serial.
RB7/PGD	40	Bit 1 del puerto B (E/S bidireccional). Buffer E/S: TTL. Interrupción por cambio de pin. Entrada de datos para programación serial.
RC0/T1OSO/T1CKI	15	E/S Digital. Salida del oscilador Timer 1 o entrada de reloj Timer 1
RC1/T1OSI/CCP2	16	E/S Digital. Entrada del oscilador Timer 1. Entrada captura 2; Salida compara2, Salida PWM 2
RC2/CCP1	17	E/S Digital. Entrada captura 1; Salida compara 1; Salida PWM 1
RC3/SCK/SCL	18	E/S Digital. Línea de reloj serial asíncrono en el modo SPI y el modo I2C
RC4/SDI/SDA	23	E/S Digital. Línea de datos en el modo SPI o en el modo I2C
RC5/SD0	24	E/S Digital.
RC6/TX/CK	25	E/S Digital. Transmisión asíncrona (USART) o reloj síncrono (SSP).
RC7/RX/DT	26	E/S Digital. Recepción asíncrona (USART) o línea de datos (SSP).
VDD	11,32	Voltaje de alimentación DC (+)
VSS	12,31	Referencia de voltaje (GND).
MCLR	1	Entrada de RESET al microcontrolador. Voltaje de entrada durante la programación. En nivel bajo resetea al microcontrolador.
OSC1/CLKIN	13	Entrada oscilador cristal oscilador / Entrada fuente de reloj externa
OSC2/CLKOUT	14	Salida oscilador cristal. Oscilador RC: Salida con un ¼ frecuencia OSC1
RD0/PSP0	19	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD1/PSP1	20	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD2/PSP2	21	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD3/PSP3	22	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD4/PSP4	27	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD5/PSP5	28	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD6/PSP6	29	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RD7/PSP7	30	E/S Digital. Puede ser puerto paralelo en bus de 8 bits.
RE0/RD/AN5	8	E/S Digital. Puede ser pin de lectura (read) en modo microprocesador.
RE1/WR/AN6	9	E/S Digital. Puede ser pin de lectura (write) en modo microprocesador.
RE2/CS/AN7	10	E/S Digital. Puede ser pin de selección de chip (chip select) en modo microprocesador.

Figura 2.9: Distribución de pines del microcontrolador PIC16F877A

Fuente: http://cifpn1hectorm.files.wordpress.com/2013/04/descripcic3b3n-pines1.jpg

2.7.2 Microcontrolador PIC16F877A en encapsulado SMD - QFP

Básicamente un microcontrolador PIC16F877A con encapsulado QFP – *Quad Flat Package* (Encapsulado Cuadrado Plano) en tecnología de dispositivos SMD - *Surface Mount Technology* (Tecnología de Montaje Superficial), se diferencia de

su homólogo en tecnología DIP -*through hole* (Atreves del agujero), por su número de pines y tamaño, incluyendo las ventajas que la tecnología SMD proporciona.

En la figura 2.10 se puede observar el aspecto físico de este microcontrolador PIC en tecnología QFP para dispositivos SMD y su distribución de pines.

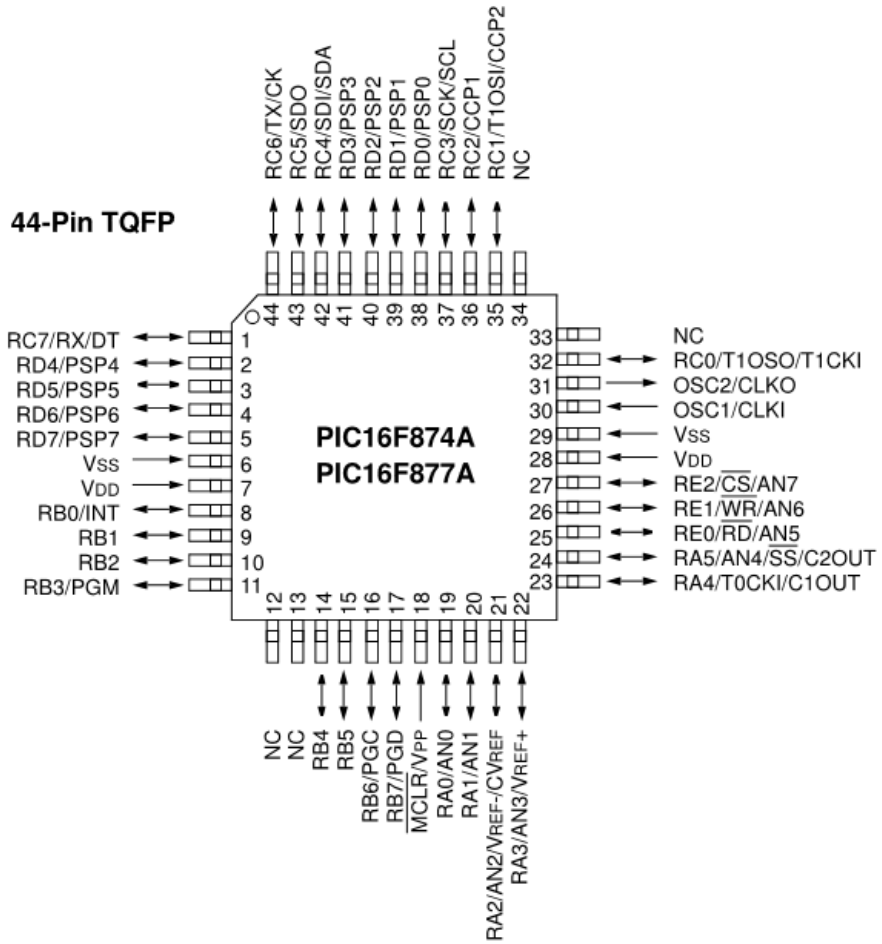


Figura 2.10: Aspecto físico y distribución de pines del PIC16F877A QFT

Fuente: Datasheet del microcontrolador PIC16F877A

2.8 Servomotores

Un servomotor, también llamado “servo”, es un dispositivo utilizado muy comúnmente en robótica, y, según sea su modelo y características eléctricas y mecánicas puede ser muy similar a un motor de corriente continua, con la ventaja que se lo puede posicionar en cualquier lugar que su rango permita y permanecer estático en dicha posición. En otras palabras podríamos decir que un servomotor

es un motor eléctrico que puede ser controlado tanto en velocidad como en posición.

2.8.1. Características a considerar para la selección de un Servo.

- Voltaje de operación
- Dimensiones
- Torque máximo
- Tipo de engranaje

2.8.2 Tipos de servos

En la actualidad existen dos diferentes tipos de servos que son los estándar o analógicos y los digitales, de los cuales se derivan un sinnúmero de modelos.

La selección del tipo de servo a utilizar dependerá del grado de precisión que se requiera en el proyecto, de su tamaño, y del tipo de control a utilizar, considerando que un servo digital es mucho más preciso que su homólogo en analógico, pero necesita de programación aunque compartan las mismas características, y lógicamente los servos digitales son más caros y consumen un poco más de corriente.

Existen servos en el mercado que son pequeños, los mismos que se los llama comúnmente como “micro servos”, pero su funcionamiento o control es similar a los de mayor tamaño sean estos analógico o digitales.

Figura 2.11: Diferentes modelos de servos analógicos y digitales

Fuente: http://www.superrobotica.com/Servosrc.htm

2.8.3 Partes de un servo

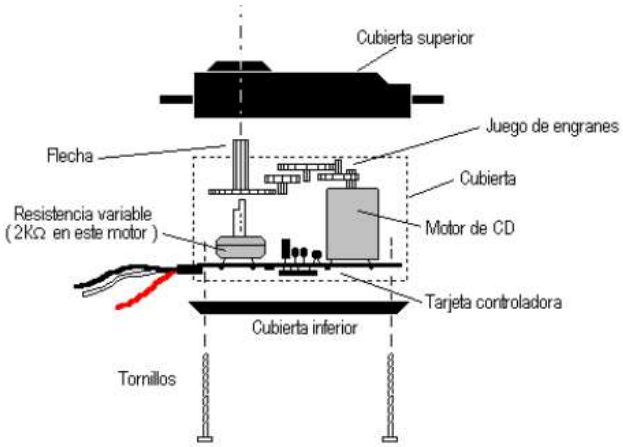


Figura 2.12: Partes de un Servo

Fuente: http://www.monografias.com/trabajos60/servo-motores/servo-motores.shtml

Los servos normalmente están compuestos en su interior por un motor de corriente continua, un sistema de engranaje o caja reductora, un sensor de posición y un circuito de control, el mismo que se encarga de controlar al servo tanto en velocidad como en posición.

Figura 2.13: Piezas que conforman a un servo

Fuente: http://ramsurobot.blogspot.be/2008/08/servomotores.html

2.8.4 Funcionamiento general de un Servo Analógico

El principio de funcionamiento de un servo analógico, se basa en el control del ancho de pulso que recibe de un tren de pulso como señal de entrada, donde ésta a su vez es transformada a voltaje por un circuito convertidor, ésta señal indicará la posición angular deseada en la que el servo se mantendrá, comprendida entre 0º y 180º, luego, la señal pasará a la lógica de control, la misma que comparará el voltaje que se obtuvo de la señal de pulso de entrada y el voltaje que envía el sensor de posición, este sensor esta conformado únicamente por un potenciómetro, que es el encargado de supervisar el ángulo actual del servo. En caso que esta comparación no sea igual, la lógica de control detecta el "error" entre los dos voltajes, y automáticamente enviará una señal de respuesta a la etapa de potencia, para que ésta a su vez accione al motor, y por medio de la caja reductora posicione al servo en la posición correcta corrigiendo así el "error" o ángulo, ésto se da cuando el voltaje obtenido por la señal de pulsos de entrada sea igual al voltaje que envía el potenciómetro, en ese instante el servo estará ubicado en la posición deseada.

En la figura 2.14 podemos observar más detalladamente el esquema en bloque del funcionamiento de un servo, donde podemos observar que el motor esta unido mecánicamente con la caja reductora, y ésta a su vez con el potenciómetro y el eje de salida, lo que significa que el mínimo movimiento del motor por medio de la caja reductora afectará la posición del eje de salida y la del potenciómetro.

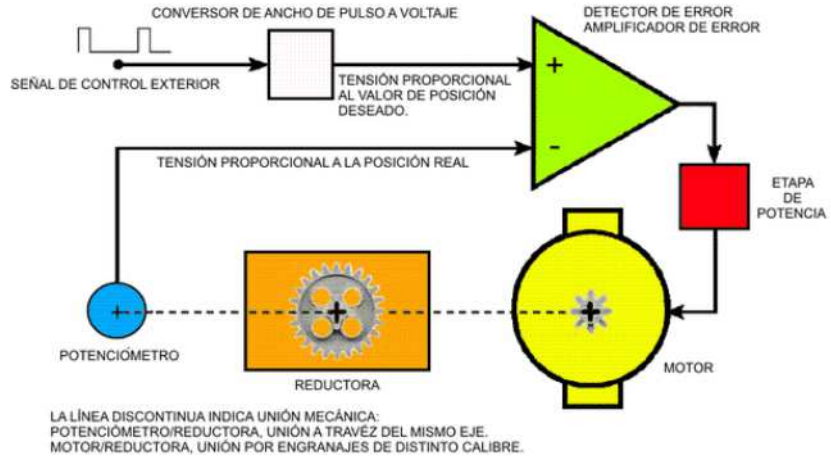


Figura 2.14: Diagrama en bloques de un servomotor

Fuente: http://www.pickey.es/uso-practico-del-pic12f675--iv-.html

2.8.5 Caja reductora

Uno de los factores más importantes del servo es su caja reductora, ya que de esta dependerá la "carga" o peso que el servo pueda soportar en su eje y también es la que dará la opción a soportar movimientos más bruscos e incluso golpes, ya que al ser metálica, el riesgo de desgaste o ruptura de los engranes es mínimo.

La caja reductora está compuesta por un juego de engranes plásticos o metálicos como muestra la figura 2.15, siendo las de engranes metálico las que ayudarán al servo a generar una mayor fuerza de tracción para mover mayores cargas, y su funcionamiento es básicamente la de disminuir la velocidad de un motor con poca fuerza y aumentarla enormemente, pero en consecuencia se reducirá su velocidad.

Figura 2.15: Caja reductora metálica y plástica

Fuente: Los autores

2.8.6 Voltaje de operación y señal de control de servos.

El voltaje de operación estándar de los servos está comprendido entre los 4.8V y los 6VDC, y su señal de control es un tren de pulsos que debe permanecer constante en el servo para que este a su vez mantenga la posición angular deseada.

El estándar de la señal controladora para todos los servos de tipo analógico, es un pulso de onda cuadrada de 1,5ms (milisegundos) que se repite cada 15ms a 20ms, es decir a una frecuencia entre 50 y 60Hz. Mientras el pulso se mantenga en ese ancho de 1.5 ms, el servo se ubicará en la posición central de su recorrido. Si el ancho de pulso disminuye, el servo se mueve de manera proporcional hacia un lado. Si el ancho de pulso aumenta, el servo gira hacia el otro lado. Generalmente el rango de giro de un servo está comprendido entre 0° y 180° para señales de trenes de pulsos entre los 0.5ms hasta los 2.5ms, o un poco más, según la marca y modelo del servo.

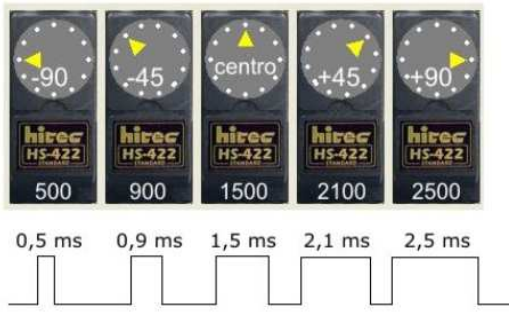


Figura 2.16: Posición angular del servo según su pulso de control

Fuente: http://www.superrobotica.com/Servosrc.htm

Ese tren de pulsos debe ser permanente, es decir que si se cancela el envío del tren de pulsos al servo, éste dejara de funcionar así se encuentren conectados sus pines de alimentación, para evitar eso, el controlador deberá mandar un tren de pulsos constantemente a una frecuencia comprendida entre 50Hz o 60Hz.

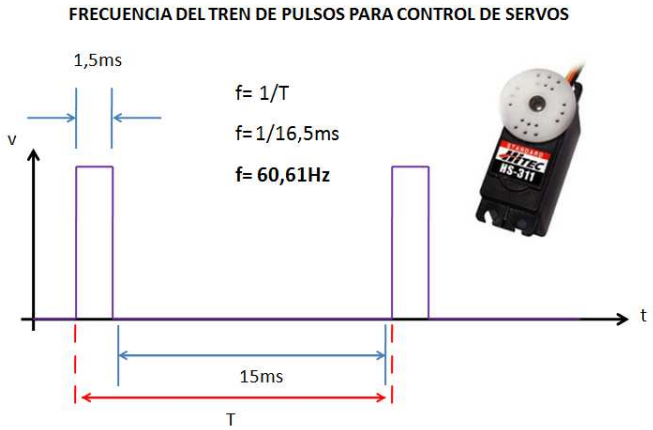


Figura 2.17: Frecuencia del tren de pulsos

Fuente: Los autores

2.8.7 Trucaje de un servo

Es posible modificar un servo para eliminar su restricción de giro y permitirle dar giros completos de 360º, a esto se lo conoce como "trucar al servo". Esto, sin embargo, convierte al servo en un motor similar a uno de corriente continua normal. Para conseguirlo es necesario modificar el circuito de control, cambiando el potenciómetro de posición por resistencias de 2.2K fijas.

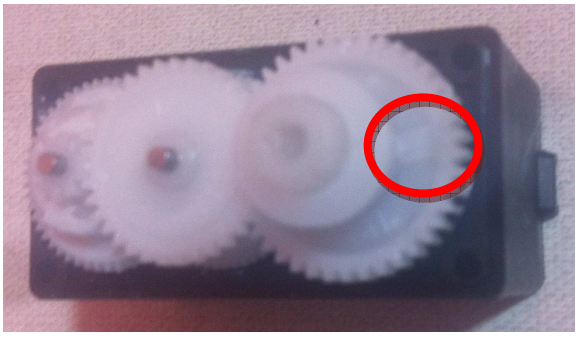


Figura 2.18: Servo Trucado

Fuente: Los autores

2.8.8 Servos HITEC

Este tipo de servos son muy comunes para prácticas de robótica, gracias a que son muy fáciles de encontrar en el mercado local, poseen una gran variedad de modelos y por las propiedades que estos brindan en general.

Los modelos HITEC más comunes en aplicaciones de robótica son:

- HS-311
- HS485HB

Figura 2.19: Servos HITEC

Fuente: Los autores

Los servos HS-311 y HS-485HB, físicamente son muy similares, pero internamente el servo HS-485HB tiene un sistema de engrane y motor más rígido proporcionándole mayor torque al servo en su eje.

Especificaciones	**HS-311**	**HS-485**
Tipo de motor	3 polos	3 polos
Velocidad (4.8V/6.0V)	0.19 / 0.15 sec @ 60 grad.	0.20 / 0.17 sec @ 60 grad.
Torque Kg/cm (4.8V/6.0V)	3.0 / 3.7	5.2 / 6.4
Medidas en milímetros	39.88 X 19.81 X 36.32	39.88 X19.81 X 37.85

Peso en gramos	42.81	45.08
Pulso requerido	3.5Vpp onda cuadrada	3.5Vpp onda cuadrada
Sistema de Control	Control de ancho de pulso	Control de ancho de pulso
Caja de engranaje	Plástica	Plástica

Tabla 2: Especificaciones de los servos HITEC
Fuente: Datasheet servomotores Hitec

2.8.9 Servos POWER HD

Este tipo de servomotores no son tan comunes en el mercado local, pero brindan gran variedad de modelos con un torque superior o similar a los HITEC, son más económicos, pero su funcionamiento es diferente con respecto al ancho de pulso que necesitan para operar. Se podría decir que estos servos operan a "lógica inversa" que los tradicionales HITEC o FUTABA, y para su mejor comprensión en la tabla 3 se detallan los valores de ancho de pulso que necesita el servo para operar, donde también se puede apreciar las diferencias entre ambos.

Posición	**HITEC**	**POWER HD**
Posición Inicial (90º)	1,5ms	1,5
Izquierda (180 º)	0,5	2,5
Derecha (0 º)	2,5ms	0,5ms

Tabla 3: Comparación de ancho de pulso para posición angular
Fuente: Los autores

Como se puede observar, el único valor en que son similares, es el de 1,5ms que corresponde a la posición inicial o neutral de 90º, luego, para los otros dos movimientos, tanto para la derecha como para la izquierda los valores son

contrarios. Esto es muy necesario de tomar en cuenta al momento de programar y ubicar el servo, ya que se podría tener movimientos en direcciones contrarias.

En la figura 2.20 se puede apreciar el comportamiento cartesiano de cada servo y el tiempo en ms que necesita para desplazarse.

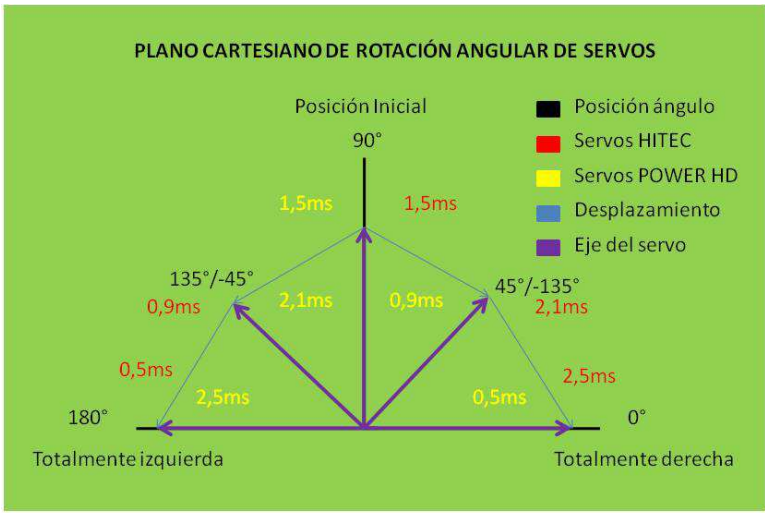


Figura 2.20: Plano cartesiano de rotación angular de cada servo

Fuente: Los autores

2.8.10 Modelos de servos POWER HD

Los modelos de servos POWER HD que se utilizarán para la elaboración de los robots son:

- HD-3001HB
- HD-1501MG

Figura 2.21: Servos POWER HD

Fuente: Los autores

Los servos HD-3001HB y HD-1501MG, físicamente son muy similares, pero internamente el servo HD-1501MG tiene un sistema de engrane mucho más rígido, compuesto por engranes metálicos que le proporcionaran un torque superior al servo en su eje, mientras que el servo HD-3001HB, simplemente su caja reductora está compuesta por engranes plásticos, haciéndolo mucho más débil que el HD-1501MG.

Las características principales de cada servo se detallan en la tabla 4, donde se puede observar que la mayor diferencia entre los dos es su torque.

Especificaciones	**HD-3001HB**	**HD-1501MG**
Tipo de motor	3 polos	3 polos
Velocidad (4.8V/6.0V)	0.15 / 0.12 sec @ 60 grad.	0.16 / 0.14 sec @ 60 grad.
Torque Kg/cm (4.8V/6.0V)	3.5 / 4.4	15.5 / 17.0
Medidas en milímetros	40.70 X 20.50 X 39.50	40.70 X 20.50 X 39.50
Peso en gramos	43.00	63.00
Pulso requerido	3.5Vpp onda cuadrada	3.5Vpp onda cuadrada
Sistema de Control	Control de ancho de pulso	Control de ancho de pulso
Caja de engranaje	Plástica	Metálica

Tabla 4: Especificaciones de los servos POWER HD

Fuente: Datasheet servos POWER HD

2.8.11 Ventaja de utilizar servos HD1501MG

Aparte de un ventajoso torque que posee, que es de 17kg/cm a 6V de trabajo, éste servo permite solucionar mucho el diseño del software de control, ya que normalmente un servo es controlado por salidas de hardware tipo PWM que poseen los microcontroladores, o por rutinas de programación que impliquen el uso de interrupciones o lasos “loop” para mantener el tren del pulso de control, el mismo que permitirá mantener su posición angular.

Este tipo de servo posee una especie de “memoria interna”, cuya ventaja es la de no necesitar de un de laso loop para mantenerlo en una determinada posición, ni tampoco necesita utilizar el PWM por hardware del microcontrolador ni de interrupciones, basta únicamente con enviar la señal de control una sola vez para que éste permanezca en la posición deseada, logrando así que el programa principal se encargue de realizar otras operaciones, reduciendo también el uso de la memoria del PIC considerablemente. Cabe recalcar que esta ventaja es solo para los servos HD1501MG y no para los HD3001HB.

2.9 GLCD

Los GLCD - *Grafic Liquid Controller Display* (Controlador Líquido de Display Gráfico), son pantallas monocromáticas (de un solo color, blanco o negro) cuyo controlador permite la representación de gráficos sobre la misma pantalla.

El GLCD proporciona mucha flexibilidad. Está compuesto de pixeles ubicados en filas y columnas. Cada pixel puede manejarse individualmente y permite mostrar gráficos, textos o la combinación de ambos como lo muestra la figura 2.22.

Figura 2.22: Pantalla GLCD con mensaje gráfico y texto

Fuente: http://www.coldfire-electronica.com/esp/index/item/13/pantalla-glcd-graficadora-128x64

2.9.1 Controladores para pantalla GLCD

Existen varios tipos de controladores para pantallas GLCD según su marca, pero entre los más comunes y usados tenemos:

- Controlador KS0108 de la marca SAMSUNG
- Controlador T6963C de la marca TOSHIBA

La ventaja de utilizar estos controladores es la simplicidad que implica poder controlarlos y que solamente se necesitaría enviarle “ordenes” para que estos las “ejecuten” y hagan mostrar el contenido deseado en la pantalla GLCD.

Cabe recalcar que estos controladores vienen directamente montados sobre la PCB de la GLCD, es decir físicamente son uno solo (controlador y pantalla GLCD), pero se diferencian en los pines de conexión y los comandos de programación.

En la figura 2.23 se puede apreciar el aspecto físico de estas dos pantallas con diferentes controladores.

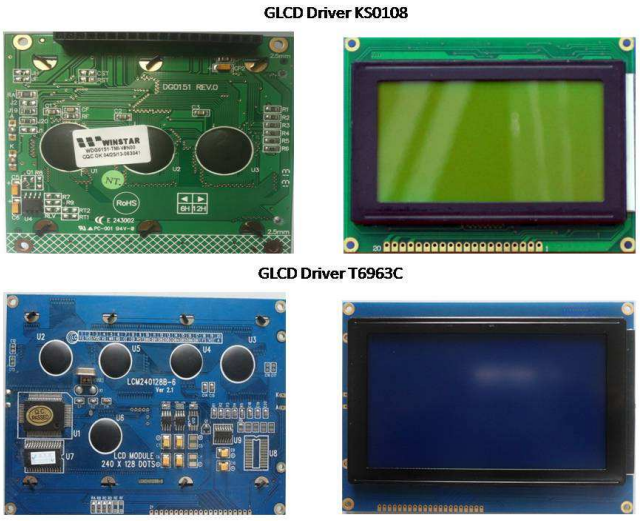


Figura 2.23: Aspecto físico de GLCD con diferentes controladores

Fuente: Los autores

Para el desarrollo de la investigación nos enfocaremos en las pantallas GLCD con controlador KS0108 de Samsung.

2.9.2 Configuración de pines para GLCD con controlador KS0108

Es muy importante saber la configuración de los pines de conexión de las GLCD antes de realizar cualquier diseño, ya que varían según su tipo de controlador.

Los pines se dividen en 5 grupos que son:

- Pines de alimentación.
- Pines de control.
- Pines de datos.
- Pines de backlight.
- Pin Vee (-Vout) y Vo.

2.9.2.1 Pines de alimentación

Estos pines son los encargados de energizar la pantalla GLCD para que su electrónica interna esté lista para trabajar. Es necesario respetar la polaridad de

estos pines ya que una mala conexión dañaría la pantalla GLCD. Estos pines tienes las siglas VDD para el positivo (+5V) y VSS para el negativo.

2.9.2.2 Pines de control

Los pines de control son los encargados de realizar la comunicación entre el microcontrolador externo PIC y el controlador de la GLCD, evitando "choques" al momento de enviar o recibir los datos, ya que estos controlan cuando se está escribiendo en la pantalla y cuando se va a leer un dato desde la pantalla, cuando se memoriza un valor o cuando está en estado de reposo la pantalla. Los pines de control son los siguientes: D/I (data input – para ingresar datos a la pantalla), R/W (read/Write – lectura o escritura de la pantalla), E (enable - habilitador de la GLCD), RST (restart – reinicio de la GLCd).

Adicional a estos pines están los pines CS1 y CS2 (CS – chip select o selector de chip), que nos permiten seleccionar la "hoja" o el lado donde se vaya a escribir, siendo esto muy importante de considerar, ya que si se configura mal, se escribiría el mensaje deseado desde la mitad de la pantalla hasta el final y no desde el inicio hasta el final, como lo muestra la figura 2.24.

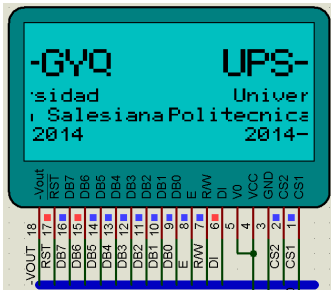


Figura 2.24: Pantalla GLCD con mal control en CS1 y CS2

Fuente: Los autores

2.9.2.3 Pines de datos

Por estos pines se van a cargar los datos que se deseen enviar hacia la pantalla, lógicamente los pines de control deben estar seleccionados en la opción de "escritura" para poder "escribir" sobre la pantalla. Están conformados por 8 pines

ya que se usan 8 bits para la transmisión de datos y su nomenclatura es desde D0 hasta D7.

2.9.2.4 Pines de backlight

Estos pines son los encargados de encender la luz de fondo de la GLCD para poder visualizar los caracteres, gráficos o textos que se escriben sobre la pantalla. Los conforman los pines A y K, que básicamente son los pines de un diodo led.

2.9.2.5 Pin Vee (-Vout) y Vo

Estos son los pines utilizados para control de contraste de la GLCD junto con el pin VO con muestra la siguiente figura.

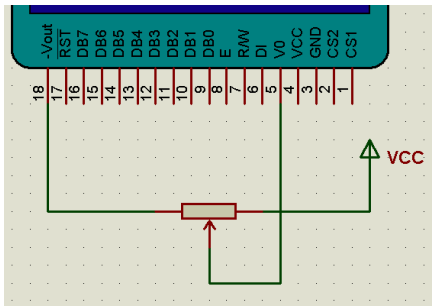


Figura 2.25: Configuración de pines de la pantalla GLCD con controlador KS0108 simulado en POTEUS

Fuente: Los autores

En la figura 2.26 podemos observar la configuración de pines de una pantalla GLCD real con controlador KS0108, anotando que el pin 1 empieza en Vss y el pin 20 termina en K.

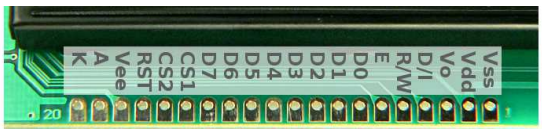


Figura 2.26: Aspecto físico de la configuración de pines pantalla GLCD con controlador KS0108

Fuente: https://www.pjrc.com/teensy/td_libs_GLCD.html

2.10 Paneles Táctiles

Hoy en día los paneles táctiles son muy utilizados para el diseño de aplicaciones electrónicas, ya que simplifican el uso del hardware del microcontrolador en caso de aplicaciones que requieran el uso de varias botoneras por ejemplo.

2.10.1 Panel táctil resistivo, principio de funcionamiento

Este tipo de paneles están conformadas por dos laminas resistivas, que al unirse (con el dedo o con el lápiz apropiado) generan un diferencial de voltaje, el mismo que será leído por el controlador y luego la interpreta para sacar las coordenadas en el eje x y en el eje y.

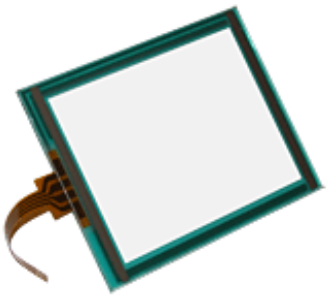


Figura 2.27: Panel táctil resistivo
Fuente: http://www.mikroe.com

Para conseguir que la pantalla entregue el diferencial de voltaje, es necesario energizarla con una alimentación de +5Vdc y posterior a esto conectarla a su respectivo controlador, el mismo que interpreta las señales analógicas que la pantalla envíe una vez pulsada.

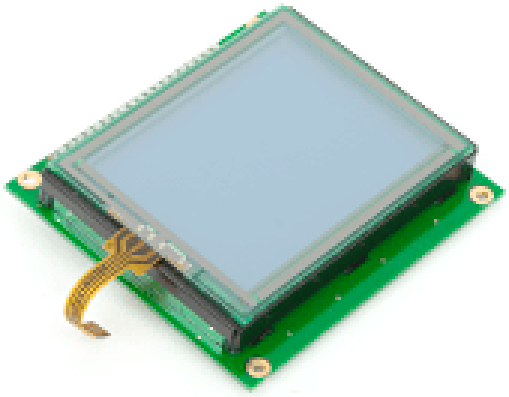


Figura 2.28: Panel táctil resistivo montado sobre una pantalla GLCD

Fuente: http://www.mikroe.com

2.10.2 Configuración de pines

Los pines de la pantalla táctil se encuentran a lo largo de un cable panol, el mismo que contiene los pines de configuración del panel táctil.

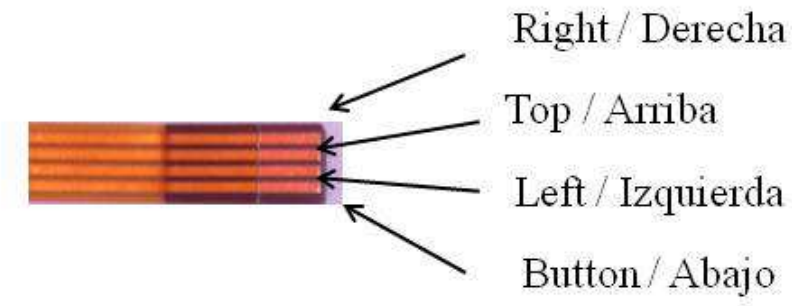


Figura 2.29: Configuración de pines de un panel táctil

Fuente: Los autores

2.10.3 Controlador para panel táctil resistivo

Este controlador está conformado por un circuito electrónico simple compuesto de resistencias, transistores y capacitores.

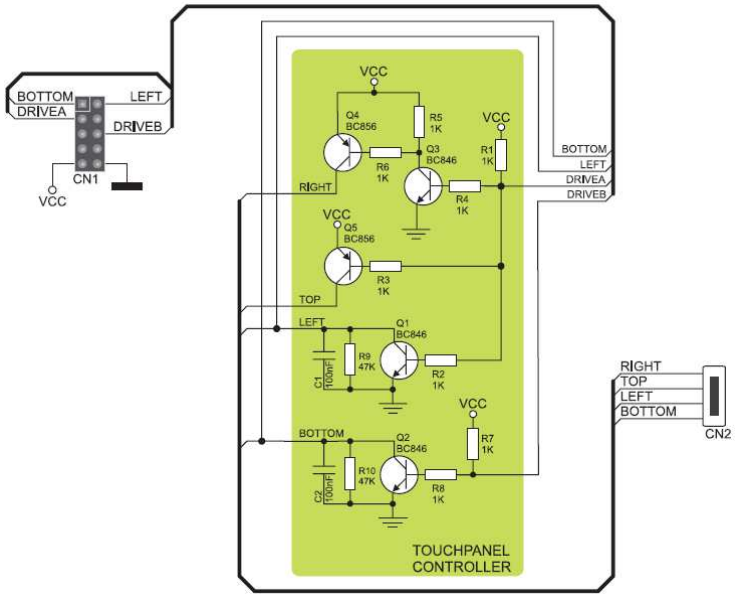


Figura 2.30: Esquemático del controlador para panel táctil resistivo

Fuente: http://www.mikroe.com

2.10.4 Conexión entre un panel táctil y un microcontrolador

El panel táctil deberá ir montado sobre una pantalla GLCD, luego conectado al circuito controlador para panel táctil resistivo y por último al microcontrolador PIC, respetando los pines de configuración de la GLCD y los pines para el control del panel táctil como se muestra en la figura 2.31.

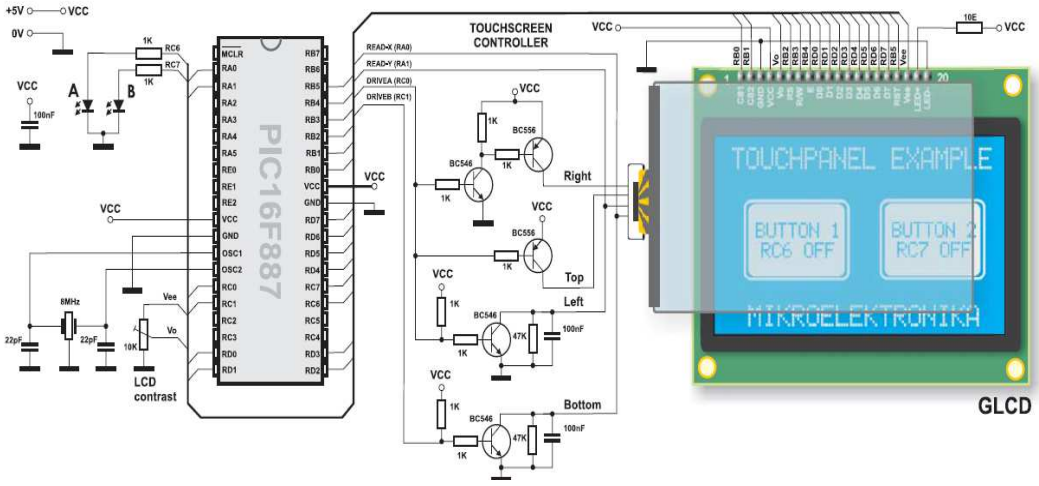


Figura 2.31: Conexión entre microcontrolador, pantalla GLCD y panel táctil

Fuente: http://www.mikroe.com

2.11 Módulo de Voz ISD1932

Este módulo es fabricado por la empresa SPARKFUN y está compuesto por una memoria de voz ISD1932 y toda la electrónica necesaria para poder interactuar con la memoria, ya sea para grabar un mensaje o para reproducirlo. En el caso de reproducir el mensaje, el módulo incorpora la etapa de amplificación para conectar un parlante de 4 Ohm.

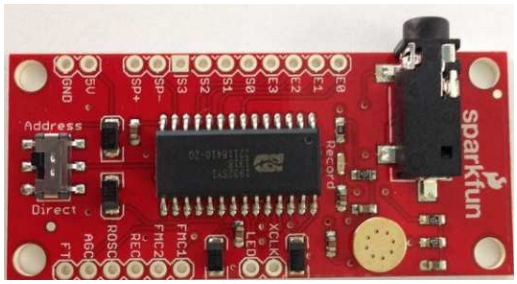


Figura 2.32: Módulo Sparkfun ISD1932

Fuente: Los autores

2.11.1 Modos de operación del módulo de voz

Este módulo posee dos modos de operación, los cuales son: modo dirección y modo directo, los mismos que poseen diferente configuración de pines según su modo de operación.

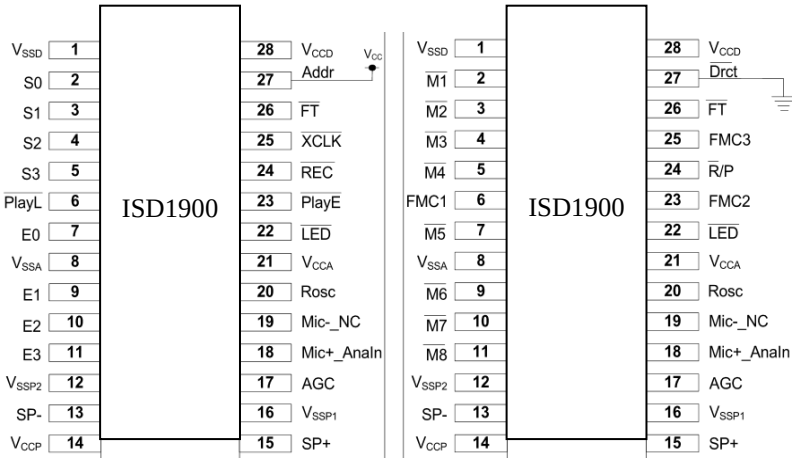


Figura 2.33: Configuración de pines en modo Dirección (izquierda) y Directo (derecha)

Fuente: Datasheet módulo de voz Sparkfun ISD1900

2.11.1.1 Modo Directo

En este modo, los pines de configuración son FMC1, FMC2 y FMC3 que nos permitirán definir hasta 8 mensajes con duración similar de grabado.

Los pines de control están comprendidos entre M1 hasta M8 (cada uno activa un mensaje diferente) y el pin R/P permitirá seleccionar entre grabar un mensaje o reproducirlo.

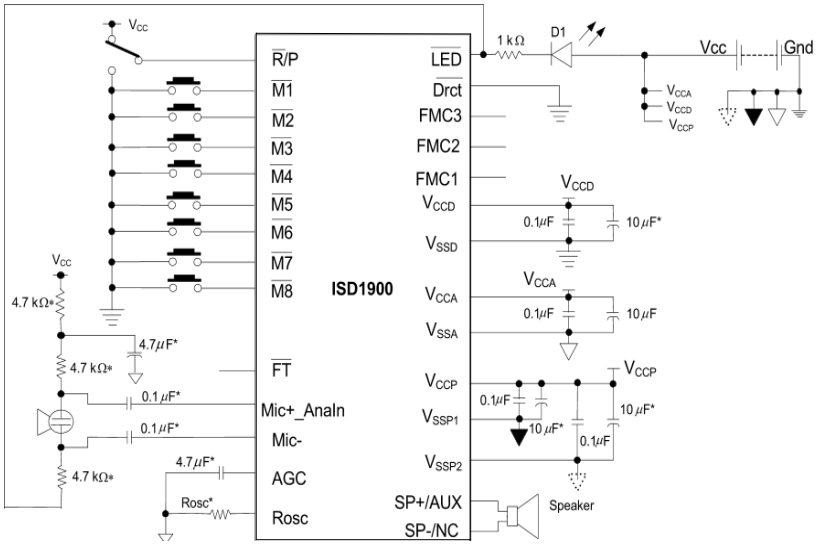


Figura 2.34: Diagrama de conexión en Modo Directo
Fuente: Datasheet módulo de voz Sparkfun ISD1900

2.11.1.2 Modo Dirección

En este modo, los pines de configuración están definidos por una dirección inicial de memoria (S0, S1, S2, S3) y una dirección final de memoria (E0, E1, E2, E3), dividiendo así el bloque de memoria en 16 espacios.

Los pines de control REC, PLAYE, PLAYL son los encargados de hacer ejecutar las órdenes de grabación de mensajes, reproducción de los mismos.

La función de grabación o /REC, está conectada con un pulsador a tierra, lo que indica que al activarse, se procederá a grabar un mensaje de voz desde la dirección de memoria inicial (S3, S2, S1, S0) hasta la dirección de memoria final (E3, E2,

E1, E0) y de este rango de memoria (dirección final – dirección inicial) dependerá el tiempo en segundos del mensaje a grabar y además de eso el diodo LED se encenderá, indicando que se está grabando un nuevo mensaje.

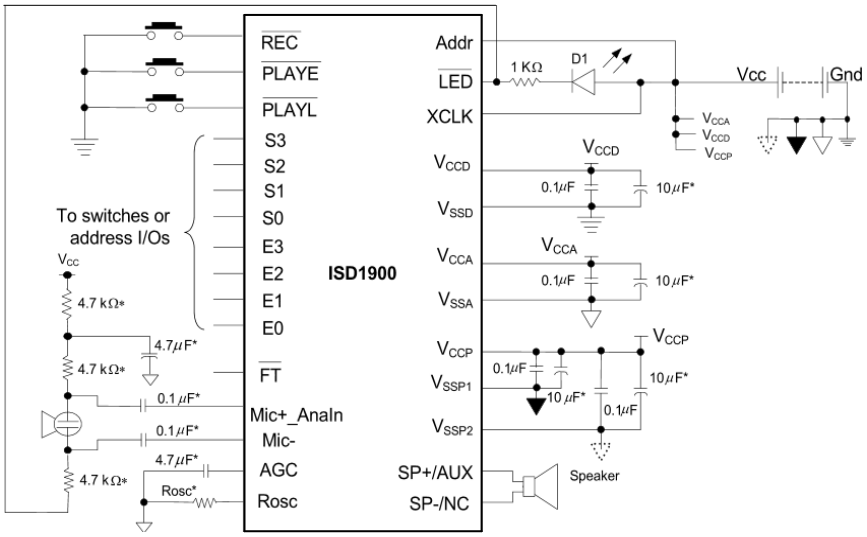


Figura 2.35: Diagrama de conexión en Modo Dirección

Fuente: Datasheet módulo de voz Sparkfun ISD1900

2.12 Sensor LDR

El sensor LDR -*Light Dependent Resistor* (Resistencia Dependiente de Luz), también llamado Fotocelda o Fotoresistencia es un sensor resistivo cuyo valor de resistencia depende de la intensidad de luz que incida sobre ella, teniendo un valor elevado como no hay luz o en la oscuridad, llegando a varios mega ohmios y un valor muy bajo cuando hay luz sobre ella pudiendo llegar a unos pocos ohmios.

Figura 2.36: Aspecto físico da una LDR

Fuente: http://www.evelta.com/industrial-control/sensors/ldr-12mm-metal-casing

2.12.1 Divisor de voltaje con LDR

Este circuito permitirá tener un valor en porcentaje (%) de la intensidad de luz que incide sobre la fotocelda, el mismo que deberá ser detectado por un puerto analógico del microcontrolador ya que enviará valores comprendidos entre los 0,0V y 5,0V.

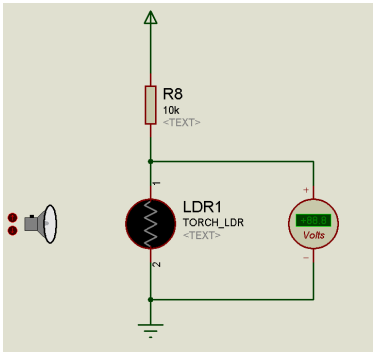


Figura 2.37: Divisor de voltaje con LDR

Fuente: Los autores

Este divisor de voltaje permitirá saber cuando la intensidad de luz es alta o cuando es baja, ya que enviará una señal de voltaje analógica proporcional a la intensidad de luz que incide sobre la LDR como muestra la figura 2.38.

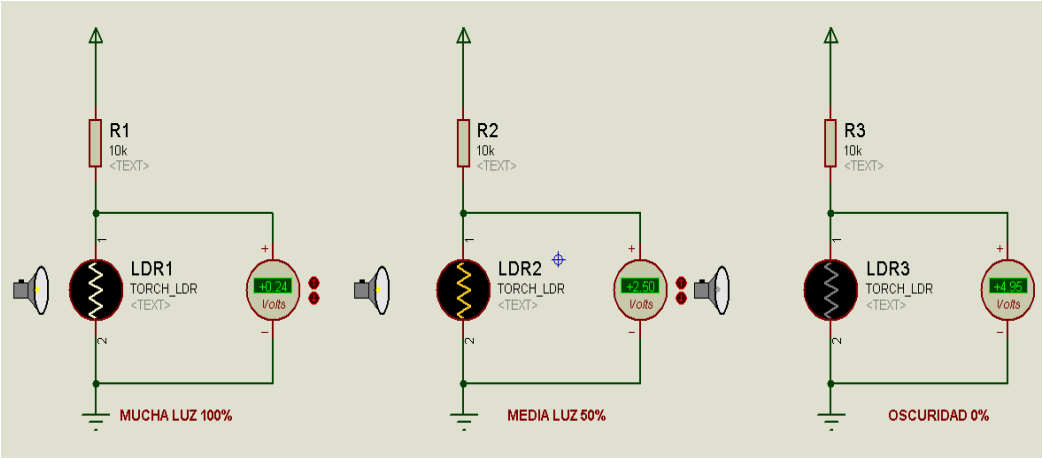


Figura 2.38: Salida de voltaje según la intensidad de luz sobre la LDR

Fuente: Los autores

2.13 Sensor Ping – Parallax

Figura 2.39: Sensor Ping de Parallax

Fuente: http://www.parallax.com/product/28015

Este tipo de sensor ultrasónico proporciona un método muy fácil para medir distancias, siendo uno de los sensores con mayor uso para este tipo de aplicaciones gracias a su precisión, utilización de pocos recursos y amplio rango de medición ya sea ésta en movimiento o estática.

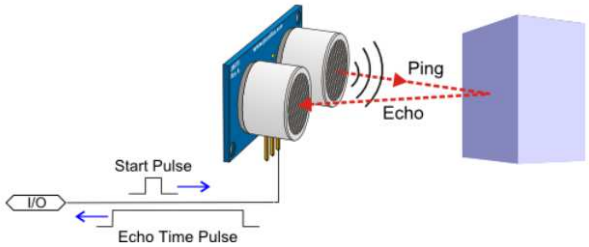


Figura 2.40: Forma de medición con el sensor Ping de Parallax

Fuente: http://learn.parallax.com/KickStart/28015

2.13.1 Características del Sensor Ping

Entre las características principales tenemos:

- Brinda precisión para mediciones dentro de un rango comprendido entre 2cm a 3m
- La medición por ultrasonido puede trabajar en ambientes con mucha luminosidad, haciéndolo la mejor opción para medir distancias dejando a un lado los métodos por infrarrojo
- Su comunicación es muy simple, utilizando únicamente un pin

- Posee un indicador led que indica cuando la medición está en proceso
- Con su conector de 3 pines en su cabecera, se hace muy simple y fácil al momento de conectarlo al protoboard (tablero para prototipos o pruebas), ya que no se necesita utilizar cables externos
- Duración del pulso para el eco, mínimo de 115us, y máximo de 18.5ms
- El sensor trabaja a un voltaje de 5Vdc, con lógica TTL

2.13.2 Aplicaciones

Entre las aplicaciones más destacadas tenemos:

- Sistemas de seguridad
- Sistemas para parqueos
- Robótica en general

2.13.3 Conexión del sensor Ping con el microcontrolador PIC

La interfaz de conexión con el microcontrolador es muy simple, únicamente se necesita un pin que se lo pueda configurar como entrada y salida de datos. Una vez que tengamos ese pin, simplemente basta con enviar una señal de excitación o "trigger" para habilitar el sensor PING y esperar su respuesta (el pin debe cambiar de salida a entrada), luego se calcula la distancia midiendo el tiempo que se demora la señal de respuesta de pasar de estado alto a bajo.

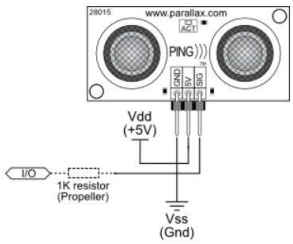


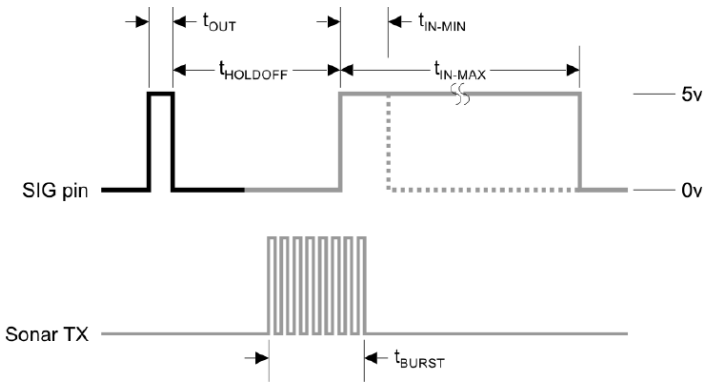
Figura 2.41: Conexión del sensor Ping con un microcontrolador

Fuente: http://learn.parallax.com/KickStart/28015

Lo que se debe considerar para tener la medida exacta, son los tiempos que el fabricante recomienda para su buen funcionamiento, los cuales son los siguientes:

- Duración de pulso de inicialización o Trigger: 2us mínimo, típico 5us
- Tiempo de respuesta máxima : 18.5ms
- Tiempo de respuesta mínima : 115us
- Espera para re inicialización de medida : 200us

Esos tiempos los podemos ver reflejados en la figura 2.42.



▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

Figura 2.42: Tiempos necesarios para poder medir con un sensor Ping de Parallax

Fuente: http://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf

2.14 Sensor SRF05

Figura 2.43: Aspecto físico sensor SRF05

Fuente: http://www.superrobotica.com/s320111.htm

Este sensor permite medir distancias por ultrasonido emitiendo una frecuencia que al llegar a un objeto rebota. El tiempo que demora la onda ultrasónica en ir y venir es proporcional a la distancia entre el sensor y el objeto detectado.

Este sensor es el sucesor del conocido sensor de distancias por ultrasonido SRF04, con la diferencia que el SRF05 posee dos modos de trabajo, el primero compatible con el SRF04 y el segundo propio del SRF05. Además de eso la distancia de medición ha aumentado de 3 a 4 metros.

2.14.1 Características

- Rango de detección desde 2cm hasta 4m
- Voltaje de alimentación de 5Vdc
- Dos modos de trabajo, uno a 3 pines y otro a 4 pines (compatible con el SRF04)
- Tiempo de espera entre inicialización y medición de 50ms
- Pulso de inicialización de 10uS
- Pulso de medición entre 100uS y 25ms, pasado los 30ms se considera que no detectó ningún objeto dentro de su rango de trabajo

2.14.2 Configuración de pines

Los pines utilizados dependen del modo de trabajo que vayamos a utilizar, pero en general éste sensor posee 5 pines, los cuales son:

- Alimentación positiva o VCC, generalmente son 5Vdc
- Alimentación negativa o GND
- Pin de inicialización
- Pin de eco o “echo”
- Pin N/C o de no conexión

.

2.14.3 Modos de trabajo

El sensor SRF05 posee dos modos de trabajo los cuales son:

2.14.3.1 Modo 1- Señal de inicialización y eco independientes:

Este modo usa la configuración de pines idéntica a la del sensor SRF04, utilizando 4 pines los cuales son: Vcc, Gnd, Echo, Sign.

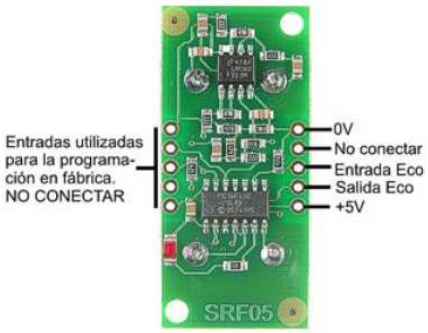


Figura 2.44: Configuración de pines modo 1

Fuente: http://www.superrobotica.com/s320111.htm

En la figura 2.45 podemos observar el diagrama de tiempos que necesita el sensor en este modo.

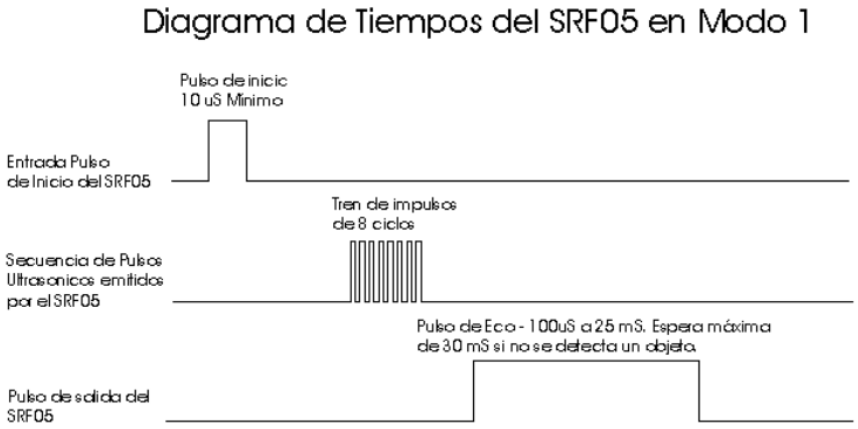


Figura 2.45: Diagrama de tiempos en modo 1

Fuente: http://www.superrobotica.com/s320111.htm

2.14.3.2 Modo 2 – Pin único para activación y eco

Este modo utiliza únicamente un solo pin para la inicialización del sensor y para la recepción de la señal de eco. En este modo se debe configurar el pin del

microcontrolador para que trabaje como salida de datos (inicialización) y entrada de datos (recepción del eco).

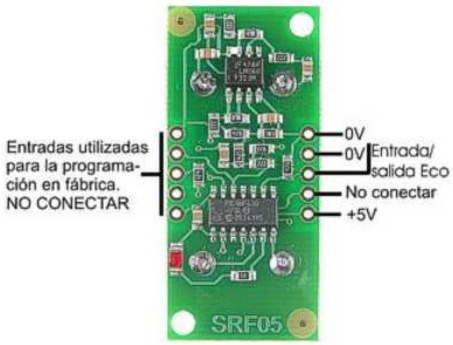


Figura 2.46: Configuración de pines en modo 2

Fuente: http://www.superrobotica.com/s320111.htm

En la figura 2.47 podemos observar el diagrama de tiempos que necesita el sensor en este modo.

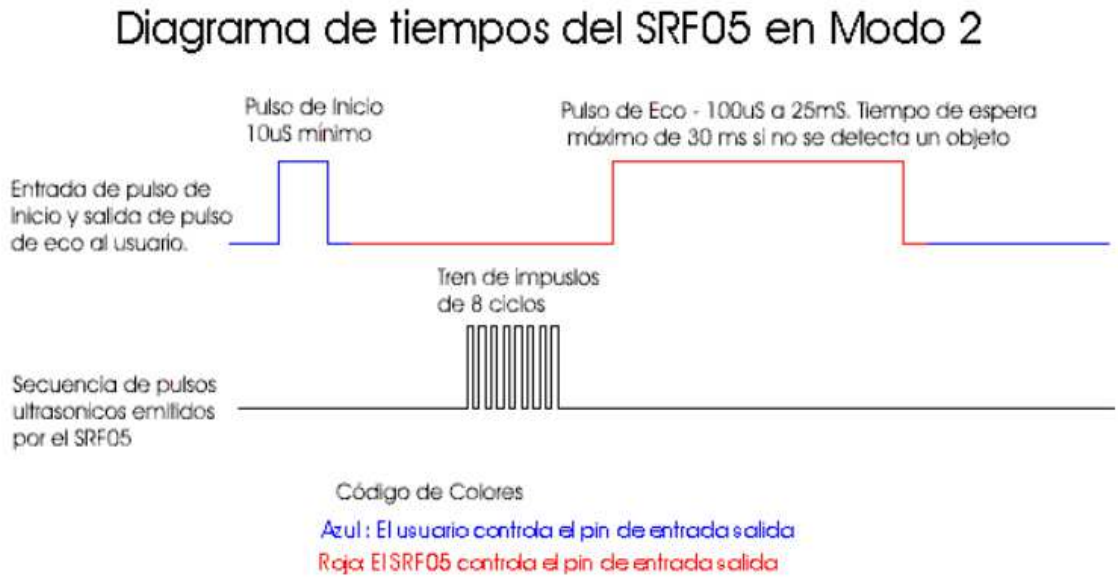


Figura 2.47: Diagrama de tiempos en modo 2

Fuente: http://www.superrobotica.com/s320111.htm

2.15 Sensor de Temperatura LM35

Este sensor de temperatura es de tipo analógico, lo que significa que su señal de salida no es digital (datos binarios 1-0), sino, como salida envía tensiones de voltaje comprendidas entre los 0.0V y los 5.0V, teniendo una resolución de 10mv/ºC.

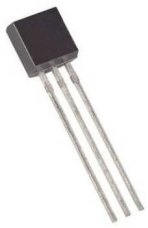


Figura 2.48: Sensor de temperatura LM35

Fuente: http://www.electronicoscaldas.com/sensores-de-temperatura/194-sensor-de-temperatura-lm35.html

Este sensor es muy utilizado a nivel de electrónica por su facilidad al momento de adquirir la señal que envía y transformarla a un valor de temperatura real, ya que solo se necesita "leer" un puerto análogo del microcontrolador, luego transformarlo por la resolución que usa el convertidor análogo/digital seleccionado y tendremos el valor de la temperatura real en ese momento dado.

2.15.1 Características

- Fácil de utilizar gracias a su salida analógica
- Pequeño tamaño
- Mide temperaturas en un rango comprendido entre -55ºC hasta 150ºC
- Salida de señal lineal de 10mv/ºC
- Alimentación desde los 4Vdc hasta los 30Vdc
- Baja impedancia de salida, de 0.1Ω para una carga de 1mA

2.15.2 Tipos de encapsulados

Este tipo de sensor viene en tres tipos diferentes de encapsulado, los cuales son el TO-92, SOIC-8, TO220 y el TO tal como lo muestra la figura 2.49.

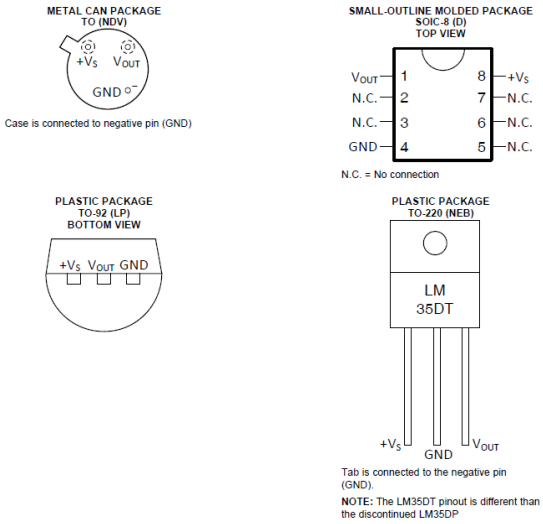


Figura 2.49: Tipos de encapsulados disponibles para el sensor LM35

Fuente: Datasheet del sensor LM35

2.15.3 Configuración de pines

Los pines del sensor de temperatura LM35 son básicamente 3 (Para el sensor en encapsulado SOIC-8 son 8 pines pero solo se usan 3), los cuales son:

- **Vcc:** Es la alimentación positiva (+) del sensor, conocida también como Vs y generalmente es de 5Vdc
- **Vout:** Es el pin de salida de señal análogo del sensor, comprendida en un rango de 0-5Vdc con resolución de 10mn/ºC.
- **Gnd:** Conexión a tierra o negativo (-) del sensor

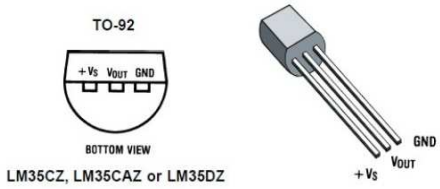


Figura 2.50: Configuración de pines del sensor LM35

Fuente: http://www.electronicoscaldas.com/sensores-de-temperatura/194-sensor-de-temperatura-lm35.html

2.15.4 Modos de operación

Este sensor puede trabajar en dos modos, el modo básico y el modo full rango.

2.15.4.1 Modo básico

El sensor se configura automáticamente para trabajar en un rango de temperatura comprendido entre los 2ºC hasta los 150ºC, este es el modo de operación más utilizado. En este modo la salida a 2ºC es de 0V aumentando 10mv/ºC.

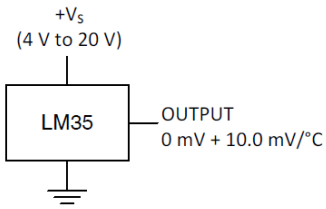


Figura 2.51: Sensor LM35 en modo básico
Fuente: Datasheet sensor LM35

2.15.4.2 Modo Full Rango

En este modo el sensor necesita de una resistencia externa para configurarse en modo Full Rango, aumentando su capacidad sensorial en un rango comprendido entre los -55ºC hasta los 150ºC. En este modo la relación es directa entre salida y temperatura teniendo valores como: 1500mV a 150ºC, 250mV a 25ºC y -550mv a -55ºC.

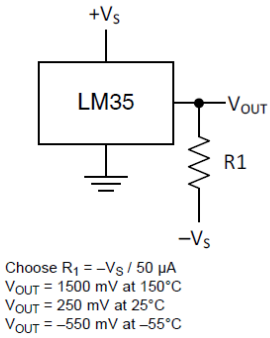


Figura 2.52: Sensor LM35 en modo Full Rango

Fuente: Datasheet sensor LM35

2.16 Sensor de Temperatura DS18B20

Este tipo de sensor de temperatura es digital, lo que significa que envia el valor de la temperatura en una trama de datos configurable de 9 a 12 bits usando la comunicación “one wire”. Se lo puede configurar para resoluciones de 0.1ºC o de 0.5ºC, siendo un sensor muy preciso.

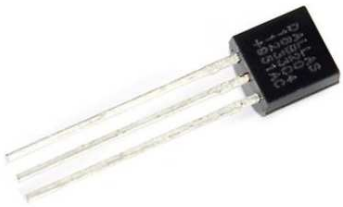


Figura 2.53: Aspecto fisico sensor de temperatura DS18B20

Fuente: http://www.reuk.co.uk/DS18B20-Temperature-Sensor-with-Raspberry-Pi.htm

2.16.1 Características

- Únicamente necesita de un solo cable para su comunicación
- Capacidad de conexión a multipunto para muestrear varias temperaturas bajo un mismo bus
- No requiere de elementos externos
- Puede ser alimentado desde su propia linea de dato sin necesidad de una fuente externa

- Voltaje de operación varía desde los 3.0V hasta los 5.5V
- Medición de temperatura desde los -55ºC hasta los 125ºC
- Resolución programable de 9 a 12 bits
- Se lo puede utilizar como termostato

2.16.2 Palabras de control

Este sensor convierte la lectura de temperatura de 9 o 12 bits a una palabra o "word" digital de 16 bits, este señal es la que el microcontrolador la "lee" para interpretarla y saber el valor de la temperatura. En la figura 2.54 se muestran algunos valores de estas "palabras" que envía el sensor y junto a ella su valor de temperatura.

Para conseguir esta salida, hay que configurar previamente al sensor, inicializarlo y enviarle palabras de control para que el sensor las interprete y sepa que decisión tomar (Véase ANEXO 2).

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+125°C	0000 0111 1101 0000	07D0h
+85°C	0000 0101 0101 0000	0550h*
+25.0625°C	0000 0001 1001 0001	0191h
+10.125°C	0000 0000 1010 0010	00A2h
+0.5°C	0000 0000 0000 1000	0008h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1000	FFF8h
-10.125°C	1111 1111 0101 1110	FF5Eh
-25.0625°C	1111 1110 0110 1111	FF6Fh
-55°C	1111 1100 1001 0000	FC90h

Figura 2.54: Salida digital o "word" que envía el sensor DS18B20

Fuente: Datasheet sensor DS18B20

2.16.3 Configuración de pines

Este sensor utiliza 3 pines para el encapsulado TO-92 y 8 pines para el encapsulado SOIC, los cuales son:

- VCC: alimentación positiva de 3.0 - 5.5 Vdc
- DI/O: ingreso y salida de información, es por donde el microcontrolador envía la trama de datos de control y por donde recibe la “Word” o “palabra” con la información del valor de temperatura
- GND: conexión a tierra o negativo de la fuente de poder

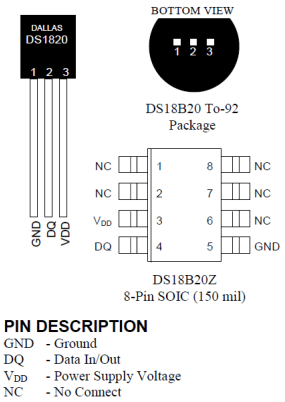


Figura 2.55: Distribución y configuración de pines en sus dos tipos de encapsulado

Fuente: Datasheet sensor DS18B20

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DE LOS SISTEMAS RELACIONADOS A LOS ROBOTS TRANSFORMER

3 DISEÑO E IMPLEMENTACIÓN DE LOS SISTEMAS RELACIONADOS A LOS ROBOTS TRANSFORMER

3.1 Introducción

En este capítulo se describirá el desarrollo de la estructura general de los robots, así como también el desarrollo de los diversos módulos de control, programación y su función dentro del robot.

Dentro del módulo de control que conforma a los robots tenemos las siguientes tarjetas:

- Tarjeta de control principal
- Tarjeta de control de sensores
- Control remoto (no va dentro del robot)

3.2 Diseño del módulo de control principal para el robot humanoide

Para su diseño final, empezamos desarrollando el prototipo en ISIS de PROTEUS, para poder simular la programación y realizar pruebas de control del robot como muestra en la figura 3.1.

.

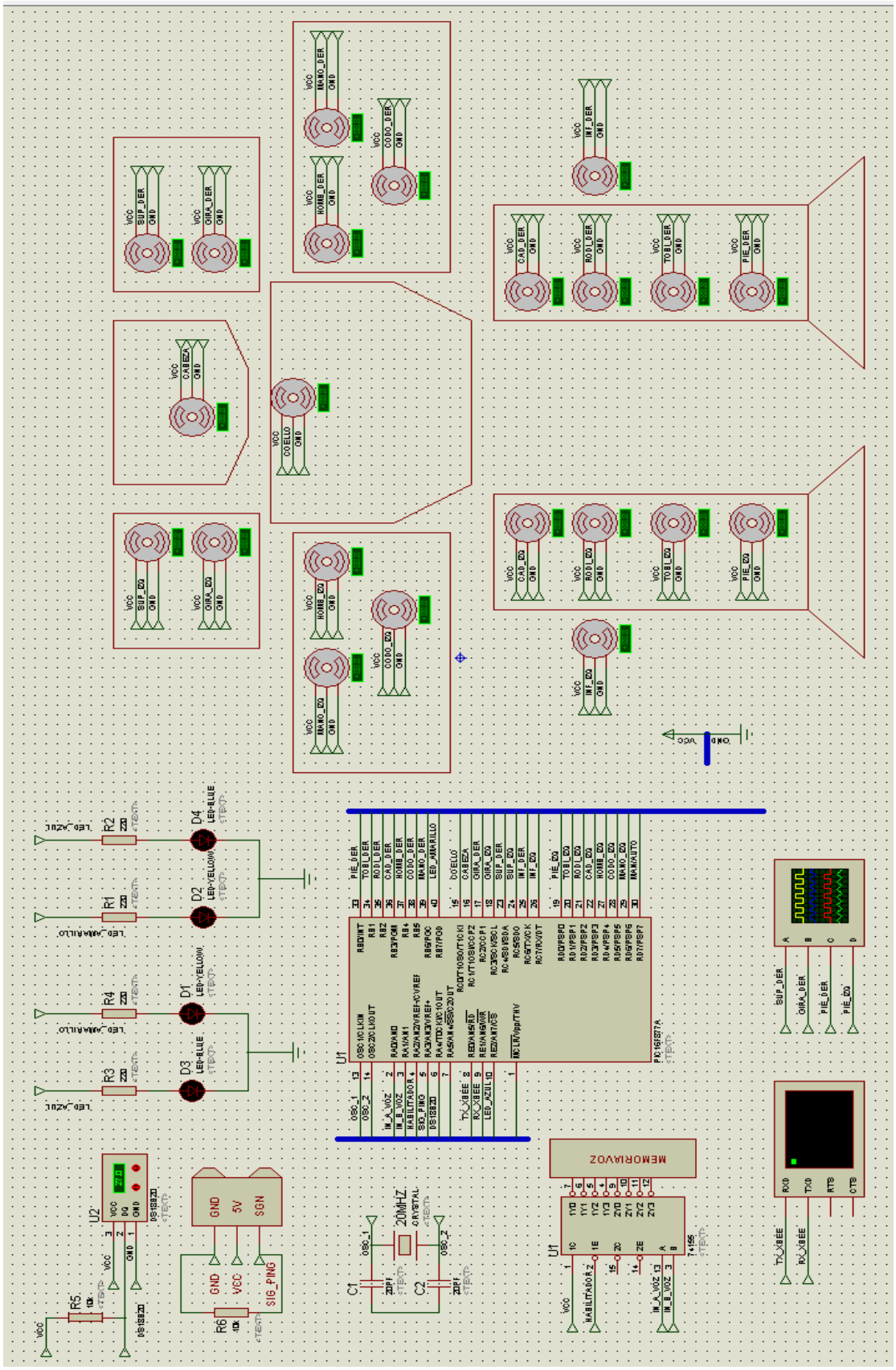


Figura 3.1: Esquemático del módulo de control simulado en ISIS PROTEUS

Fuente: Los autores

3.2.1 Configuración de conexiones del módulo de control principal

Esta configuración nos indica la forma de realizar las conexiones entre el microcontrolador PIC y los elementos externos, como los servomotores, sensores, memoria de voz, y otros elementos, tal como se muestra en la figura 3.2.

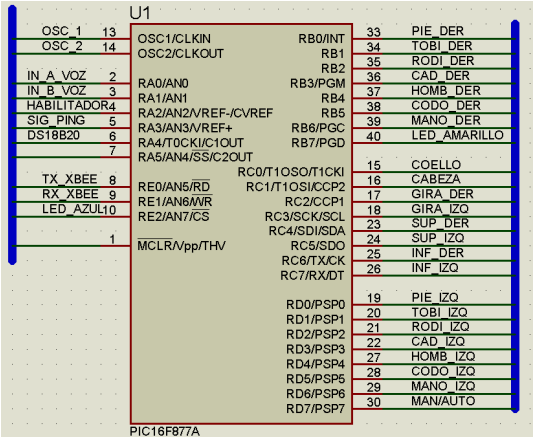


Figura 3.2: Configuración de conexiones del módulo principal

Fuente: Los autores

Cabe anotar que en estas conexiones no se considera la fuente de alimentación de +5V, ya que al tratarse de una simulación, ésta viene conectada por defecto en el microcontrolador. Además es necesario mencionar que la simulación se la realiza con un microcontrolador PIC16F877A en encapsulado DIP, mientras que en la práctica, el módulo de control principal está realizado con el mismo microcontrolador pero en montaje SMD QFT. Como consecuencia tenemos que los números de pines no coinciden, pero el nombre y número del puerto si es decir únicamente cambia el pin físico designado como lo podemos ver en la tabla 5.

PUERTO	DIP	SMD	DESCRIPCIÓN
A0	2	19	Salida para control voz, entrada A
A1	3	20	Salida para control voz, entrada B
A2	4	21	Habilitador tarjera control de voz
A3	5	22	Pin de señal In/Out del sensor PING
A4	6	23	Pin de comunicación con sensor DS18B20

A5	7	24	No usado
B0	33	8	Servo pie derecho
B1	34	9	Servo tobillo derecho
B2	35	10	Servo rodilla derecha
B3	36	11	Servo cadera derecha
B4	37	14	Servo Hombro derecho
B5	38	15	Servo codo derecho
B6	39	16	Servo mano derecha
B7	40	17	Led indicador de señal color amarillo
C0	15	32	Servo del cuello
C1	16	35	Servo de la cabeza
C2	17	36	Servo sostenedor de rueda derecha
C3	18	37	Servo sostenedor de rueda izquierda
C4	23	42	Servo rueda superior derecha
C5	24	43	Servo rueda superior izquierda
C6	25	44	Servo rueda inferior derecha
C7	26	1	Servo rueda inferior izquierda
D0	19	38	Servo pie izquierdo
D1	20	39	Servo tobillo izquierdo
D2	21	40	Servo rodilla izquierda
D3	22	41	Servo cadera izquierda
D4	27	2	Servo hombro izquierdo
D5	28	3	Servo codo izquierdo
D6	29	4	Servo mano izquierda
D7	30	5	Switch selector modo manual / automático
E0	8	25	Pin de transmisión TX hacia el xbee
E1	9	26	Pin de recepción RX hacia el xbee
E2	10	27	Led indicador de señal azul
/MCLR	1	18	Pin de reseteo conectado a VCC general

Tabla 5: Distribución de pines en DIP y SMD del PIC de la tarjeta de control principal

Fuente: Los autores

3.2.2 Diseño del módulo de control principal en ARES de PROTEUS

Una vez simulado el módulo de control principal, se procede a la realización del diseño de su tarjeta física para luego ser probada en forma real. Para esto se utiliza la herramienta de ARES PROTEUS para llevar a cabo el diseño de la tarjeta.

En la figura 3.3 se muestra el diseño de la tarjeta del módulo de control principal y en la figura 3.4 el aspecto físico de la misma en 3D.

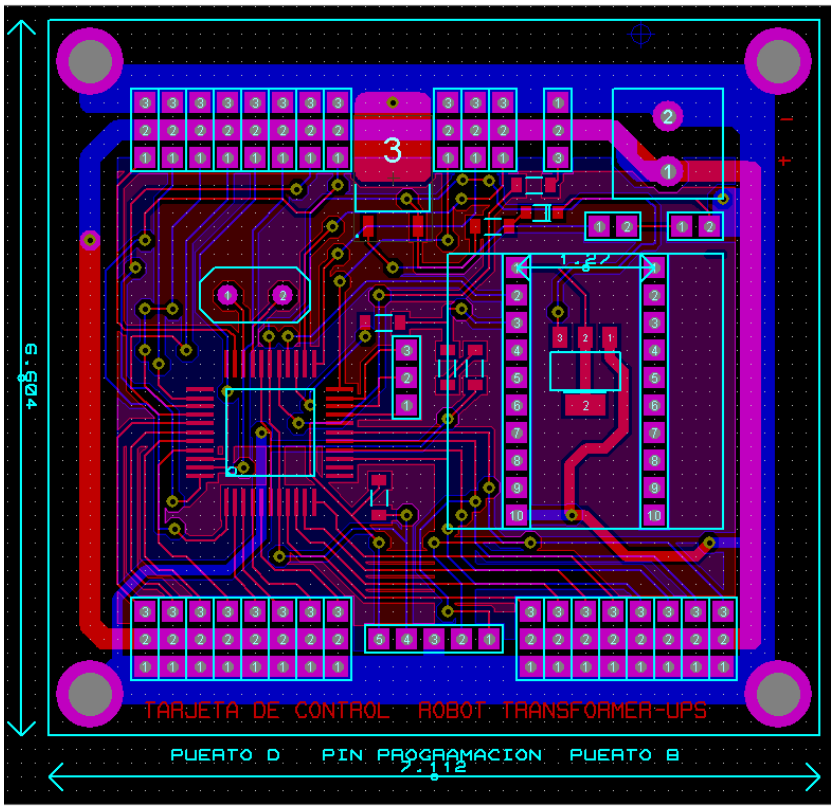


Figura 3.3: Diseño de la tarjeta del módulo de control principal en ARES PROTEUS

Fuente: Los autores

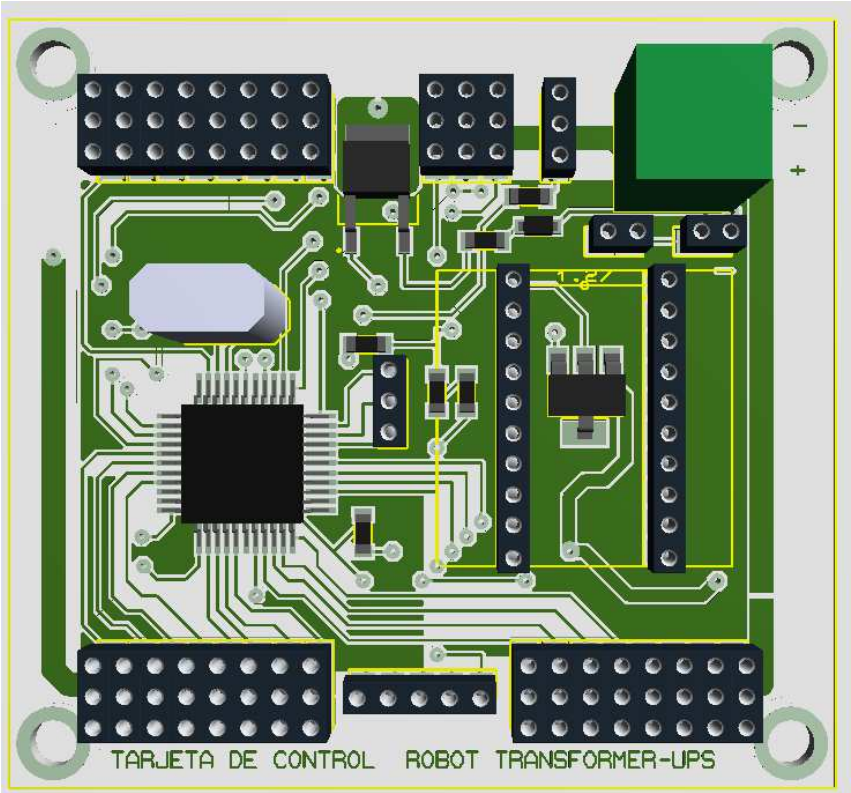


Figura 3.4: Aspecto físico en 3D de la tarjeta del módulo de control principal

Fuente: Los autores

3.2.3 Descripción de la tarjeta de control principal

Esta tarjeta es la encargada de controlar todo lo referente al robot, incluyendo las otras tarjetas, es decir es la tarjeta principal del robot y está conformada por las siguientes partes:

- Fuente de poder principal
- Fuente de poder secundaria
- Microcontrolador principal
- Etapa transmisora – receptora inalámbrica
- Pines I/O para control

En la figura 3.5 se puede apreciar el aspecto físico de la tarjeta de control principal o “mainboard” pero sin el módulo de transmisión y recepción inalámbrica XBEE conectado.

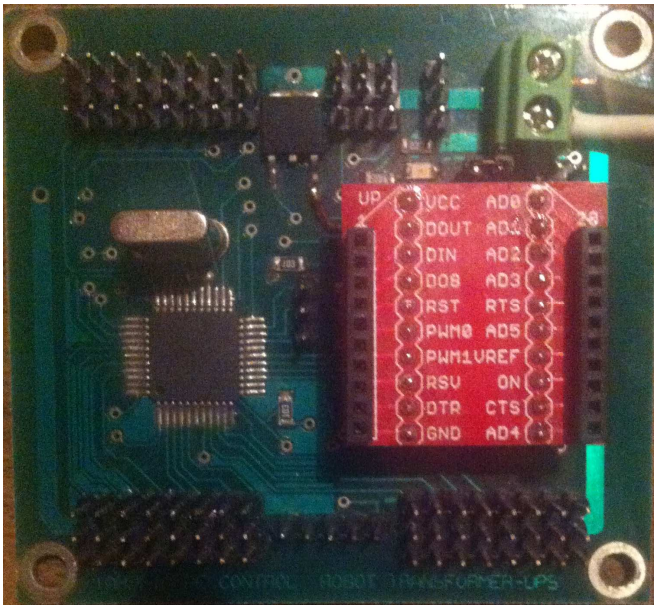


Figura 3.5: Aspecto físico de la tarjeta de control principal

Fuente: Los autores

3.2.3.1 Fuente de poder principal

Esta fuente de poder la conforma el regulador de voltaje LM7805 en montaje superficial. Para funcionar, simplemente le deben llegar los 9V de la batería externa y éste se encarga de reducir el voltaje a +5V que es el voltaje apropiado para el funcionamiento del microcontrolador PIC. En la figura 3.6 se muestra la imagen de este regulador sobre la tarjeta de control principal.

Figura 3.6: Fuente de poder principal conformada por el regulador LM7805 SMD

Fuente: Los autores

En la figura 3.7 se puede apreciar el diagrama esquemático realizado en ISIS PROTEUS.

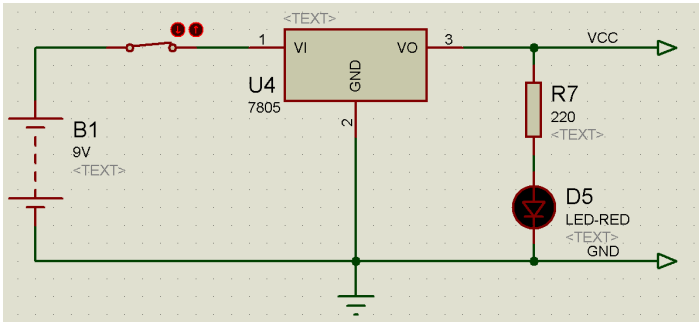


Figura 3.7: Esquemático de la fuente de poder principal realizada en ISIS PROTEUS

Fuente: Los autores

3.2.3.2 Fuente de poder secundaria

Esta fuente está conformada por el regulador de voltaje AMS1117-3.3, el mismo que entrega una salida de voltaje regulada de +3.3V recibiendo en su entrada los +5V que suministra la fuente de poder principal. Este voltaje es necesario para alimentar al módulo de transmisión y recepción inalámbrica conformado por el Xbee.

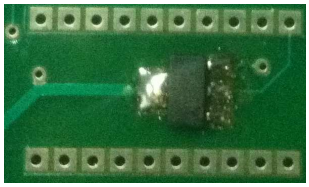


Figura 3.8: Fuente de poder secundaria conformada por el regulador AMS1117-3.3 SMD

Fuente: Los autores

3.2.3.3 Microcontrolador PIC principal

Este microcontrolador es el encargado de gestionar y ejecutar todo el algoritmo de programación que se encuentra en su interior, logrando así controlar a los demás

dispositivos conectados en él, como por ejemplo controla a los servomotores. Este microcontrolador es el PIC16F877A-QFT.

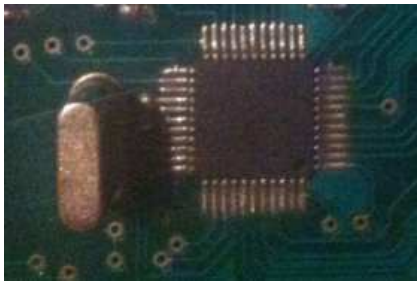


Figura 3.9: Microcontrolador principal PIC 16F877A-QFT

Fuente: Los autores

3.2.3.4 Etapa transmisora – receptora inalámbrica

Esta etapa está conformada únicamente por el módulo de comunicación inalámbrica Xbee, el mismo que se comunica con el microcontrolador principal por medio del protocolo RS-232.

En la figura 3.10 se puede apreciar el aspecto físico de este módulo montado sobre la tarjera de control principal.

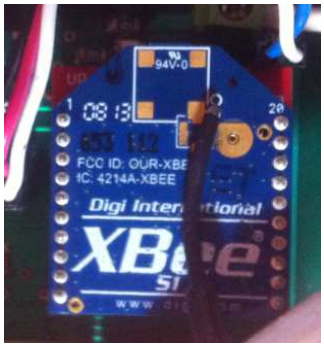


Figura 3.10: Módulo de transmisión inalámbrica XBEE

Fuente: Los autores

3.2.3.5 Pines I/O de control

Los pines I/O (Input / Output o Entrada / Salida) son los pines del microcontrolador PIC utilizados como periféricos de entrada o salida según sea el caso. Estos pines están divididos en cinco grupos, los cuales son:

- Pines para control de servomotores
- Pines para control de modulo de voz
- Pines para comunicación con sensores
- Pines para salidas LED
- Pines de programación ICSP

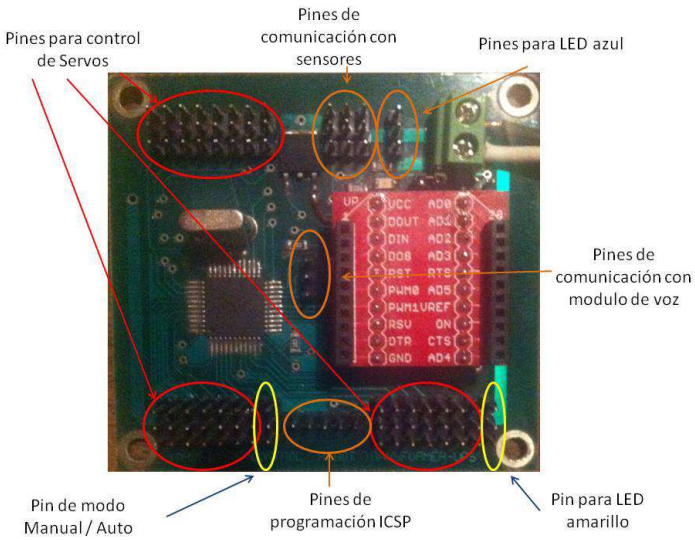


Figura 3.11: Distribución de pines de la tarjeta de control

Fuente: Los autores

3.3 Diseño del control remoto inalámbrico

El control remoto será el encargado de permitir comandar a los robots de forma inalámbrica a una distancia no mayor a los 25mts en línea de vista. Consta de una pantalla grafica y un panel táctil para su manipulación como se muestra en la figura 3.12.

Para su diseño, se parte desde un prototipo desarrollado en ISIS PROTEUS, el cual permite simular y desarrollar pruebas de comunicación serial mediante la herramienta Virtual Terminal de ISIS, logrando así simular la transmisión y recepción de datos inalámbricos como si se estuviera utilizando un XBEE.

Cabe recalcar que en la simulación se omite el circuito de control para el panel táctil, debido a que simplemente con dos potenciómetros se puede simular la variación de resistencia de cada panel para cada eje.

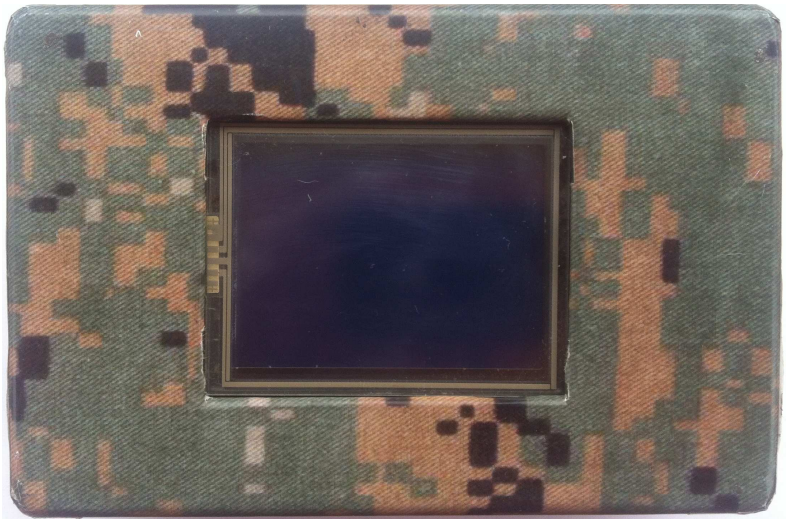


Figura 3.12: Control remoto

Fuente: Los autores

En la figura 3.13 se aprecia el diagrama esquemático del control remoto.

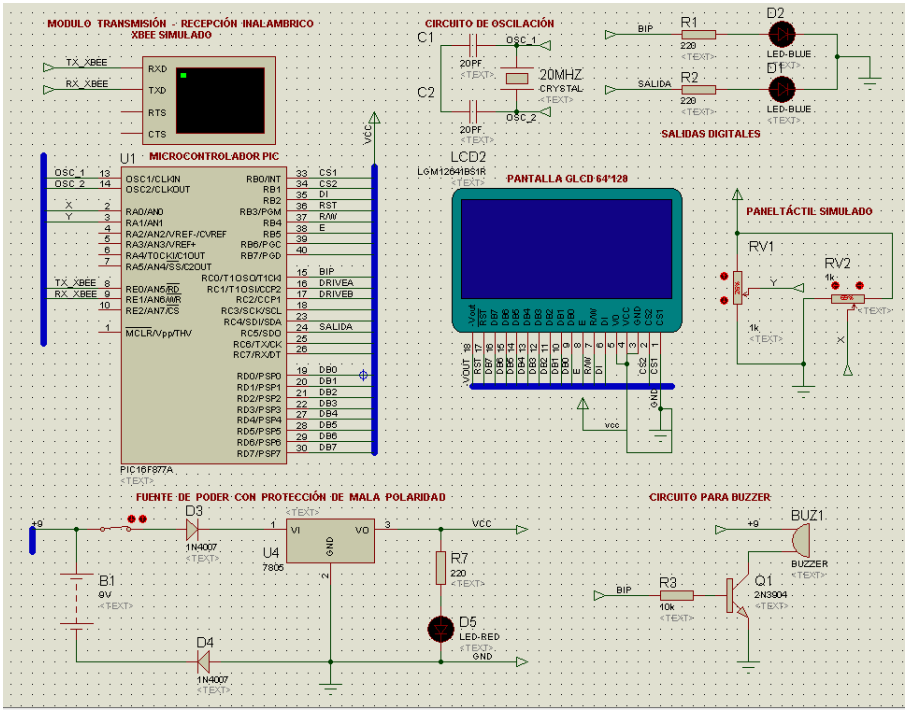


Figura 3.13: Diagrama esquemático del circuito del control remoto

Fuente: Los autores

3.3.1 Configuración de conexiones de la tarjeta del control remoto

La configuración nos indica las conexiones realizadas entre el microcontrolador PIC y los elementos externos como la pantalla GLDC, el panel táctil, el módulo inalámbrico XBEE, tal como se aprecia en la figura 3.14.

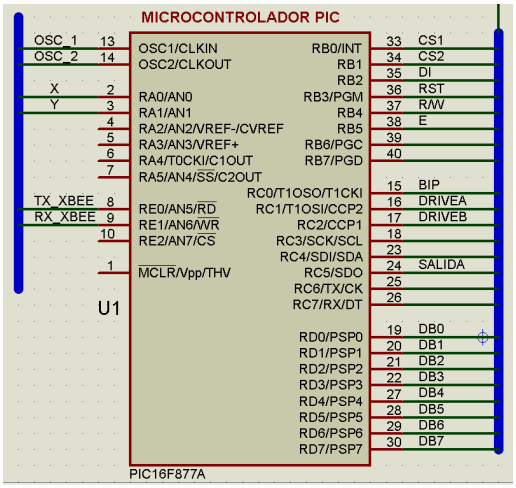


Figura 3.14: Configuración de conexiones del microcontrolador del control remoto

Fuente: Los autores

Al igual que en la tarjeta de control principal, la simulación está desarrollada con un microcontrolador PIC 16F877A con encapsulado tipo DIP, pero en la práctica se utiliza el mismo microcontrolador pero en encapsulado QFP. Como consecuencia tenemos que los números de pines no coinciden, pero el nombre y número del puerto si (Puerto B7 es el mismo tanto para DIP como SMD, únicamente cambia el pin físico designado), como se puede apreciar en la tabla 6.

PUERTO	DIP	SMD	DESCRIPCIÓN
A0	2	19	Entrada de señal análoga del eje X del panel
A1	3	20	Entrada de señal análoga del eje Y del panel
A2	4	21	Pin no usado
A3	5	22	Pin no usado
A4	6	23	Pin no usado

A5	7	24	Pin no usado
B0	33	8	Pin de control CS1
B1	34	9	Pin de control CS2
B2	35	10	Pin de control DI
B3	36	11	Pin de control RST
B4	37	14	Pin de control RW
B5	38	15	Pin de control E
B6	39	16	Pin no usado
B7	40	17	Pin no usado
C0	15	32	Pin de salida para buzzer
C1	16	35	Pin de salida DRIVER A para control de panel
C2	17	36	Pin de salida DRIVER B para control de panel
C3	18	37	Salida digital libre
C4	23	42	Salida digital libre
C5	24	43	Salida digital libre
C6	25	44	Salida digital libre
C7	26	1	Salida digital libre
D0	19	38	Salida de dato D0 para pantalla gráfica
D1	20	39	Salida de dato D1 para pantalla gráfica
D2	21	40	Salida de dato D2 para pantalla gráfica
D3	22	41	Salida de dato D3 para pantalla gráfica
D4	27	2	Salida de dato D4 para pantalla gráfica
D5	28	3	Salida de dato D5 para pantalla gráfica
D6	29	4	Salida de dato D6 para pantalla gráfica
D7	30	5	Salida de dato D7 para pantalla gráfica
E0	8	25	Pin de transmisión TX hacia el Xbee
E1	9	26	Pin de recepción RX hacia el Xbee
E2	10	27	Pin no usado
/MCLR	1	18	Pin de reset conectado a VCC general

Tabla 6: Distribución de pines en DIP y SMD

Fuente: Los autores

3.3.2 Diseño de la tarjeta del control remoto en ARES de PROTEUS

Una vez simulado el esquemático del circuito del control remoto, se procede a la realización del diseño de su tarjeta física para luego ser probado de forma real. Para esto se utiliza la herramienta ARES de PROTEUS para llevar a cabo el diseño de la tarjeta.

En la figura 3.15 se muestra el diseño de la tarjeta del control remoto y en la figura 3.16 el aspecto físico de la misma en 3D.

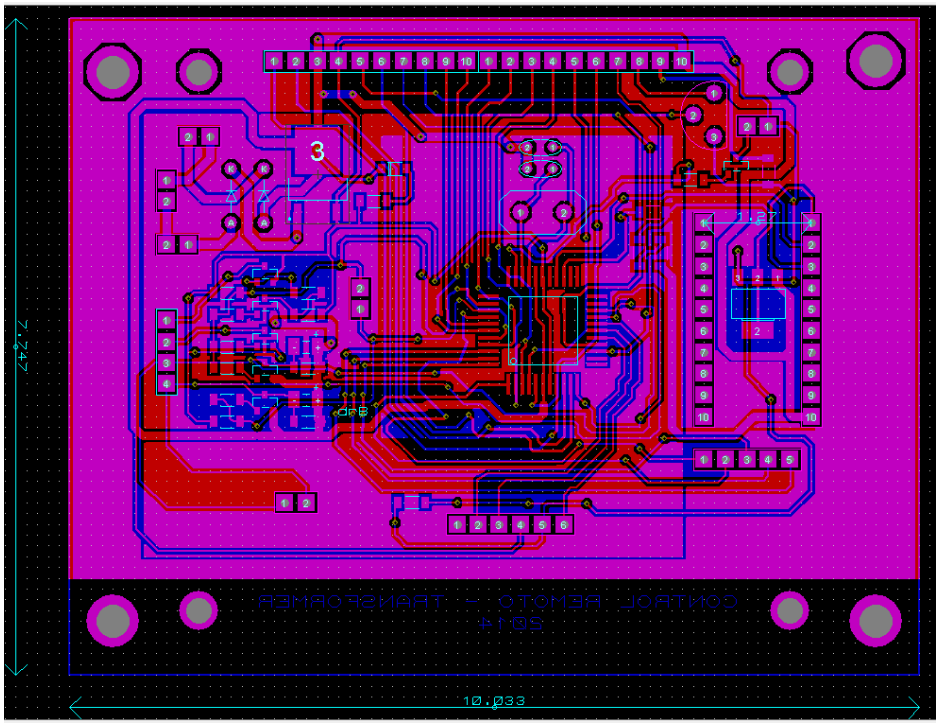


Figura 3.15: Diseño de la tarjeta del control remoto en ARES PROTEUS

Fuente: Los autores

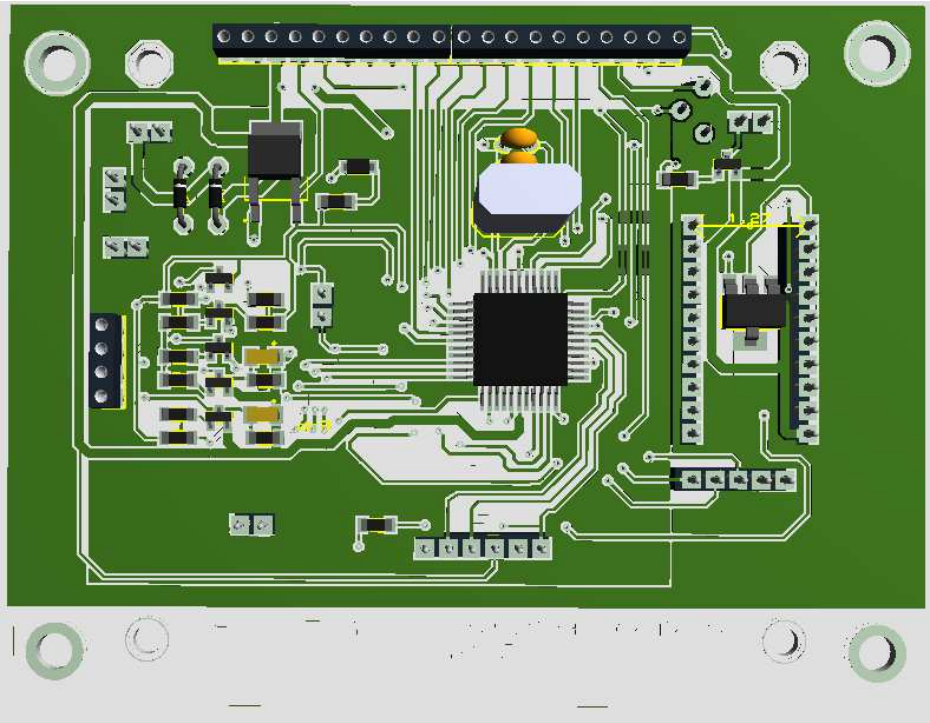


Figura 3.16: Aspecto físico en 3D de la tarjeta del control remoto

Fuente: Los autores

3.3.3 Descripción de la tarjeta del control remoto

Esta tarjeta es la encargada de enviar las órdenes de control para que el robot las recepte, interprete y las ejecute. Así mismo el robot enviará comandos de respuesta para saber que el enlace de comunicación no se ha perdido, logrando así una comunicación constante y segura.

La tarjeta del control remoto consta de las siguientes partes:

- Fuente de poder principal
- Fuente de poder secundaria
- Microcontrolador principal
- Etapa transmisora – receptora inalámbrica
- Pines para pantalla GLCD
- Pines para panel táctil

- Pines I/O libres
- Etapa transistorizada para control de panel táctil y acople con microcontrolador principal

En la figura 3.17 se muestra el aspecto físico de la tarjeta de control remoto.

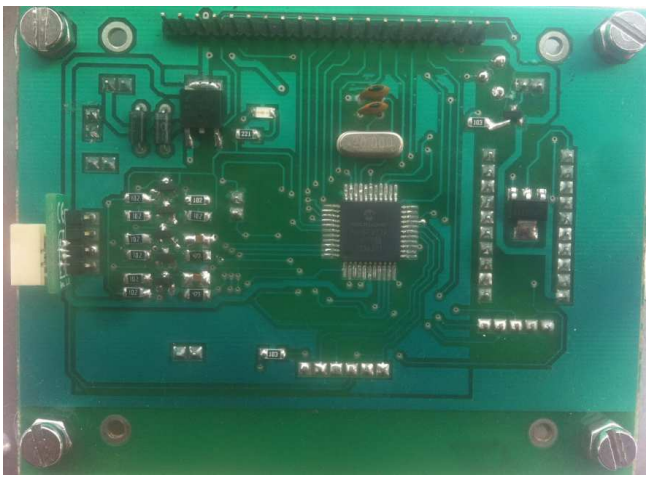


Figura 3.17: Aspecto físico de la tarjeta del control remoto

Fuente: Los autores

3.3.3.1 Fuente de poder principal del control remoto

Similar a la tarjeta de control principal, está formada por un regulador de voltaje de +5V, el famoso LM7805 pero en encapsulado SMD.

La diferencia entre estas dos fuentes es que la del control remoto posee un circuito de protección para mala polaridad mientras que la del control principal no la posee.

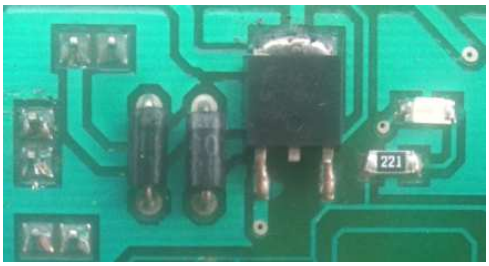


Figura 3.18: Fuente de poder principal conformada por el regulador LM7805 SMD y diodos de protección para mala polaridad

Fuente: Los autores

En la figura 3.19 podemos ver el esquemático realizado en ISIS PROTEUS.

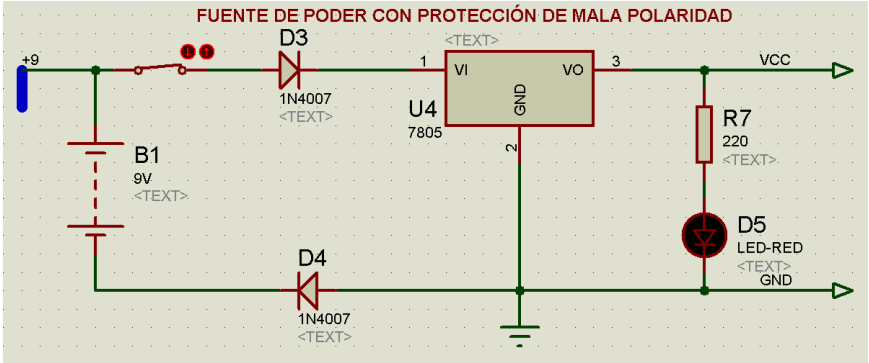


Figura 3.19: Esquemático de la fuente de poder principal del control remoto realizada en ISIS PROTEUS

Fuente: Los autores

3.3.3.2 Fuente de poder secundaria del control remoto

Similar a la fuente de la tarjeta de control principal, consta de un regulador de voltaje AMS1117-3.3 de +3.3 utilizado para la alimentación del módulo de transmisión y recepción inalámbrica.

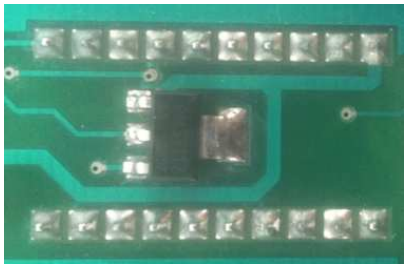


Figura 3.20: Fuente de poder secundaria de la tarjeta del control remoto conformada por el regulador AMS1117-3.3 SMD

Fuente: Los autores

3.3.3.3 Microcontrolador principal del control remoto

Este microcontrolador es el encargado de gestionar y ejecutar todo el algoritmo de programación para controlar la pantalla GLCD, el panel táctil, el módulo XBEE y las salidas digitales como la salida del buzzer.

Figura 3.21: Microcontrolador principal PIC 16F877A-QFT

Fuente: Los autores

3.3.3.4 Etapa transmisora – receptora inalámbrica

Esta etapa está conformada únicamente por el módulo de comunicación inalámbrica Xbee.

Figura 3.22: Módulo de transmisión inalámbrica XBEE sobre la tarjeta de control remoto

Fuente: Los autores

3.3.3.5 Pines para conexión de pantalla GLCD

Estos pines son usados para conectar la pantalla gráfica o GLCD de 64*128 caracteres.

En total son 20 pines, empezando desde la izquierda con el pin #1 y terminando en la derecha con el pin #20.

Figura 3.23: Pines para pantalla gráfica o GLCD de 64*128 monocromática
Fuente: Los autores

3.3.3.6 Pines para conexión del panel táctil

Estos pines están soldados sobre una placa que contiene el conector para el panel táctil resistivo.

Figura 3.24: Conector para panel táctil resistivo
Fuente: Los autores

3.3.3.7 Pines I/O libres

Estos pines son para propósito general, no cumplen ninguna función que tenga relación con el robot o con el control remoto pero se los puede utilizar en aplicaciones futuras. Su configuración empieza desde la izquierda con el pin C7 y

termina en la derecha con el pin C3, es decir los pines para uso general son C3-C4-C5-C6-C7.

Figura 3.25: Pines para uso general

Fuente: Los autores

3.3.3.8 Etapa transistorizada para control de panel táctil y acople con microcontrolador principal

Esta etapa permite enviar señales digitales para control del panel táctil y recibir señales analógicas (de 0.0 a 5.0V en todo su rango) que posteriormente el microcontrolador las interpretara para saber la coordenada en el eje X y en el eje Y.

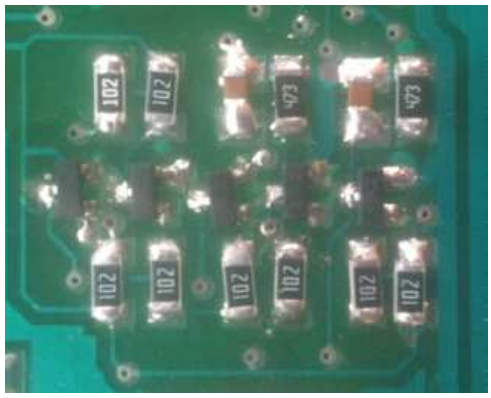


Figura 3.26: Etapa transistorizada para control de panel táctil

Fuente: Los autores

En las figuras 3.27 y 3.28 se puede apreciar una vista general de la ubicación de cada parte que conforman a la tarjeta del control remoto.

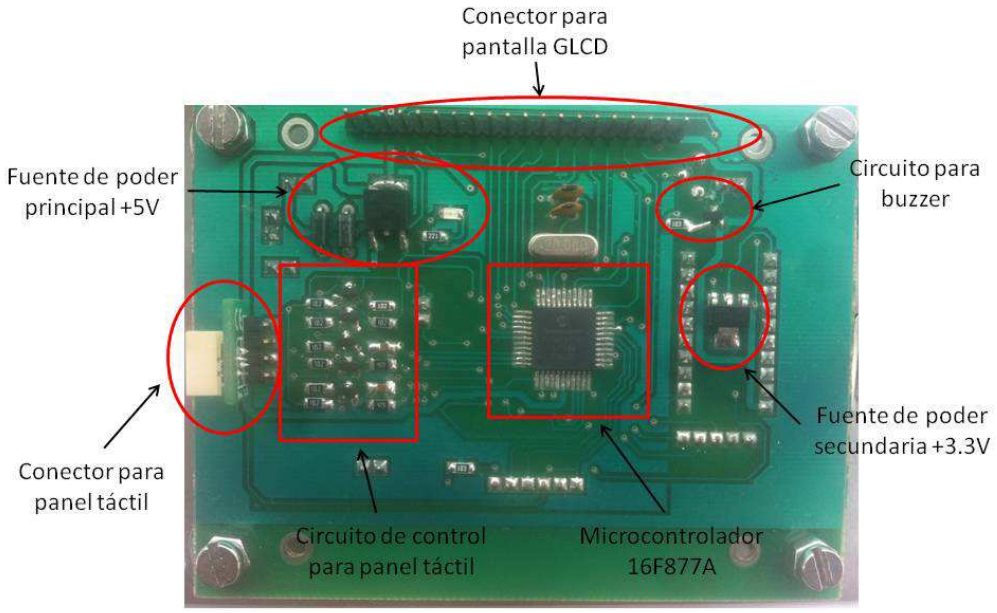


Figura 3.27: Descripción de la vista superior de la tarjeta de control remoto

Fuente: Los autores

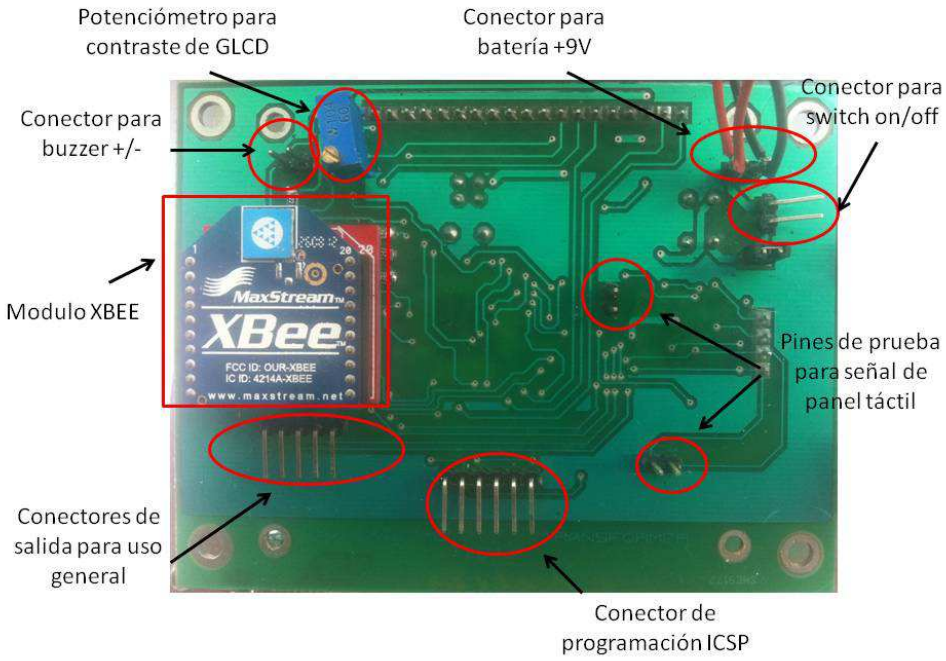


Figura 3.28: Descripción de la vista inferior de la tarjeta de control remoto

Fuente: Los autores

3.3.4 Funcionamiento del control remoto

El control remoto funciona de forma bidireccional, es decir, así como transmite información hacia el robot, también recibe información desde el robot. Esto lo realiza siempre y cuando el robot se encuentre en modo manual.

El funcionamiento del control remoto se lo divide en cuatro secciones, las cuales son.

- Inicialización
- Menú principal
- Transformaciones
- Sensores

Cada sección envía o recibe un tipo de información diferente, esta información son los comandos que el robot tendrá que ejecutar o los estados que el robot enviará hacia el control para no perder comunicación o saber si se la ha perdido (cuando no se recepta información).

3.3.4.1 Inicialización

Para la inicialización, se procede primero a encender el control remoto y luego el robot, ya que el robot inicialmente necesita enviar “códigos basuras” para inicializar el Xbee y poder luego realizar un buen enlace entre ambos.

La primera pantalla que muestra el control remoto una vez encendido se muestra en la figura 3.29.

Figura 3.29: Primer pantallazo al encender el control remoto

Fuente: Los autores

Una vez encendido el control, éste esperará un código de inicialización que el robot le envía para saber que se encuentra listo para ser utilizado. Si el código nunca llega hasta el control, significa que existe alguna falla en la inicialización o simplemente el robot no fue encendido correctamente, como resultado el control remoto se "congelará" en la pantalla de inicialización esperando la confirmación del robot como se muestra en la figura 3.30.

Figura 3.30: Pantalla de espera de señal de inicialización por parte del robot

Fuente: Los autores

Una vez que se reciba la señal de inicialización por parte del robot, el control remoto la detectará e indicará que el robot se encuentra listo para ser utilizado como se muestra en la figura 3.31.

Figura 3.31: Mensaje indicando que existe comunicación entre el control remoto y el robot

Fuente: Los autores

3.3.4.2 Menú principal

Una vez que se haya realizado la comunicación inicial entre el control remoto y el robot, se presentará en el control remoto la pantalla del menú principal, la cual indicará las opciones o instrucciones que el robot puede realizar. Las opciones que el menú principal muestra son las siguientes:

- **Auto:** Transforma el robot de “Humano” a “Auto”, siempre y cuando el robot se encuentre en estado “Humanoide”
- **Humano:** Transforma el robot de “Auto” a “Humano”, siempre y cuando el robot se encuentre en estado “Auto”
- **Sensores:** Le pide al robot que envíe información sobre la temperatura ambiente y la distancia entre un objeto y él

En la figura 3.32 podemos ver el menú principal que muestra el control remoto.

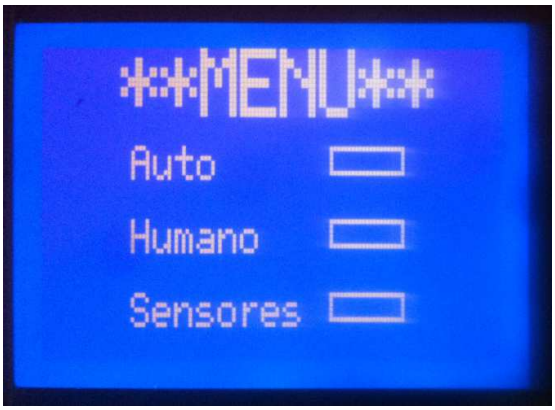


Figura 3.32: Menú principal

Fuente: Los autores

3.3.4.3 Transformaciones

Consiste en enviar los comandos necesarios para efectuar las transformaciones deseadas. Una vez transformado el robot, el control remoto muestra un submenú con varias opciones según sea el tipo de la transformación realizada y, el robot, a su vez envía una confirmación de forma inalámbrica hacia el control para que éste sepa que la transformación se ejecutó en su totalidad sin ningún inconveniente.

En esta etapa de transformaciones, los algoritmos de programación que están dentro del robot y del control remoto trabajan a la par mediante la comunicación inalámbrica, ya que siempre se debe tener presente el tipo de transformación actual debido a que cada transformación está enlazada con varias subrutinas para llegar a la transformación final, y si no se respeta ese orden, se podrían efectuar daños en los servomotores del robot por excesos de fuerza o golpes. En forma de ejemplo se puede decir que para que el robot se transforme de auto a humano, éste debe estar primero en forma de auto y no en la de un humano, de lo contrario ejecutaría otras subrutinas correspondientes a un estado erróneo del robot, perjudicándolo en su totalidad.

3.3.4.4 Tipos de Transformaciones

Las transformaciones que el robot puede realizar son:

- De modo Humano a Auto y de modo Auto a Humano (Humanoide)
- De modo Auto a Hexápodo y de modo Hexápodo a Auto (Hexápodo)

3.3.4.4.1 Transformación a modo auto

En esta etapa el robot pasa de estar en modo “Humano” a modo “Auto”. En la figura 3.33 se muestra la pantalla del submenú en modo auto.

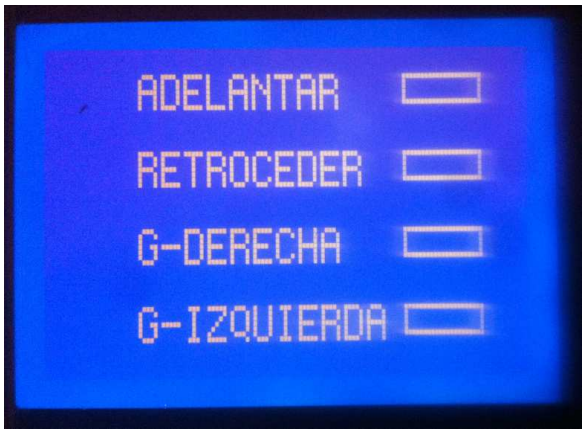


Figura 3.33: Submenú del modo auto

Fuente: Los autores

Como se puede observar, los comandos que el robot puede ejecutar son:

- **Adelantar:** Hace mover al robot hacia adelante
- **Retroceder:** Mueve el robot hacia atrás
- **G-Derecha:** Hace girar al robot hacia la dirección derecha
- **G-Izquierda:** Hace girar al robot hacia la dirección izquierda

3.3.4.4.2 Transformación a modo humano

En esta etapa, el robot debe estar en modo “auto” inicialmente para luego pasar a modo “humano”. En la figura 3.34 se muestra el submenú en modo humano.

Figura 3.34: Submenú del modo humano

Fuente: Los autores

Como se puede observar en la figura 3.34, los comandos que el robot puede ejecutar son:

- **Caminar:** Envía el comando para que el robot camine hacia adelante.

Al presionar en la pantalla sobre “el botón” menú, éste regresa al menú principal y el robot se alista para recibir las órdenes anteriores (auto, humano, sensores).

3.3.4.5 Sensores

En ésta instrucción, se le solicita al robot que envíe los datos que posee actualmente en sus dos sensores (temperatura y distancia), teniendo como resultado la lectura de los mismos en la pantalla GLCD como se muestra en la figura 3.35.

Figura 3.35: Lectura de datos de los sensores de temperatura y distancia

Fuente: Los autores

3.3.5 Instrucciones enviadas entre el control remoto y el robot

La comunicación entre el control remoto y el robot se la realiza inalámbricamente mediante el módulo de transmisión y recepción inalámbrica XBEE, al cual se le envían datos seriales por el microcontrolador. Estos datos seriales son las instrucciones que el robot las interpreta para ejecutar una rutina y, así mismo, el robot envía de forma serial datos que el control remoto interpreta para saber su estado actual o la información de los sensores en caso de solicitarlo.

Las instrucciones enviadas se las muestra en la tabla 7.

Orden de envío	**Comando**	**Acción**
Robot – Control	“2”	Indica que el robot está listo para ser utilizado
Control – Robot	“4”	Transformación de Auto a Humano
Robot – Control	“1”	Recepción de dato satisfactoriamente
Control – Robot	“5”	Transformación de Humano a Auto
Control – Robot	“6”	Pide datos de los sensores

Control – Robot	“3”	Movimiento hacia adelante en modo auto
Control – Robot	“7”	Movimiento hacia atrás en modo auto
Control – Robot	“8”	Movimiento hacia la derecha en modo auto
Control – Robot	“9”	Movimiento hacia la izquierda en modo auto
Control – Robot	“1”	Caminar en modo humano (sub menú)
Control – Robot	“0”	Salida a menú principal
Robot - Control	Dato int8	Valor de variables de temperatura y distancia

Tabla 7: Comandos seriales e instrucciones que se envían bidireccionalmente entre el robot y el control remoto

Fuente: Los autores

3.3.6 Diagrama de flujo del funcionamiento del control remoto

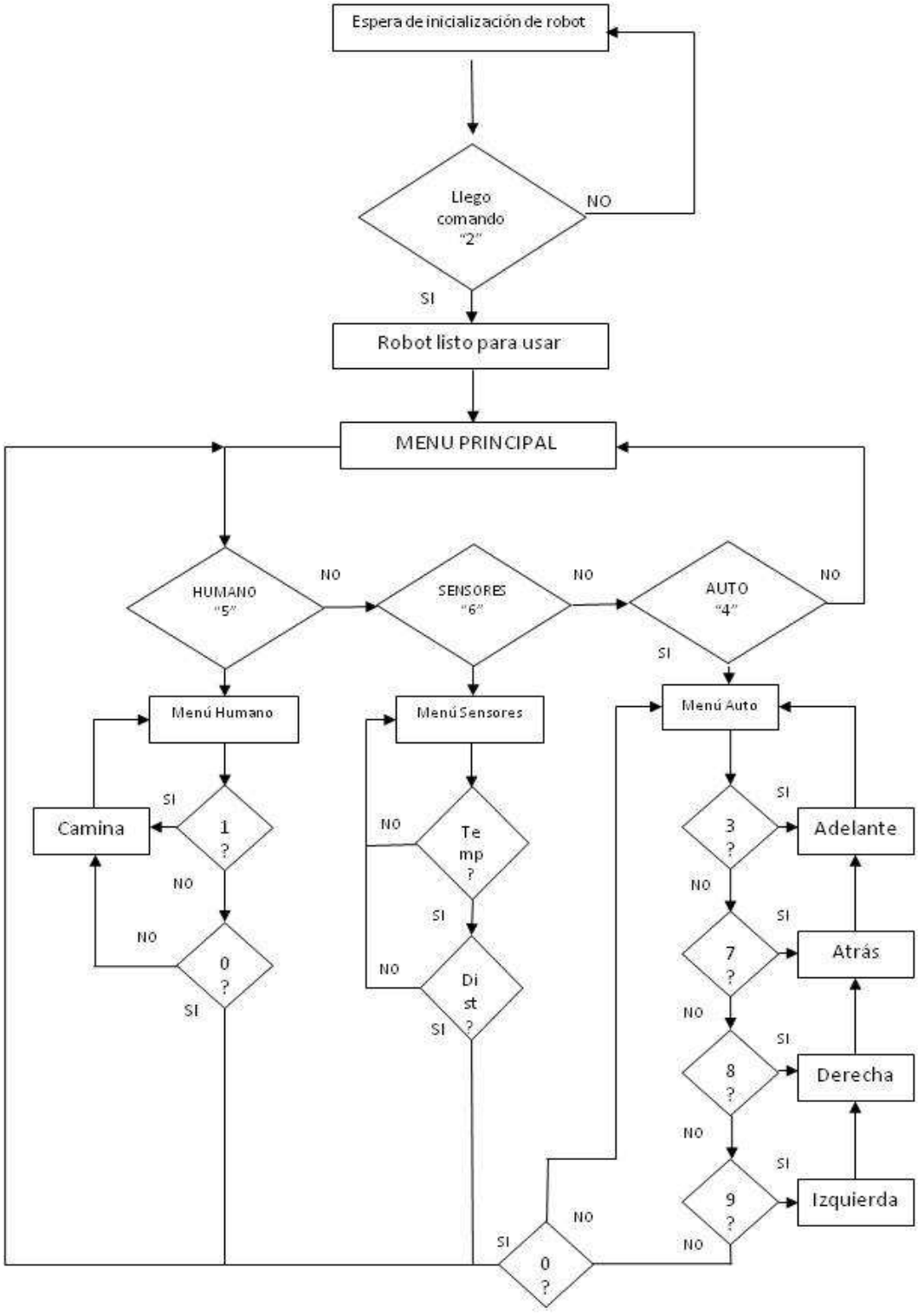


Figura 3.36: Diagrama de flujo del funcionamiento del control remoto

Fuente: Los autores

3.4 Diseño del circuito para el control de los sensores

Al igual que en ocasiones anteriores, se parte desde un diagrama esquemático realizado en ISIS PROTEUS para guiarnos en la simulación del circuito antes de su realización en forma real y final. En la figura 3.37 se puede apreciar el esquemático del circuito para el control de sensores.

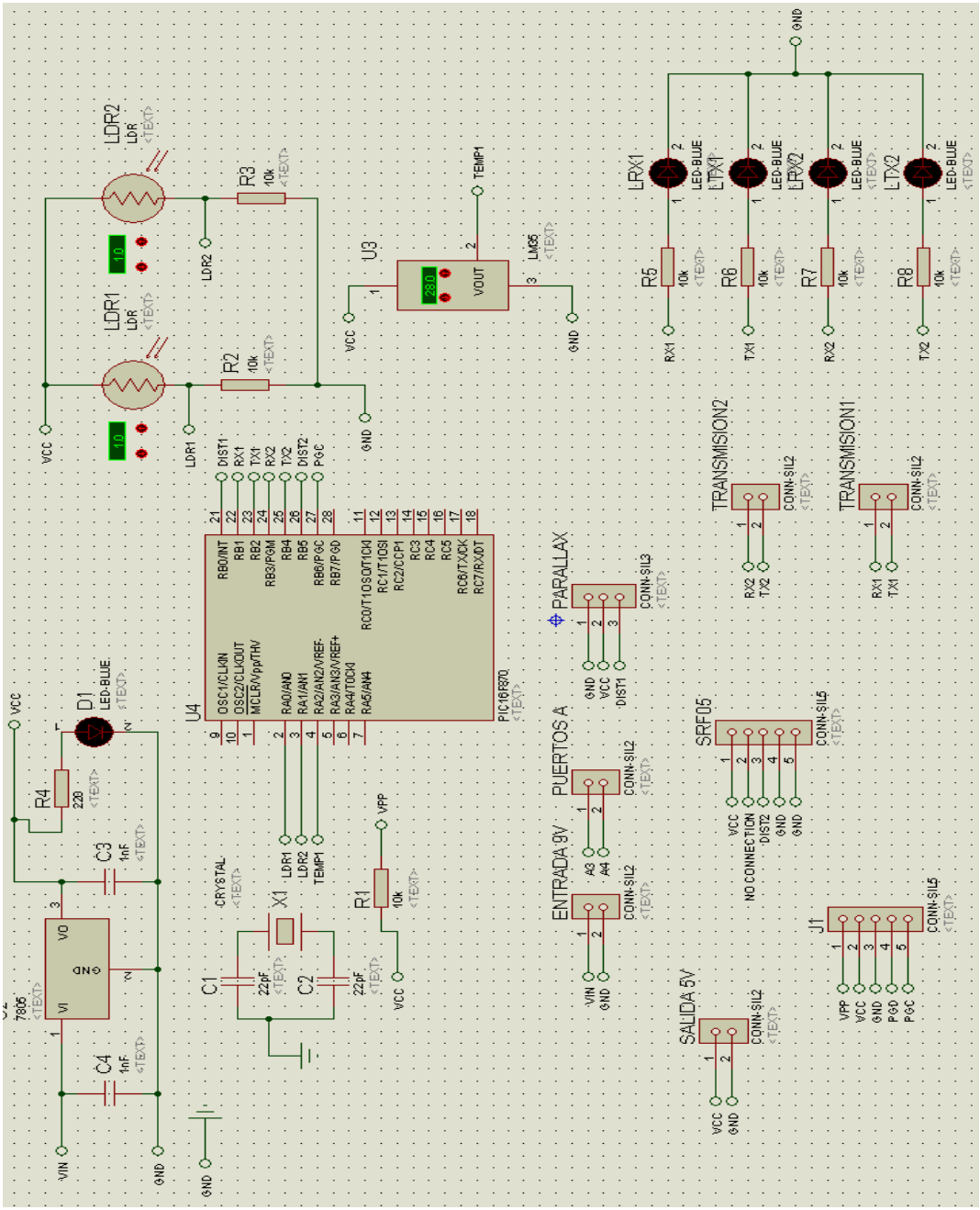


Figura 3.37: Esquemático del circuito para control de sensores

Fuente: Los autores

3.4.1 Configuración de conexiones del circuito de control de sensores

Conocer la configuración de conexiones ayuda a identificar los pines de unión entre el microcontrolador, los sensores y las líneas de transmisión y recepción para la comunicación con el microcontrolador principal tal como se aprecia en la figura 3.38.

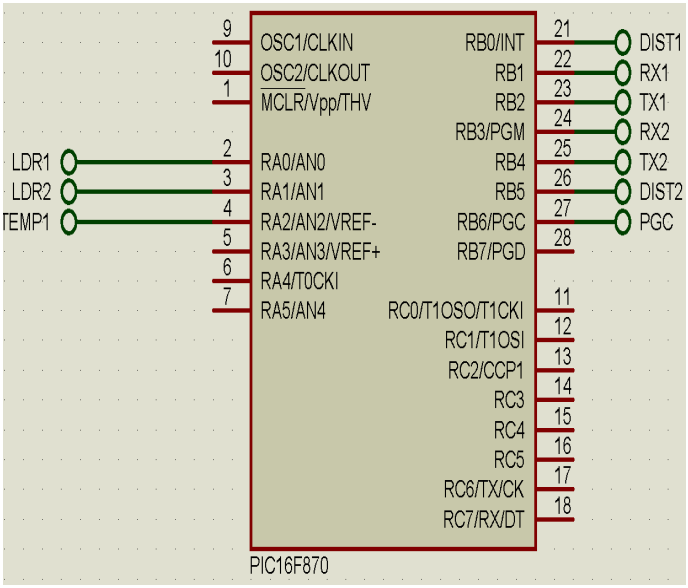


Figura 3.38: Configuración de pines del microcontrolador secundario PIC16F870

Fuente: Los autores

En este caso, la configuración del microcontrolador PIC16f870 en encapsulado DIP, es la misma que en encapsulado SMD, por lo tanto los mismos pines que se utilizan para la simulación se los utilizara para el ensamblaje final.

PUERTO	PIN	DESCRIPCIÓN
A0	2	Pin para conexión de LDR1
A1	3	Pin para conexión de LDR2
A2	4	Pin para sensor de temperatura LM35
B0	21	Pin para sensor de distancia SRF05

B1	22	Pin para recepción de datos 1
B2	23	Pin para transmisión de datos 1
B3	24	Pin para recepción de datos 2
B4	25	Pin para transmisión de datos 2
B5	26	Pin para sensor de distancia 2
B6	27	Pin libre para uso general
B7	28	Pin libre para uso general
C0-C7	11-18	Pines libres para uso general
/MCLR	1	Conectado a la alimentación positiva o VCC

Tabla 8 Distribución de pines en DIP y SMD del microcontrolador secundario

Fuente: Los autores

3.4.2 Diseño de la tarjeta para el control de sensores en ARES PROTEUS

Una vez simulado el esquemático del circuito para el control de los sensores del robot, se procede a la realización del diseño de su tarjeta física para luego ser probado de forma real. Para llevar a cabo el diseño de la tarjeta se utiliza la herramienta ARES de PROTEUS.

En la figura 3.39 se muestra el diseño de la tarjeta del circuito para el control de los sensores del robot y en la figura 3.40 el aspecto físico de la misma en 3D.

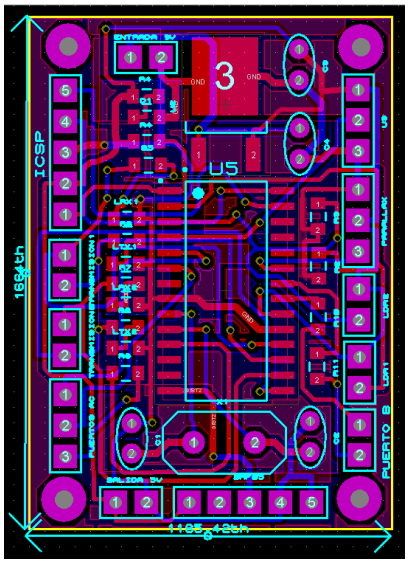


Figura 3.39: Diseño de la tarjeta para el control de los sensores del robot en ARES

Fuente: Los autores

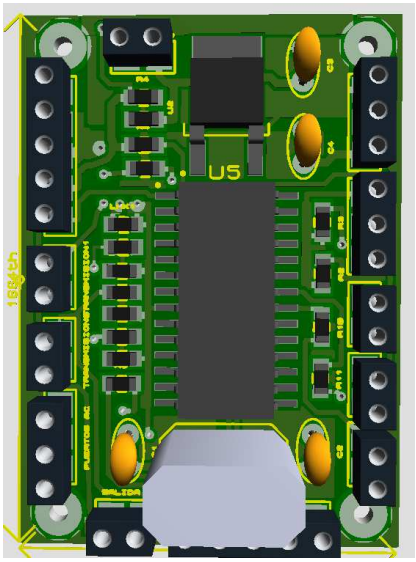


Figura 3.40: Aspecto físico en 3D de la tarjeta para el control de los sensores del robot

Fuente: Los autores

3.4.3 Descripción general de la tarjeta de control de sensores

Esta tarjeta está diseñada para servir de "puente de comunicación" entre los sensores y el microcontrolador principal. Su desarrollo nace por la carencia de

números de puertos sobre la tarjeta de control principal (para el robot Hexápodo), pero teniendo como principal ventaja que se dedica específicamente un algoritmo para que se encargue de la comunicación con los sensores, pero, su desventaja es que ocupa un espacio más sobre el robot.

3.4.4 Partes que conforman a la tarjeta de control de sensores

Esta tarjeta está conformada por las siguientes partes:

- Microcontrolador secundario
- Fuente de poder
- Pines para sensores y comunicación

En la figura 3.41 se muestra el aspecto físico de la tarjeta de control de los sensores para el robot.

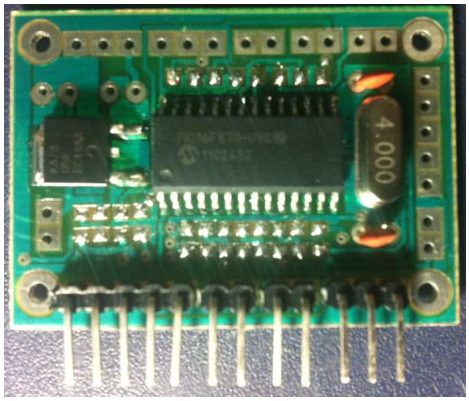


Figura 3.41: Aspecto físico de la tarjeta de control para los sensores del robot
Fuente: Los autores

3.4.4.1 Microcontrolador secundario

Está conformado por un microcontrolador PIC16F870 en encapsulado DIP-SMD, es el cerebro de esta tarjeta, ya que es el encargado de realizar la gestión necesaria entre la comunicación de los sensores con el microcontrolador principal PIC 16F877A.

Figura 3.42: Microcontrolador secundario PIC16F870

Fuente: Los autores

3.4.4.2 Fuente de poder

Al igual que las anteriores, la conforma un regulador de voltaje de +5V, el conocido LM7805 en encapsulado SMD.

Figura 3.43: Regulador de voltaje LM7805-SMD en la tarjeta de control de sensores

Fuente: Los autores

3.4.4.3 Pines para sensores y comunicación

Estos pines son los encargados de recibir las señales de los sensores y de permitir la comunicación con el microcontrolador principal. Dicha comunicación se la realiza de forma serial directa sin necesidad de un módulo externo de comunicación serial.

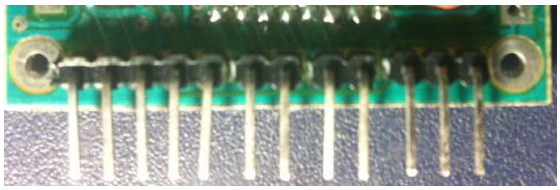


Figura 3.44: Pines de comunicación entre microcontrolador secundario, sensores y microcontrolador principal

Fuente: Los autores

3.5 Diseño del módulo de control para el robot hexápodo

Al igual que el módulo de control para el robot humanoide, se parte desde una simulación realizada en ISIS de PROTEUS como se muestra en la figura 3.45.

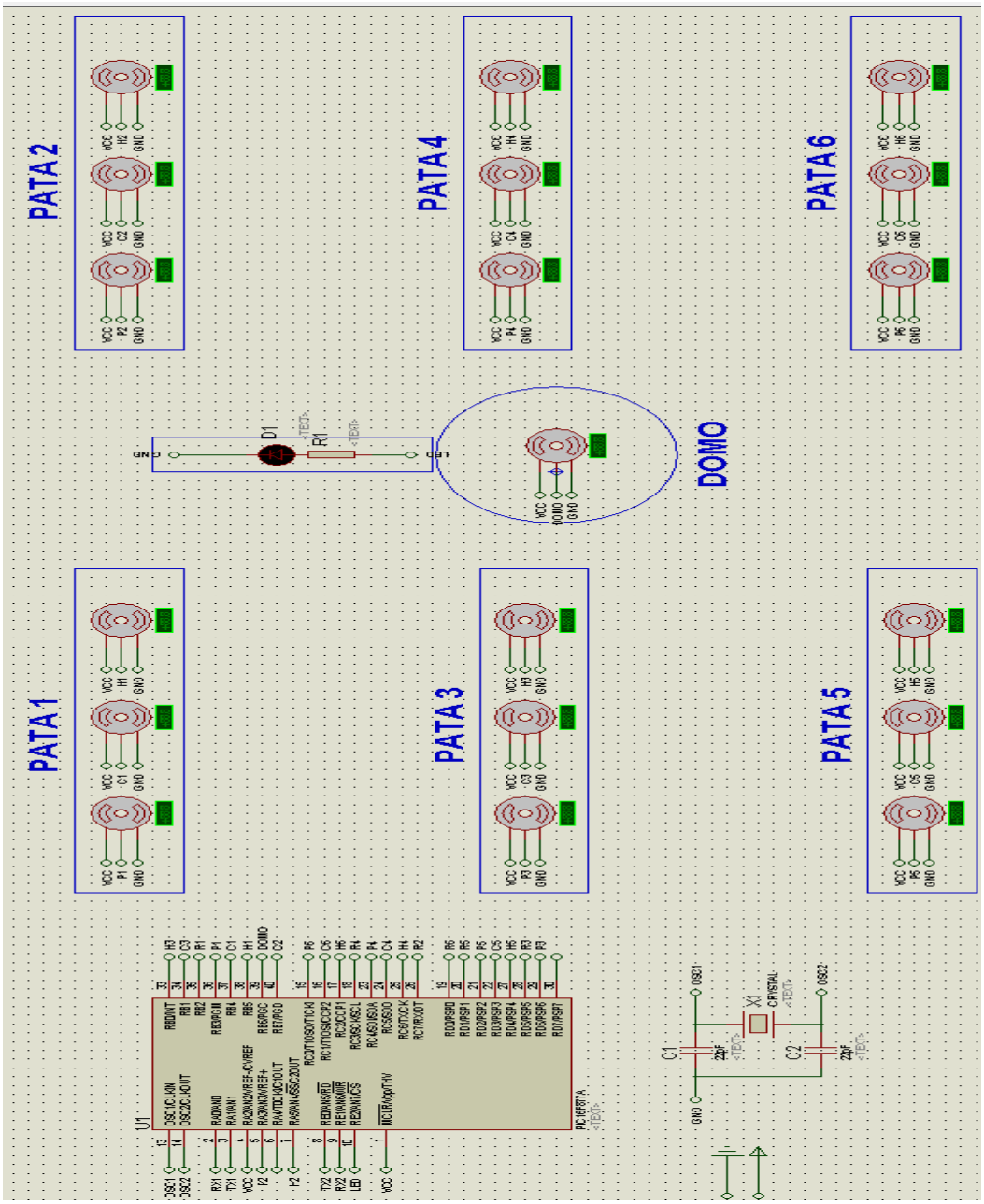


Figura 3.45: Esquemático del módulo de control del robot hexápodo simulado

Fuente: Los autores

3.5.1 Configuración de conexiones del módulo de control principal del robot hexápodo

Similar al módulo de control del robot humanoide, ésta configuración nos indica las conexiones realizadas entre el microcontrolador PIC y los elementos externos, como los servomotores, sensores, memoria de voz, tarjeta de sensores, entre otros elementos tal como se muestra en la figura 3.46.

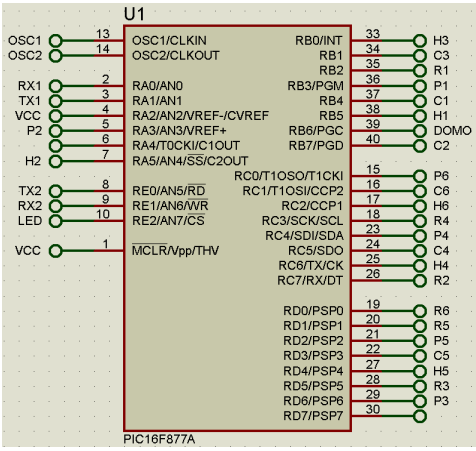


Figura 3.46: Configuración de conexiones del módulo principal del robot hexápodo

Fuente: Los autores

Cabe recalcar que al igual que el módulo anterior, este diseño está simulado con un microcontrolador PIC16F877A con encapsulado DIP, mientras que en el desarrollo final se utiliza el mismo microcontrolador pero en encapsulado QFT-SMD, por lo tanto la distribución de pines cambia como se muestra en la tabla 9.

PUERTO	DIP	SMD	DESCRIPCIÓN
A0	2	19	Entrada RX1 hacia tarjeta de sensores
A1	3	20	Salida serial TX1 hacia tarjeta de sensores
A2	4	21	No usado
A3	5	22	Servo pata 2
A4	6	23	No usado
A5	7	24	Servo hombro 2

B0	33	8	Servo hombro 3
B1	34	9	Servo codo 3
B2	35	10	Servo rueda 1
B3	36	11	Servo pata 1
B4	37	14	Servo codo 1
B5	38	15	Servo hombro 1
B6	39	16	Servo domo
B7	40	17	Servo codo 2
C0	15	32	Servo pata 6
C1	16	35	Servo codo 6
C2	17	36	Servo hombro 6
C3	18	37	Servo rueda 4
C4	23	42	Servo pata 4
C5	24	43	Servo codo 4
C6	25	44	Servo hombro 4
C7	26	1	Servo rueda 2
D0	19	38	Servo rueda 6
D1	20	39	Servo rueda 5
D2	21	40	Servo pata 5
D3	22	41	Servo codo 5
D4	27	2	Servo hombro 5
D5	28	3	Servo rueda 3
D6	29	4	Servo pata 3
D7	30	5	No usado
E0	8	25	Pin de transmisión TX hacia el Xbee
E1	9	26	Pin de recepción RX hacia el Xbee
E2	10	27	Led indicador de para LEDS
/MCLR	1	18	Pin de RESET conectado a VCC general

Tabla 9: Diferencia entre pines de encapsulado DIP y QFT y función que realizan

Fuente: Los autores

3.5.2 Distribución de puertos de la tarjeta de control principal del robot hexápodo

Como posee un aspecto físico similar a la tarjeta de control del robot humanoide (tarjeta de control principal del robot humanoide es exactamente la misma que la del robot hexápodo), no se detallará la descripción ni la configuración de los pines de la misma (véase CAPÍTULO 3.2.3), simplemente se detallará la distribución de los puertos en dicha tarjeta como se muestra en la figura 3.47.

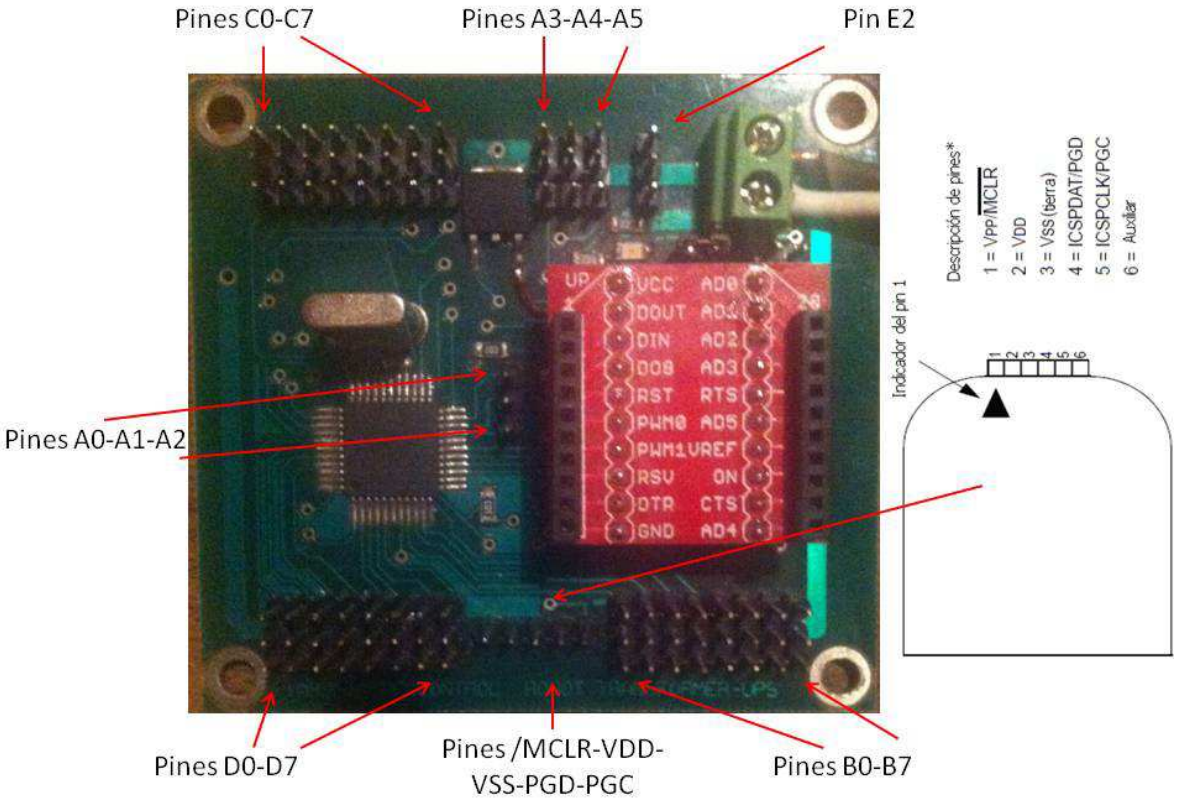


Figura 3.47: Configuración de puertos utilizados en la tarjeta de control principal

Fuente: Los autores

3.6 Diseño estructural del robot humanoide

3.6.1 Introducción

En esta sección se detallará el desarrollo de las piezas que conforman al robot transformer humanoide - auto, así como también a su base estructural principal y el armado de las mismas.

Cabe mencionar que en la parte del diseño del robot, se tomaron inicialmente características y experiencias de modelos de robots que se venden en el mercado internacional, pero, la estructura final se la consiguió partiendo de varios modelos diseñados por los autores, donde se descartaban piezas con las pruebas que se les realizaba como, motricidad, rigidez y estabilidad, donde se decidió por el modelo actual por tener una forma estructural muy sólida que facilita la motricidad de cada una de sus partes.

El transformer humanoide- auto, consta de 22 servomotores, los mismos que trabajaran a 6 voltios con una fuente independiente a la del control principal.

3.6.2 Base estructural principal

Primeramente se elabora la base estructural principal, la misma que representa al cuerpo del robot, donde estarán instalados los servomotores de los hombros, los de soporte para las ruedas superiores y el cuello del mismo. Ésta estructura fue realizada con aluminio de 2mm de espesor con sus respectivos cortes y agujeros para la instalación de cada servo.

Se decidió utilizar el aluminio como materia prima por sus características mecánicas como peso, durabilidad, fácil utilización y también por su excelente disipación de calor.

A continuación se detalla en la figura 3.48 las medidas correspondientes a la base estructural principal que representa al cuerpo del robot transformer Humanoide – Auto.

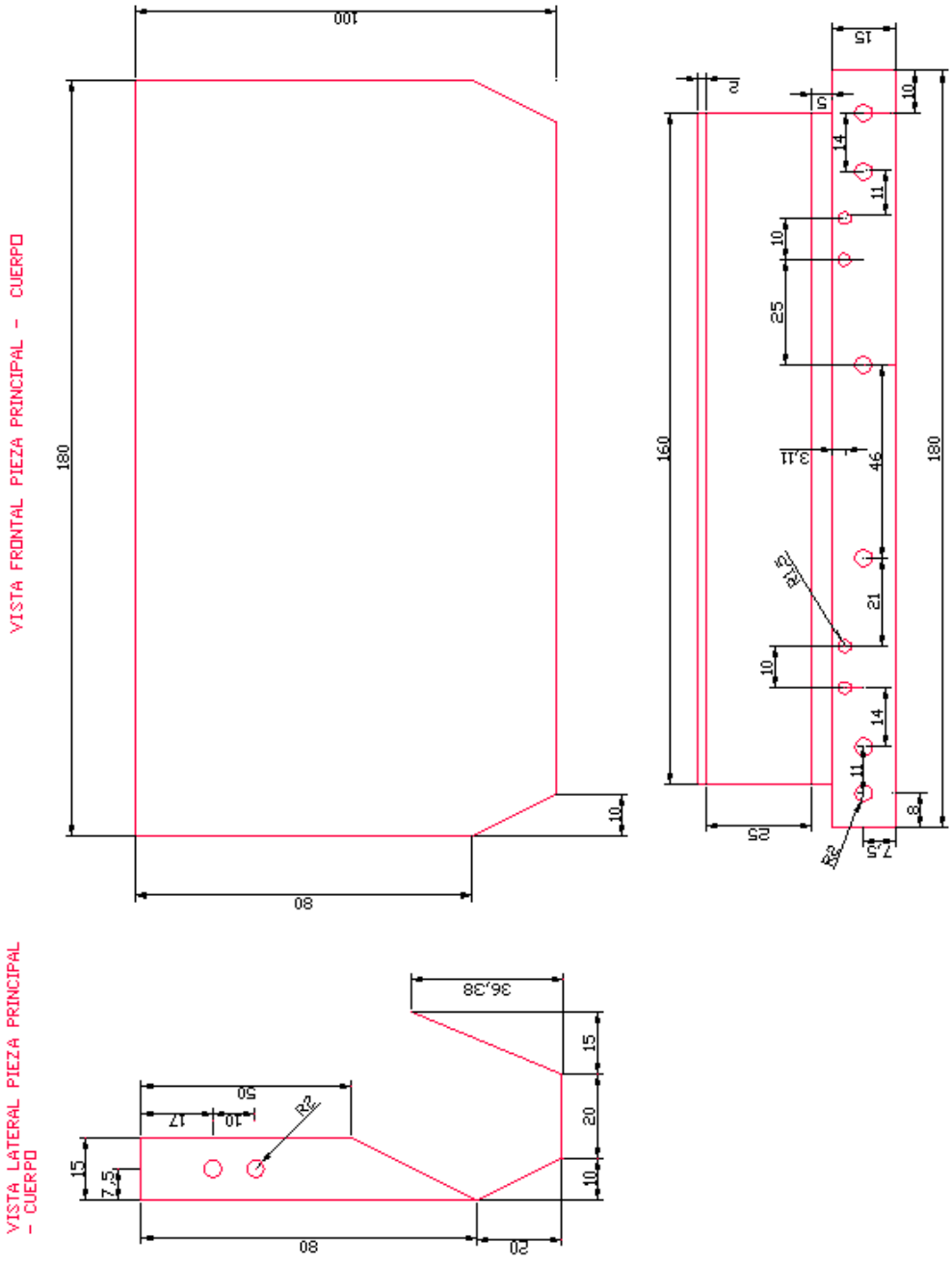


Figura 3.48: Vista frontal y lateral del cuerpo del robot humanoide - auto

Fuente: Los autores

Para la construcción de la misma se utilizaron herramientas como cortadoras, dobladora, taladro de pedestal con sus respectivas brocas y limas de metal para lograr su acabado final.

Una vez con la pieza terminada, se procede a forrarla con vinil blanco para darle el acabo final a la estructura como se muestra en la figura 3.49.

Figura 3.49: Acabado final de la estructura del cuerpo del robot humanoide - auto

Fuente: Los autores

Una vez desarrollada la estructura principal, se procede a realizar las piezas complementarias que ayudarán en la instalación de los demás servos, como los servos que sujetan a las ruedas superiores. En la figura 3.50 se detallan las medidas de estas piezas complementarias.

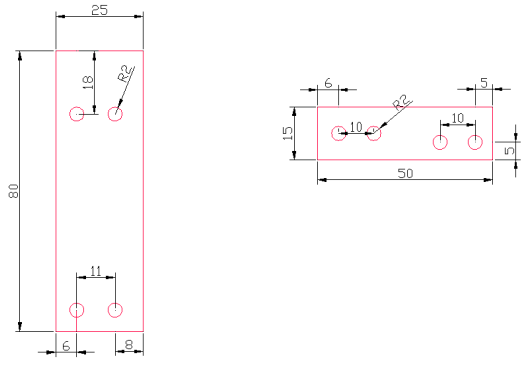


Figura 3.50: Piezas complementarias para instalación de servo para soporte de ruedas

Fuente: Los autores

Cabe recalcar que se realiza la misma forma tanto para el lado derecho como para el lado izquierdo y se las forra con vinil de color azul para su acabado final.

Para la instalación del servomotor del cuello, se realiza una pieza como se detalla en la figura 3.51.

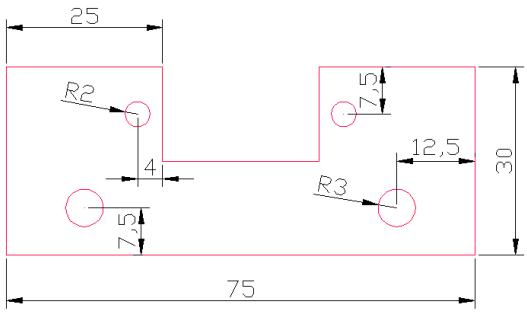


Figura 3.51: Piezas correspondiente para la instalación del cuello

Fuente: Los autores

En la figura 3.52 se aprecia la instalación de estas dos piezas sobre la estructura principal utilizando tornillos de acero inoxidable de 2mm y tornillos tripa de pato de 2mm,

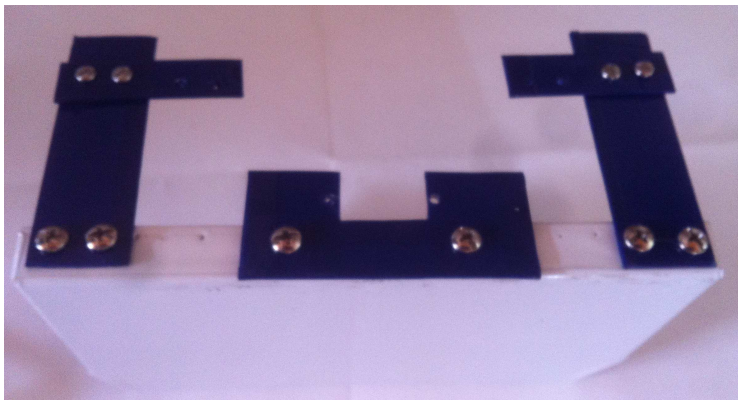


Figura 3.52: Piezas instaladas sobre la estructura principal

Fuente: Los autores

Una vez terminado el ensamble de la estructura correspondiente al cuerpo del robot, se procede a la realización de las piezas donde se instalarán los servomotores de las ruedas superiores del robot. En la siguiente figura se aprecian las medidas correspondientes a la pieza mencionada.

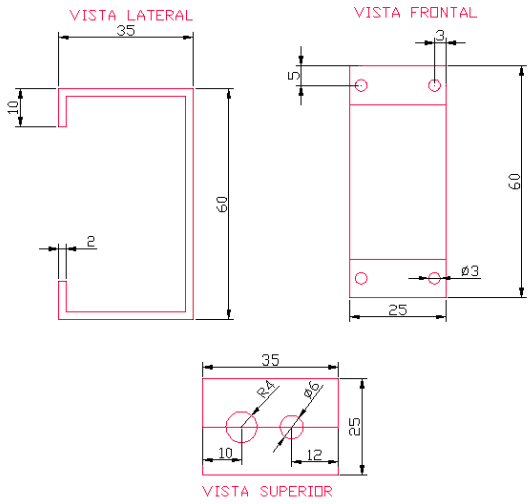


Figura 3.53: Piezas para instalación de ruedas superiores

Fuente: Los autores

Luego del desarrollo de esas piezas, se procede a instalarla sobre los servos que van en la estructura principal, los mismos que permitirán un giro para “enseñar” o “esconder” las ruedas.

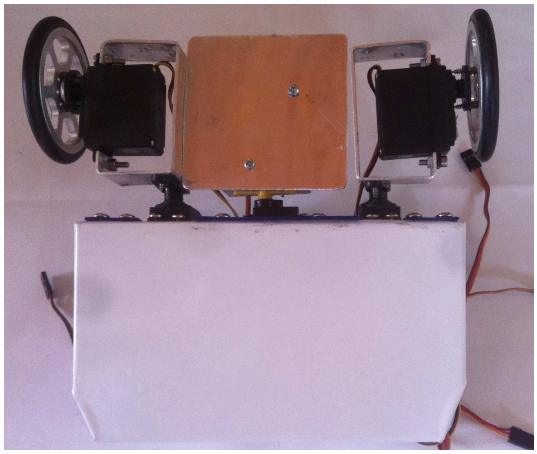


Figura 3.54: Piezas para giro de ruedas montadas sobre estructura principal

Fuente: Los autores

3.6.3: Desarrollo de piezas para las manos del robot humanoide

Para este desarrollo se considera la realización de tres piezas importante que juntas forman las manos del robot, estas piezas son: el hombro, el brazo y la mano.

Una vez instalada toda la pieza principal junto con sus piezas complementarias, viene el desarrollo de las piezas para la instalación de los hombros. Al igual que las piezas anteriores, ésta está desarrollada a base de aluminio de 2mm y forrada con vinil para darle el acabado final.

En la figura 3.55 se aprecian las medidas de la pieza correspondiente al hombro del robot. Se desarrolla la misma pieza tanto para el lado derecho como para el izquierdo, considerando únicamente la posición referencial de cada eje.

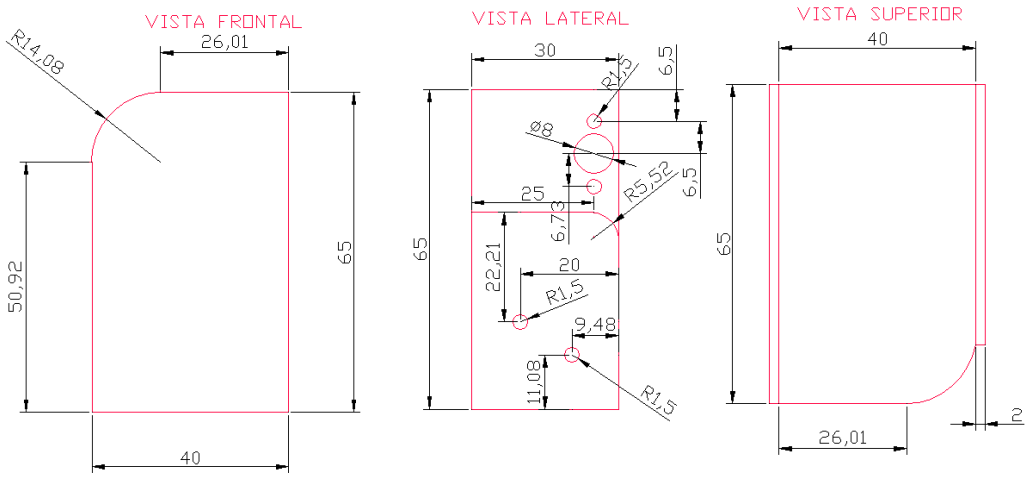


Figura 3.55: Pieza correspondiente al hombro

Fuente: Los autores

Para poder unir la pieza del hombro con la del brazo, se debe realizar una pieza complementaria, la misma que permite la unión de las dos piezas anteriormente nombradas como se muestra en la figura 3.56.

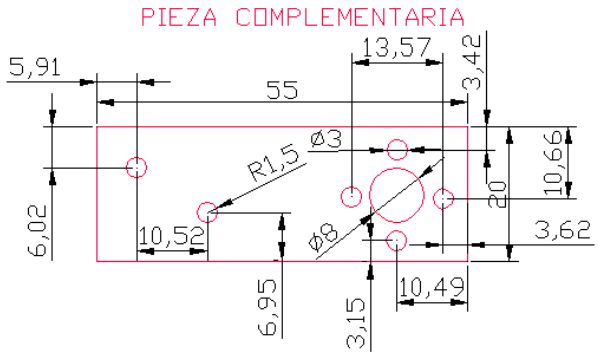


Figura 3.56: Pieza complementaria para unión de hombro con brazo

Fuente: Los autores

Para el desarrollo del brazo se consideran las medidas de la figura 3.57

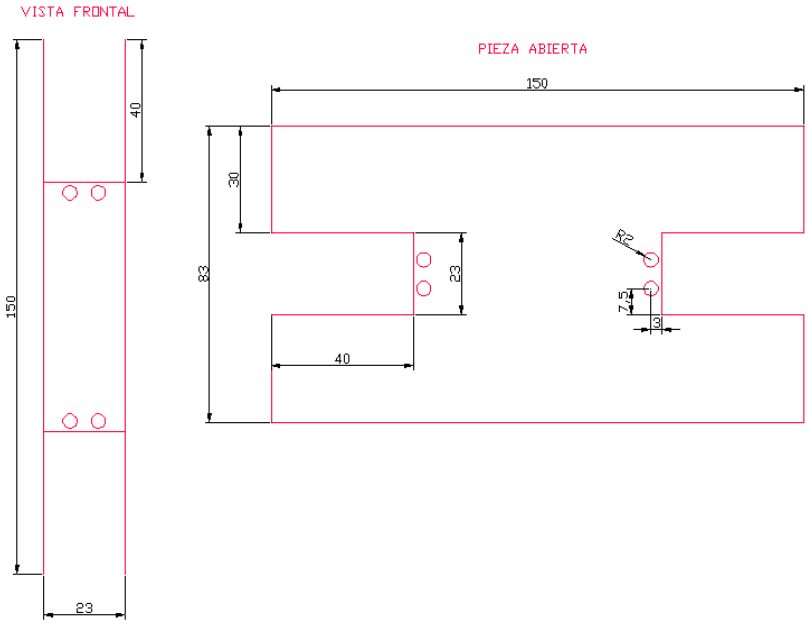


Figura 3.57: Pieza correspondiente al brazo del robot

Fuente: Los autores

En la figura 3.58 se puede apreciar el aspecto físico de esta pieza antes de ser doblada por la máquina dobladora.

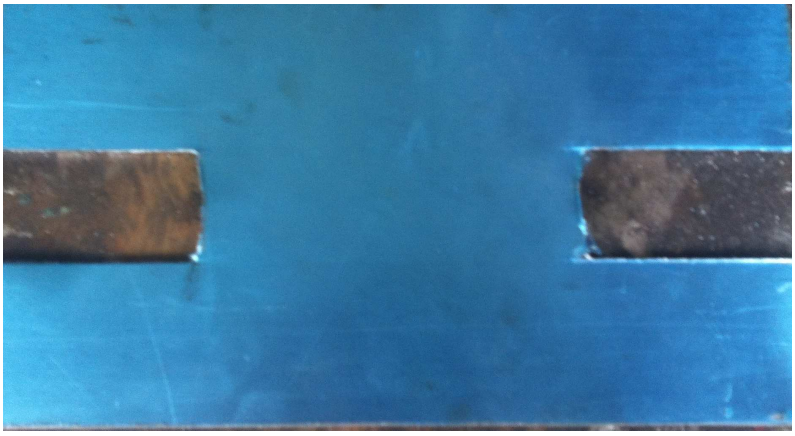


Figura 3.58: Pieza correspondiente al brazo del robot abierta

Fuente: Los autores

Para terminar, se debe realizar la pieza correspondiente a la mano del robot, la misma que se detalla en la figura 3.59.

Ésta pieza es de vital importancia ya que juega un papel muy importante al momento en que el robot se transforma de Auto a Humano, esta pieza es la encargada de dar el impulso para que el robot quede arrodillado después de pasar de modo auto y luego él mismo proceda a levantarse para pasar a modo humano.

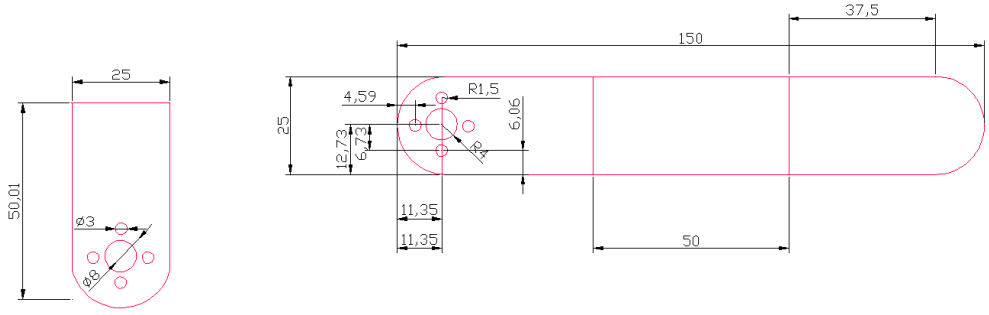


Figura 3.59: Pieza correspondiente a la mano del robot

Fuente: Los autores

Una vez armadas las tres piezas que forman a la mano del robot, se procede a unirlas mediante tornillos de 2mm con tuercas de presión para evitar que las piezas se aflojen, quedando como se ilustra en la figura 3.60.

Figura 3.60: Brazo izquierdo del robot armado en forma horizontal

Fuente: Los autores

3.6.4 Desarrollo de la cabeza del robot

A continuación detallamos el desarrollo correspondiente a la cabeza del robot, anotando que en esta pieza se montarán los sensores correspondientes a temperatura, distancia y luminosidad, así como también la tarjeta de control de los mismos.

En la figura 3.61 podemos apreciar las medidas de esta pieza.

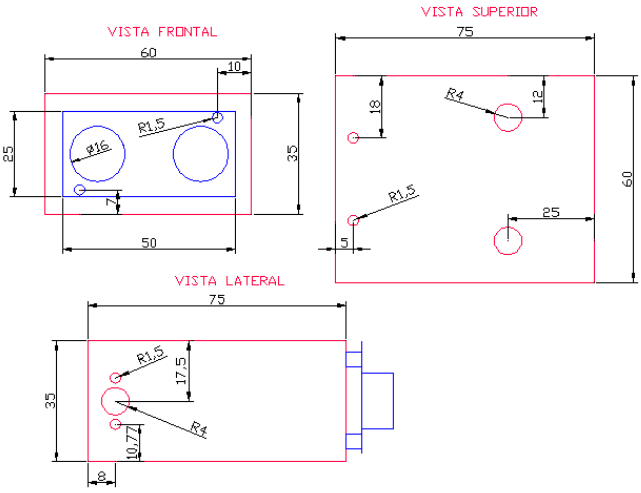


Figura 3.61: Vistas de la pieza correspondiente a la cabeza del robot

Fuente: Los autores

Cabe mencionar que lo dibujado de color azul representa al sensor de distancia PING de PARALLAX, es decir, no es parte de la pieza de aluminio de la cabeza del robot.

3.6.5 Desarrollo de la pieza de soporte para la electrónica de control

Sobre esta estructura de aluminio se montará la tarjeta de control principal, junto con la batería de alimentación para el control del circuito, los bornes de programación vía USB, el interruptor de encendido principal, el socket de la batería, y el jack de conexión para las baterías de los servomotores como se muestra en la figura 3.62.

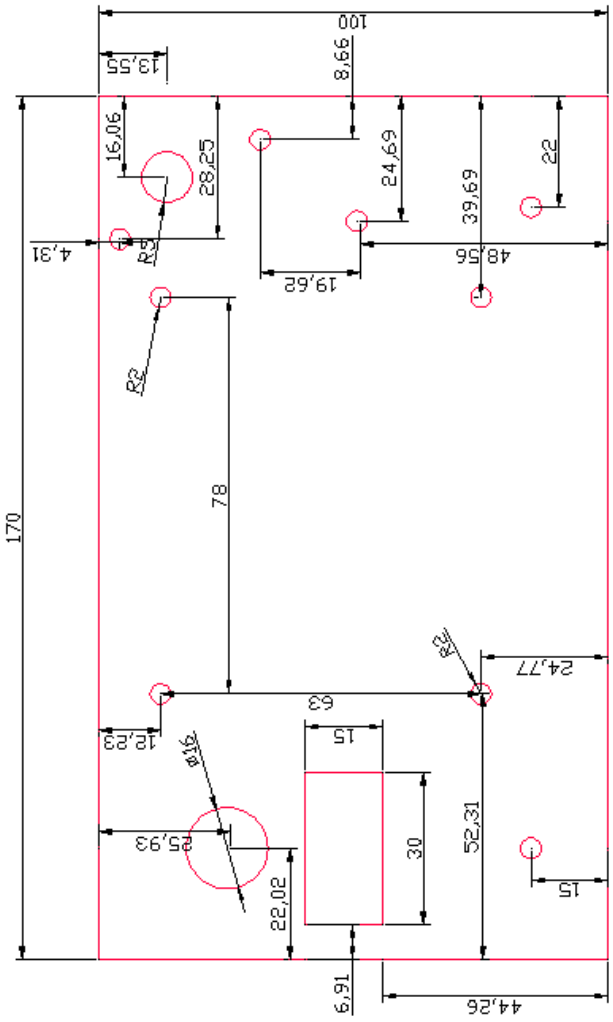


Figura 3.62: Pieza de soporte para la electrónica de control

Fuente: Los autores

En las siguientes figuras se pueden apreciar los aspectos físicos reales de la pieza nombrada, tanto en la parte interior como en la parte exterior.

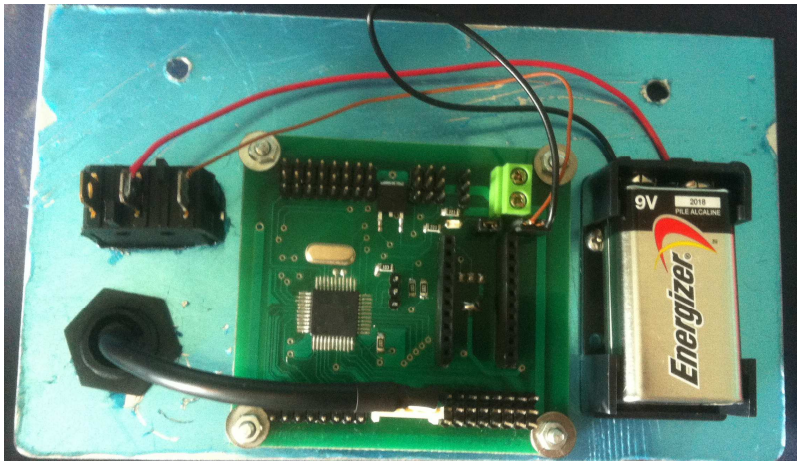


Figura 3.63: Vista interior de la pieza de soporte para la electrónica de control

Fuente: Los autores

Figura 3.64: Vista exterior de la pieza de soporte para la electrónica de control

Fuente: Los autores

Cabe mencionar que en la figura 3.63 el conector para la batería de 9V está conectado en la cara interna de esta pieza, pero, en el diseño final el conector se lo instala como se muestra en la figura 3.64.

Con todas las piezas anteriores ya ensambladas se tiene la mitad del robot armado (de la cadera hacia arriba), como se muestra en la figura 3.65.

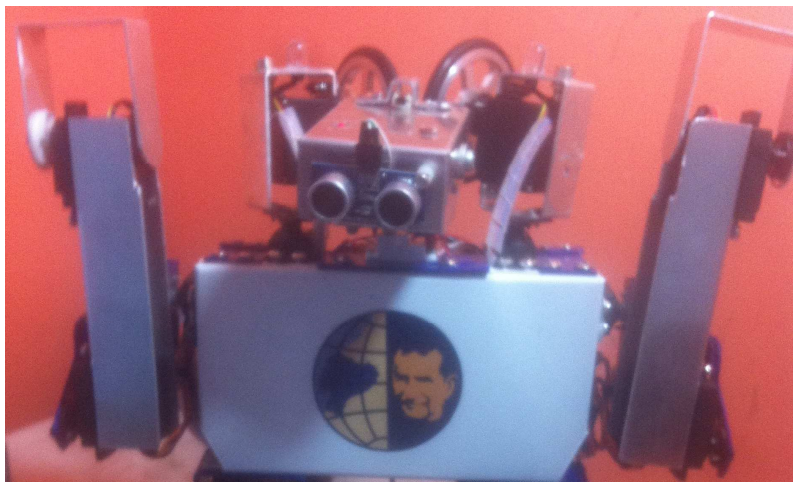


Figura 3.65: Robot transformer ensamblado hasta la mitad

Fuente: Los autores

3.6.6 Desarrollo de las piezas de las piernas

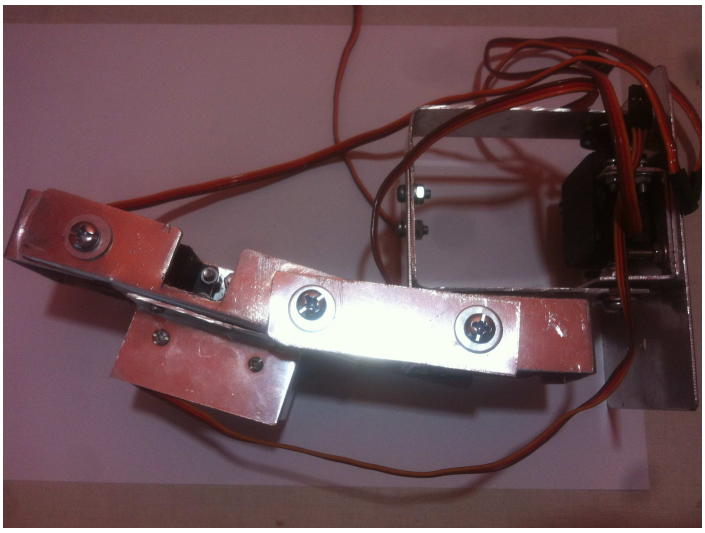


Figura 3.66: Aspecto físico de la pierna del robot

Fuente: Los autores

Para el desarrollo de las piernas se utilizaron varias herramientas como la dobladora, el taladro de pedestal con sus respectivas brocas y las limas para acero.

Se empieza el diseño de cada pierna dividiéndola en cuatro piezas fundamentales, las cuales son:

- Cadera
- Rodilla
- Tobillo
- Pie

3.6.6.1 Desarrollo de la cadera

Para el desarrollo de la cadera se utiliza una pieza de aluminio de 2mm con dimensiones de 13cm de largo y 2.5cm de ancho. Luego se procede a cortarla con la cizalla, hacerle los huecos con el taladro de pedestal y doblarla en forma de U como se muestra en la figura.

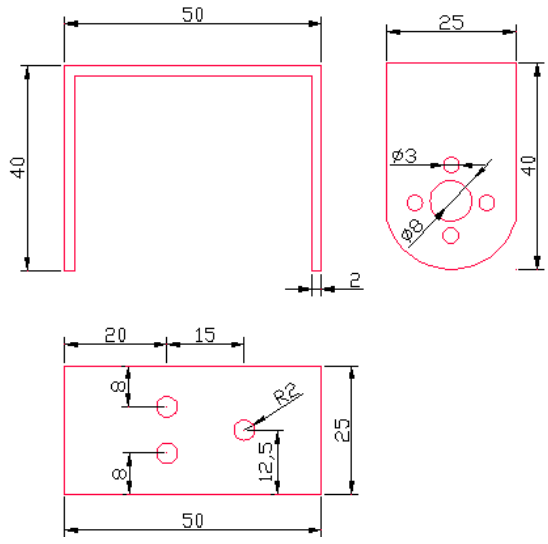


Figura 3.67: Dibujo de la pieza correspondiente a la cadera del robot

Fuente: Los autores

Una vez realizada la pieza, se la puede ensamblar sobre la estructura principal del robot como se muestra en la figura 3.68.

Figura 3.68: Ensamble de la cadera sobre la estructura principal del robot

Fuente: Los autores

3.6.6.2 Desarrollo de la rodilla

Para el desarrollo de la rodilla se debe realizar una estructura sólida que permita el acople de dos servomotores, tanto el servomotor para el movimiento de la cadera, como el servomotor propio de la rodilla. Además de esos dos servos, se acopla sobre la misma estructura un servo para la rueda, la misma que servirá cuando el robot se transforme en modo auto.

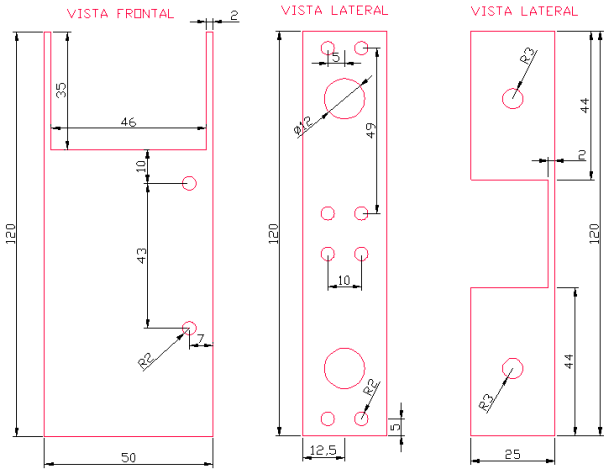


Figura 3.69: Pieza para acople de cadera con rodilla y base para rueda en modo auto

Fuente: Los autores

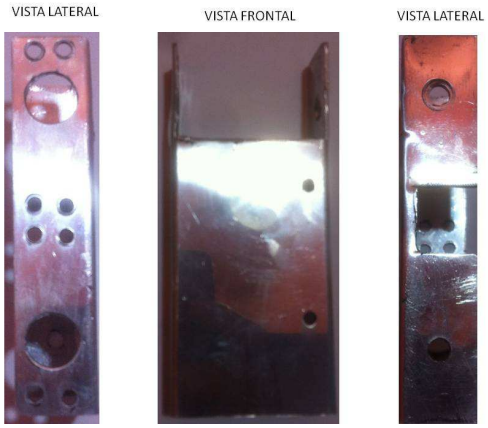


Figura 3.70: Aspecto físico de la pieza para el acople de la cadera con la rodilla y la base para la rueda en modo auto

Fuente: Los autores

Una vez desarrollada esta pieza, se empieza a fabricar las piezas complementarias, las mismas que sirven para acoplar el servomotor de la llanta y los ejes de los servomotores como se muestra en la siguiente figura.

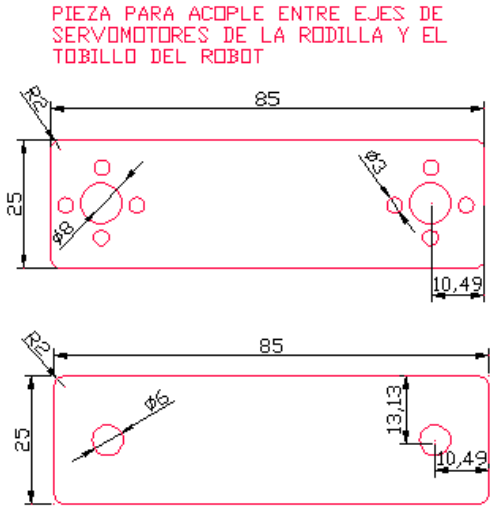


Figura 3.71: Pieza para el acople de los ejes de los servomotores de la rodilla con el tobillo

Fuente: Los autores

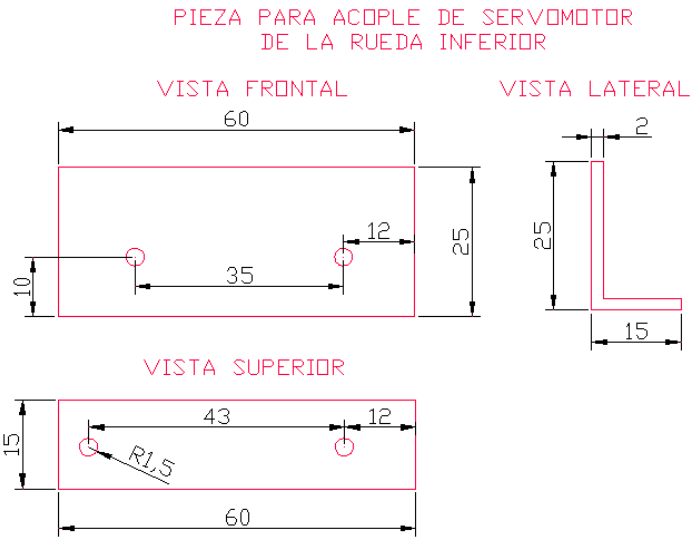


Figura 3.72: Pieza para acople de servomotor de la rueda y pieza de la rodilla

Fuente: Los autores

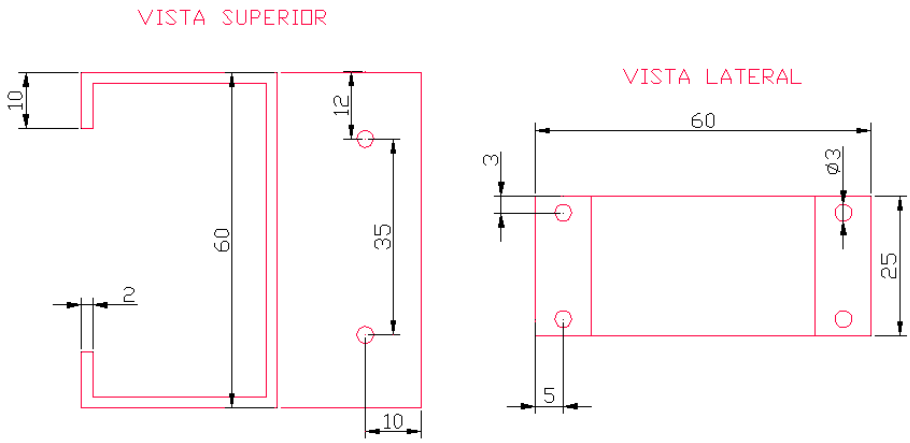


Figura 3.73: Pieza para acople de servomotor y rueda unida con la pieza para acople de la rodilla

Fuente: Los autores

En la figura 3.74 se puede apreciar el aspecto físico de estas dos piezas unidas entre sí para formar una sola pieza, la misma que permitirá montar el servomotor de la rueda inferior con la pieza principal de la rodilla del robot.

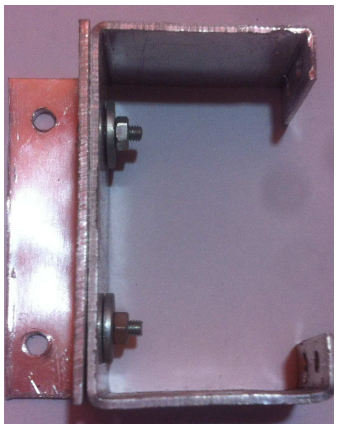


Figura 3.74: Aspecto físico de la pieza para acople de servomotor y rueda

Fuente: Los autores

3.6.6.3 Desarrollo del Tobillo

Para el desarrollo del tobillo del robot, se considera los acoples que se deben realizar entre la rodilla, el tobillo y el pie. Por esa razón se necesita realizar piezas complementarias que ayuden a la unión de las demás partes y formar la pieza del tobillo final.

En la figura 3.75 se aprecian las medidas para el desarrollo del tobillo.

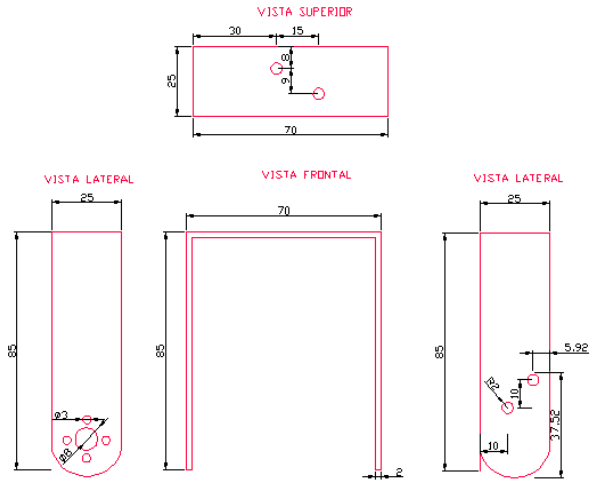


Figura 3.75: Pieza correspondiente al tobillo del robot

Fuente: Los autores

Como se mencionó anteriormente, se necesita de piezas complementarias para ayudar al acople entre las demás partes, sobre todo por los servomotores que

deben quedar fijos sobre las estructuras. Para el caso del tobillo se necesitan dos piezas complementarias para poder realizar las uniones necesarias como se muestran en las figuras 3.76 y 3.77

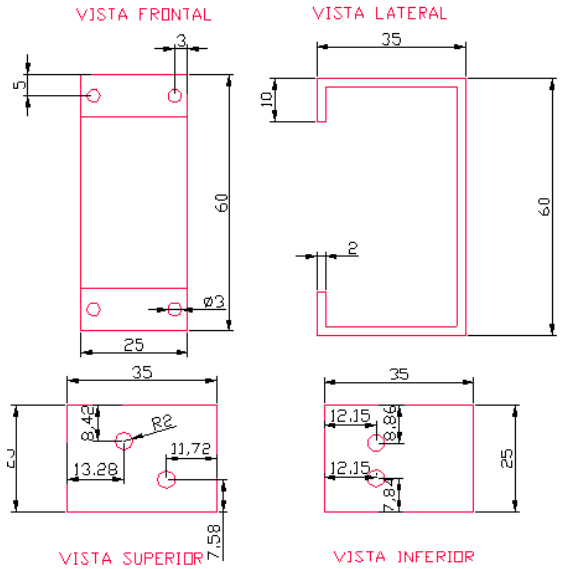


Figura 3.76: Pieza en forma de "C" para acople del servomotor del tobillo

Fuente: Los autores

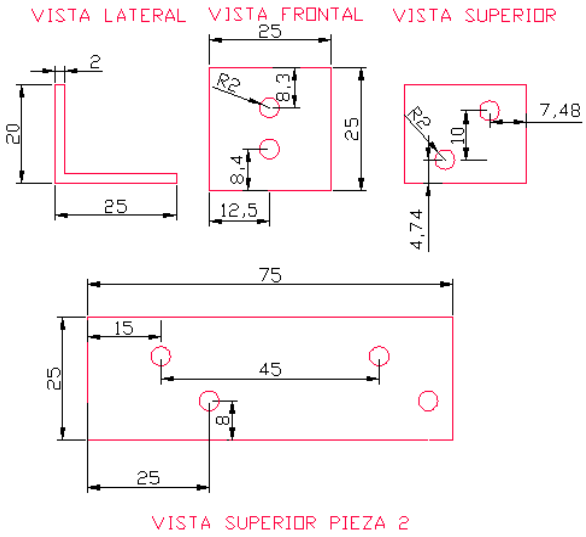


Figura 3.77: Piezas complementarias para acople del tobillo

Fuente: Los autores

En la figura 3.76 se pueden apreciar dos piezas complementarias para acople entre la pieza principal del tobillo y la pieza en forma de "C" donde va montado el servomotor.

En la siguiente figura se puede apreciar el montaje final entre todas las piezas para formar el tobillo del robot. Cabe recalcar que se deben realizar dos de esta pieza debido a que el robot posee dos tobillos.

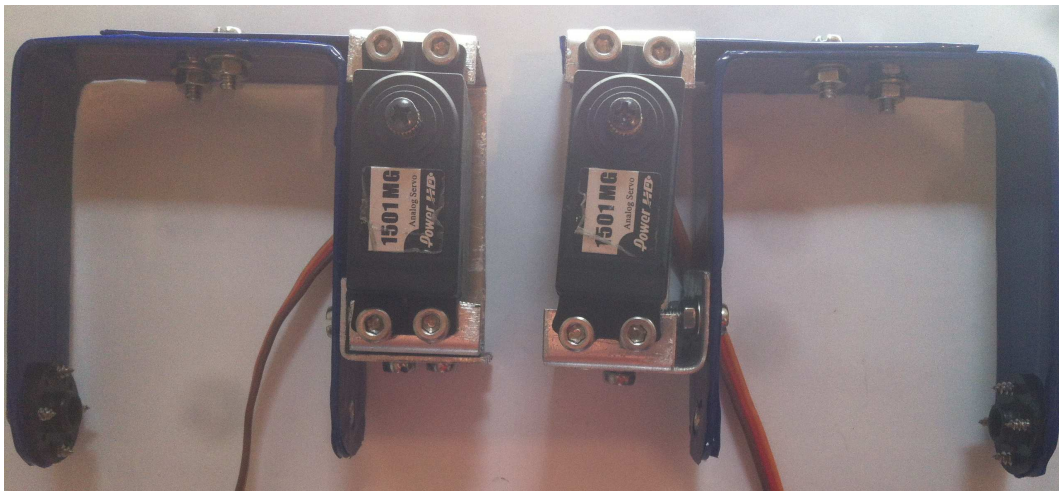


Figura 3.78: Piezas finales de los tobillos derecho e izquierdo

Fuente: Los autores

3.6.6.4 Desarrollo del Pie

Al igual que el tobillo, para desarrollar el pie se necesita de dos piezas complementarias, las cuales acoplándolas adecuadamente nos dará la forma final del pie.

Estas piezas complementarias son:

- Pieza para soporte y ajuste del servomotor del pie
- Pieza tipo vincha para sujetar la batería de los servomotores al pie

En la figura 3.79 se puede apreciar el dibujo con las medidas de la pieza para el ajuste del servomotor al pie.

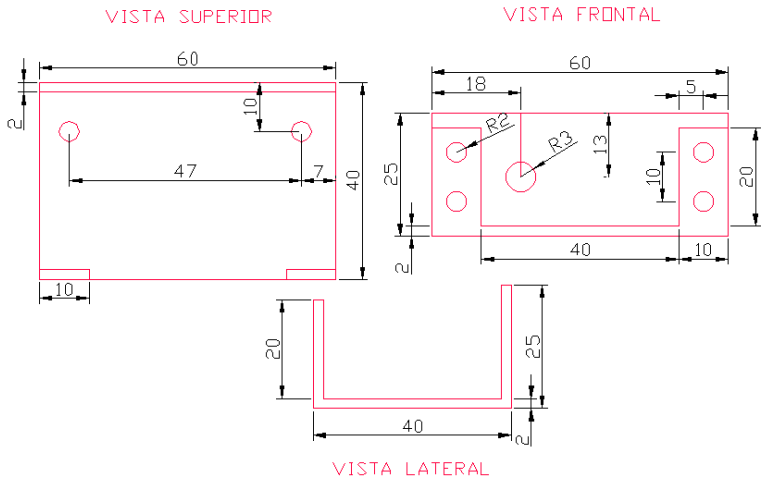


Figura 3.79: Piezas para ajuste del servomotor al pie

Fuente: Los autores

En la figura 3.80 se detallan las medidas para la realización de la pieza tipo vincha para ajuste de la batería del robot al pie del mismo.

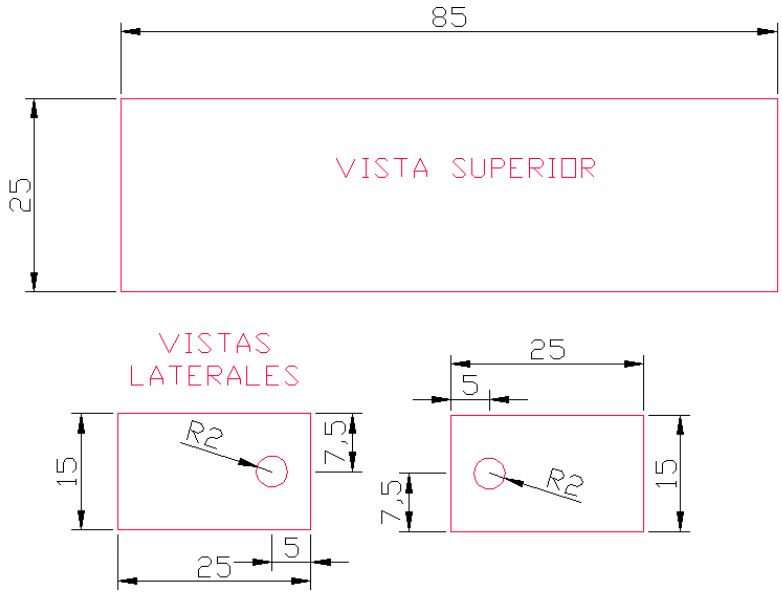


Figura 3.80: Pieza tipo vincha para el ajuste de la batería al pie del robot

Fuente: Los autores

Una vez desarrolladas las piezas complementarias, se procede a la realización de la pieza principal del pie cuyas medidas se las puede apreciar en la siguiente figura.

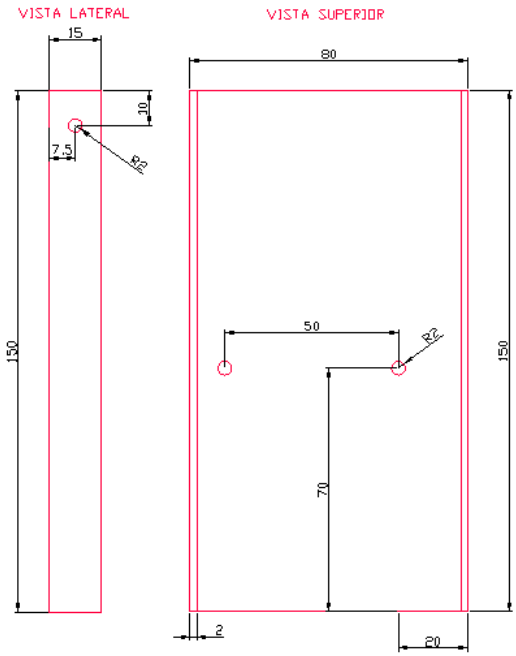


Figura 3.81: Pieza correspondiente al pie del robot

Fuente: Los autores

Una vez terminadas las piezas, se procede a forrarlas con vinil de colores y a unirlas entre sí para formar la pieza final del pie como se muestra en la figura 3.82.

Figura 3.82: Piezas unidas formando el pie del robot

Fuente: Los autores

3.6.7 Ensamble de la pierna del robot

Para ensamblar cada pierna del robot, se deben unir las piezas anteriormente armadas, las cuales con los servomotores instalados forman la pieza final de la pierna del robot como se muestra en la siguiente figura.

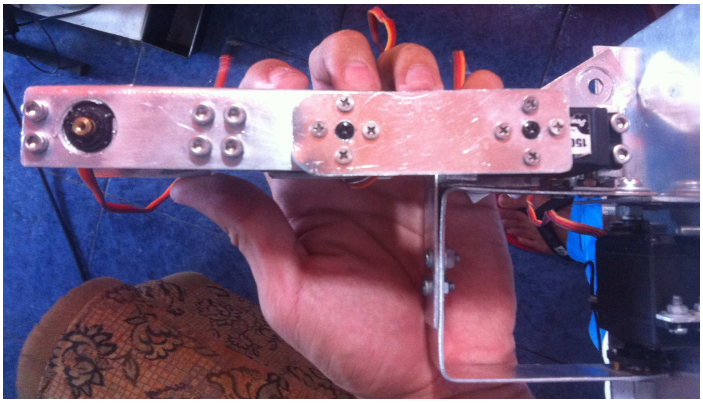


Figura 3.83: Pierna del robot ensamblada para pruebas

Fuente: Los autores

Figura 3.84: Piernas del robot ensambladas con la estructura principal para pruebas

Fuente: Los autores

Una vez realizada las pruebas necesarias para verificar la rutina de caminata del robot en modo Humano, se procede a cubrir las piezas de aluminio con vinil dándole un acabado mas agradable para su ensamblaje final.

Figura 3.85: Ensamble de cadera, rodilla, tobillo y pie en prototipo final

Fuente: Los autores

3.6.8 Ensamblaje final del robot

En esta parte de la investigación se realiza el montaje o ensamblaje final del robot, el mismo que consiste en acoplar las piernas del robot con la sección anteriormente armada (Pieza principal, cabeza, brazos), teniendo como resultado al robot "transformer" Humanoide – Auto armado en su totalidad, listo para pruebas como se muestra en la figura 3.86.

Figura 3.86: Robot transformer humanoide – auto listo para ser programado

Fuente: Los autores

No está de más mencionar que para llegar al diseño final, se realizaron varios prototipos que se utilizaron para realizar pruebas de resistencia mecánica, motricidad y estabilidad.

3.7 Diseño estructural del robot hexápodo

3.7.1 Introducción

En esta sección se detallará el desarrollo de las piezas que conforman al robot transformer Hexápodo - Auto, así como también a su base estructural principal y el armado de las mismas.

Cabe mencionar que en la parte del diseño del robot, se tomaron varios modelos de robots que se venden en el mercado internacional, donde se decidió seguir un modelo similar al actual por tener una forma estructural muy sólida que facilita la motricidad de cada una de las partes.

El transformer Hexápodo - Auto, consta de 25 servomotores, los mismos que trabajarán a 6 voltios con una fuente independiente a la del control principal.

3.7.2 Base estructural principal

Primeramente se elabora la base estructural principal, la misma que representa al cuerpo del robot, donde estarán instalados los servomotores de los hombros, la electrónica de control, las baterías y las ruedas. Esta estructura fue realizada con acrílico de 6mm de espesor con sus respectivos cortes y agujeros para la instalación de cada servo.

Se decidió utilizar el acrílico de 6mm como materia prima por sus características mecánicas como peso, rigidez por tener 6mm de espesor, durabilidad y fácil utilización.

A continuación se detalla en la figura 3.87 las medidas correspondientes de la base estructural principal que representa al cuerpo del robot transformer Hexápodo – Auto.

Cabe recalcar que se realizarán dos estructuras similares, debido a que se necesitan de las dos estructuras para sujetar los servomotores de las patas.

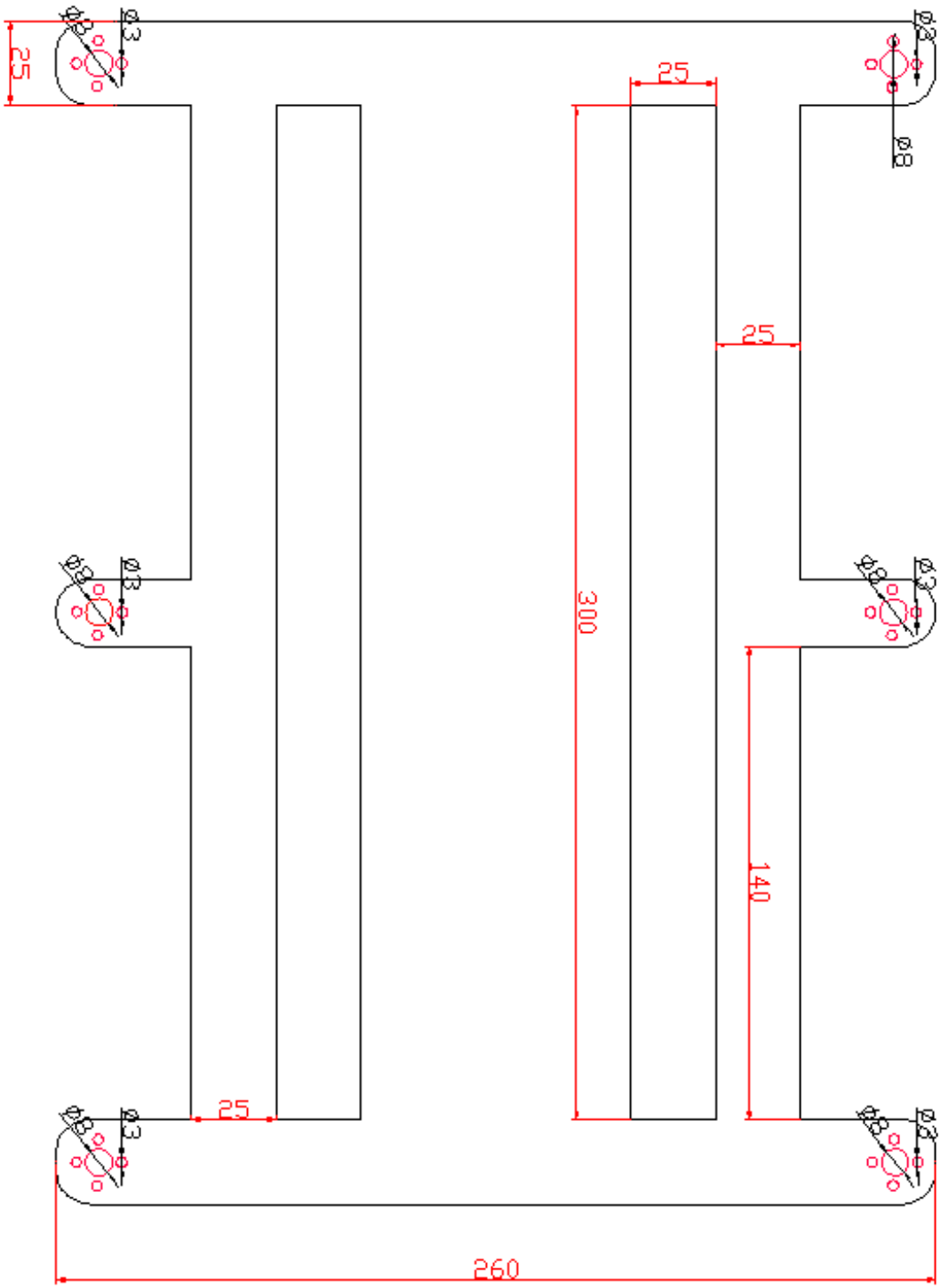


Figura 3.87: Medidas para la base principal del robot hexápodo

Fuente: Los autores

3.7.3 Construcción de la cara frontal del robot hexápodo

Esta cara será el frente del robot, la misma que se la desarrolló en acrílico de 3mm de grosor. Una vez realizada, se cubre la estructura con un impresión de vinil con los colores del ejército debido a que cuando se transforma de Hexápodo a Auto, este "auto" en realidad es un "tanque de guerra".

En la figura 3.88 se aprecia el dibujo de la cara frontal del robot hexápodo.

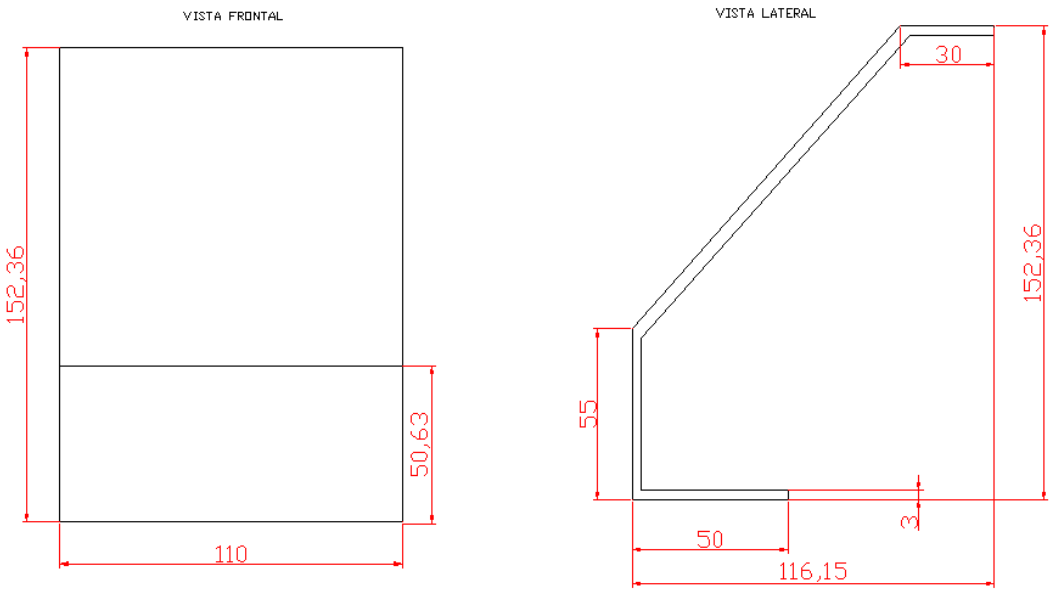


Figura 3.88: Dibujo de la cara frontal del robot hexápodo

Fuente: Los autores

En la figura 3.89 se aprecia la forma física real de la cara frontal del robot hexápodo, la misma que lleva el logo de la prestigiosa UPS-Gye.

Adicional a esto cabe recalcar que así como existe una cara frontal, también existe una cara trasera en el robot hexápodo, de las mismas dimensiones, por tal motivo se excluye el dibujo de la misma.

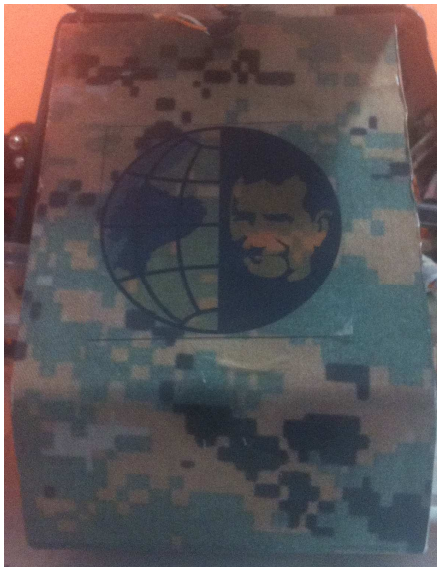


Figura 3.89: Aspecto físico final de la cara frontal del robot hexápodo

Fuente: Los autores

3.7.4 Desarrollo de las patas del hexápodo

El robot hexápodo está conformado por seis extremidades, lo que le da un total de seis patas para poder realizar sus movimientos. Esta pata fue elaborada a base de acrílico de 6mm, ya que son las encargadas de soportar todo el peso del robot en modo Hexápodo.

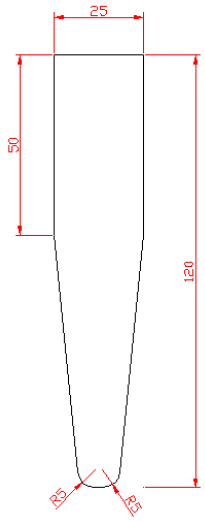


Figura 3.90: Dibujo de la pata del robot hexápodo

Fuente: Los autores

Cada pata va montada sobre un servomotor, el mismo que la impulsará para realizar cualquier tipo de movimiento dependiendo de la instrucción que la tarjeta de control le envíe.

Figura 3.91: Pata del robot montada sobre el servomotor

Fuente: Los autores

3.7.5 Diseño de las tapas laterales del robot hexápodo

Estas tapas laterales simplemente son utilizadas como soporte entre la base principal del robot hexápodo y su tapa superior, así como también para darle más estética el robot y no dejar esos espacios al aire libre. En total el hexápodo posee cuatro de estas tapas.

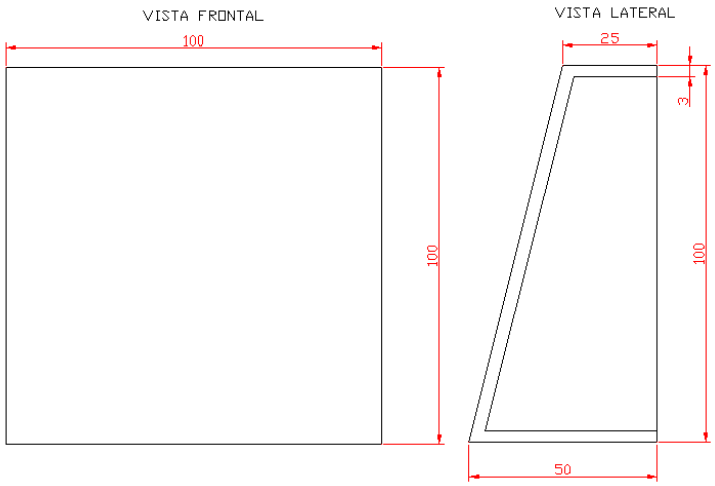


Figura 3.92: Tapa lateral del robot hexápodo

Fuente: Los autores

Figura 3.93: Aspecto físico final de la tapa lateral del robot hexápodo

Fuente: Los autores

3.7.6 Diseño de la tapa principal superior del robot hexápodo

Esta tapa superior será la encargada de cubrir toda la electrónica de control y baterías que el robot hexápodo necesite para su funcionamiento, además, sobre esta tapa se instalará un servomotor encargado de darle movimiento a la "arma" del robot en modo Auto (realmente en forma de tanque de guerra), llamada domo.

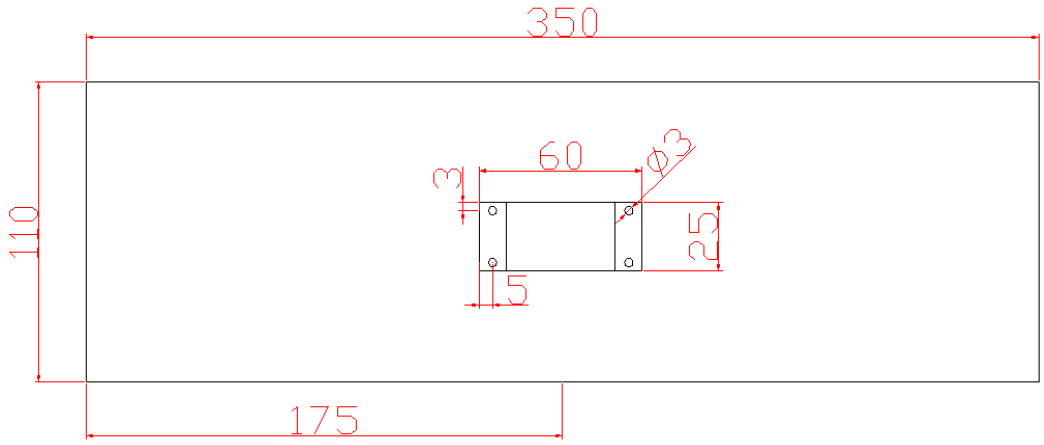


Figura 3.94: Tapa superior del robot hexápodo

Fuente: Los autores

Figura 3.95: Aspecto físico de la tapa superior del robot hexápodo con el domo instalado sobre ella

Fuente: Los autores

3.7.7 Ensamblaje del robot hexápodo

Para el ensamblaje se empieza armando las seis patas, luego se las instala en la pieza principal del hexápodo. Con las patas puestas sobre la base principal, se procede a instalar los servomotores, ruedas, y la electrónica en general, al terminar, se instalan las cuatro tapas laterales sobre la estructura terminando con la instalación de la tapa superior y el domo.

En la figura 3.97 se puede apreciar el robot hexápodo totalmente armado, listo para ser programado y utilizado.

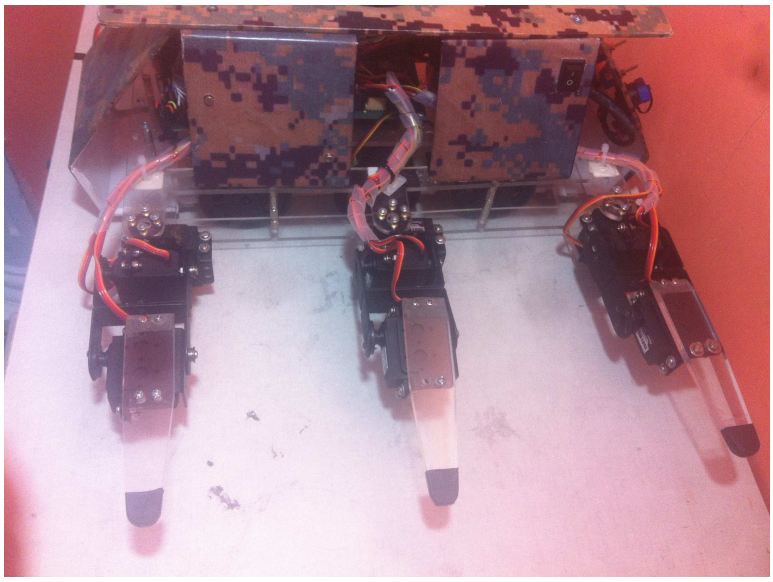


Figura 3.96: Aspecto físico de las piezas instaladas sobre la estructura principal, donde se aprecian las patas, las tapas laterales y la tapa superior del hexápodo

Fuente: Los autores

Figura 3.97: Robot transformer hexápodo – auto listo para ser usado

Fuente: Los autores

3.8 Transformaciones de los robots transformer

3.8.1 Introducción

Se detallara mediante imágenes las formas físicas que los robots adquieren en cada tipo de transformación, tomando en cuenta diferentes ángulos de vistas.

3.8.2 Forma física de las transformaciones del robot humanoide – auto

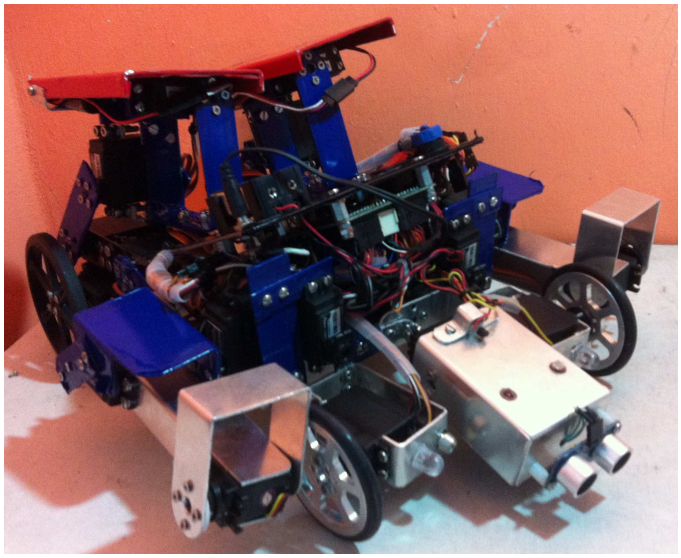


Figura 3.98: Robot transformer humanoide – auto en modo auto vista 1

Fuente: Los autores

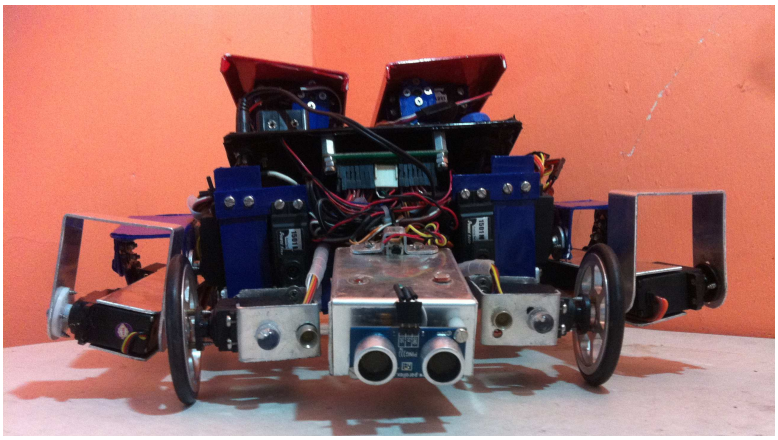


Figura 3.99: Robot transformer humanoide – auto en modo auto vista 2

Fuente: Los autores

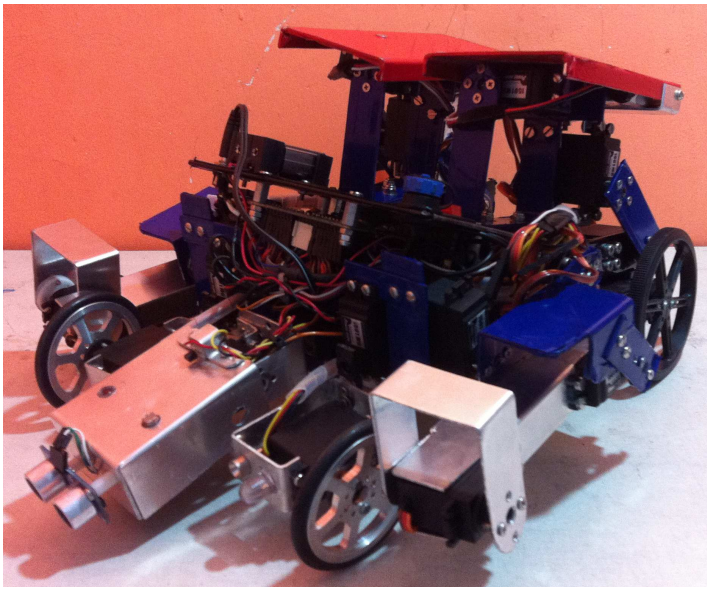


Figura 3.100: Robot transformer humanoide – auto en modo auto vista 3

Fuente: Los autores

Figura 3.101: Robot transformer humanoide – auto en modo humano vista 1

Fuente: Los autores

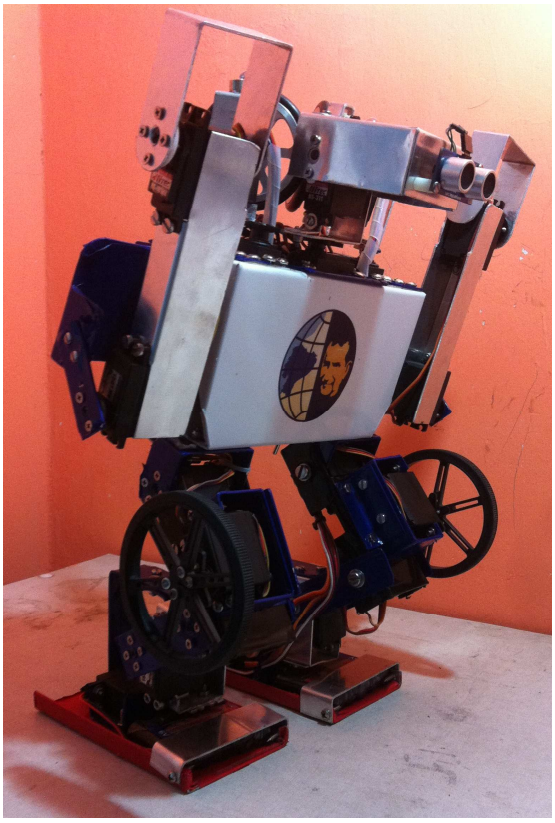


Figura 3.102: Robot transformer humanoide – auto en modo humano vista 2

Fuente: Los autores

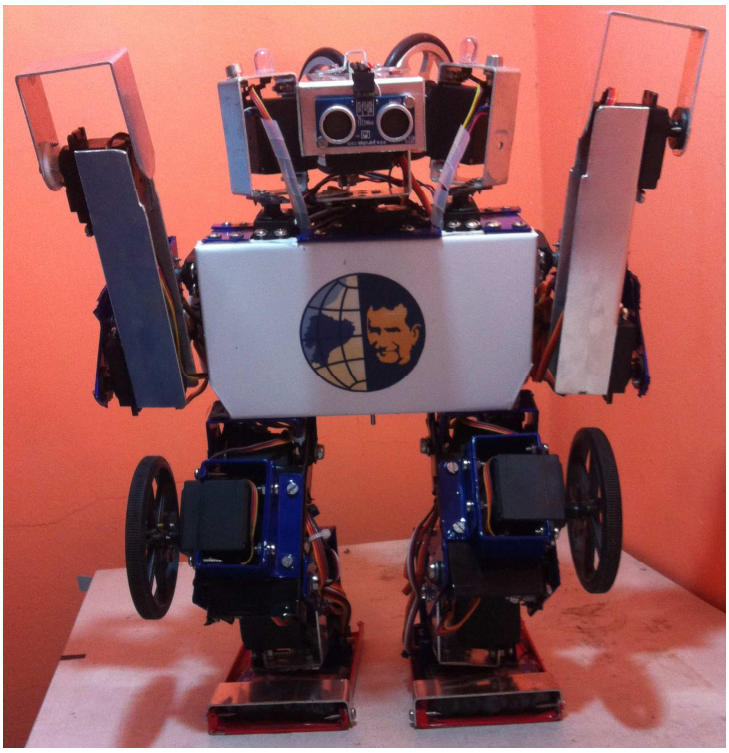


Figura 3.103: Robot transformer humanoide – auto en modo humano vista 3

Fuente: Los autores

3.8.3 Forma física de las transformaciones del robot hexápodo – auto

Figura 3.104: Robot transformer hexápodo – auto en modo hexápodo vista 1

Fuente: Los autores

Figura 3.105: Robot transformer hexápodo – auto en modo hexápodo vista 2

Fuente: Los autores

Figura 3.106: Robot transformer hexápodo – auto en modo auto

Fuente: Los autores

3.9 Algoritmo de programación para los robots Transformer y en control remoto

3.9.1 Introducción

Este algoritmo es el encargado de controlar todas las funciones que el robot realiza ya sea en modo manual o modo automático. El algoritmo está desarrollado en el compilador PIC *CC Compiler*, el mismo que soporta únicamente microcontroladores PIC y su lenguaje de programación es el lenguaje en C.

En este capítulo solo se detalla la programación principal de cada robot, las rutinas de programación secundarias y complementarias no serán detalladas. (Véase ANEXO 2 para el detalle de las subrutinas de programación).

3.9.2 Algoritmo de programación del robot transformer humano – auto

```
include <16f877a.h>
 #use delay(clock=20000000)

 #fuses hs,nowdt,noprotect,nolvp

 #use rs232(baud=9600,xmit = pin_e0 , rcv = pin_e1, bits=8)


// #priority timer1,rda       //prioridad al timer_1

 #use standard_io(a)

 #use standard_io(b)

 #use standard_io(c)

 #use standard_io(d)

 #use standard_io(e)
```

```
#include <servo.c>

//#include <camina.c>

#include <inicio.c>

#include <humano.c>

#include <carro_2.c>

#include <1wire.c>

#include <ds1820.c>

#include <medicion_sensores.c>

#include <camina_prueba.c>


#define man_auto pin_d7


 const int16 Ticks4Window = 12500; // PWM Window for servo = 2.5 ms x 8 =
20 ms centro

 const int16 Centro = 7500; // PWM High for Center Position = 1.5 ms


 static int16 Servo_PWM[8]={Centro,Centro,Centro,Centro,0,0,0,0};

 static int8 Servo_Idx=0;


 static int1 SERVO1_ON=1;

 static int1 SERVO2_ON=1;

 static int1 SERVO3_ON=0;

 static int1 SERVO4_ON=0;
```

```
static int1 SERVO5_ON=0;

static int1 SERVO6_ON=0;

static int1 SERVO7_ON=1;

static int1 SERVO8_ON=1;


static int1 flag_Phase;


static int16 Ticks4NextInterrupt=53036; //2.5ms "valor de carga del timer no ticks"



#int_timer1

void timer1_isr(void){


    if(flag_Phase==0){

    if(Servo_Idx==0 && SERVO1_ON) output_high(coello);

    if(Servo_Idx==1 && SERVO2_ON) output_high(cabeza);

    if(Servo_Idx==2 && SERVO3_ON) output_high(rueda_sup_der);

    if(Servo_Idx==3 && SERVO4_ON) output_high(rueda_sup_izq);

    if(Servo_Idx==4 && SERVO5_ON) output_high(rueda_inf_der);

    if(Servo_Idx==5 && SERVO6_ON) output_high(rueda_inf_izq);

    if(Servo_Idx==6 && SERVO7_ON) output_high(mano_der);

    if(Servo_Idx==7 && SERVO8_ON) output_high(mano_izq);
```

```
    Ticks4NextInterrupt = 65535 - Servo_PWM[Servo_Idx];
    set_timer1(Ticks4NextInterrupt);
   }
  if(flag_Phase==1){
    if(Servo_Idx==0 && SERVO1_ON) output_low(coello);
    if(Servo_Idx==1 && SERVO2_ON) output_low(cabeza);
    if(Servo_Idx==2 && SERVO3_ON) output_low(rueda_sup_der);
    if(Servo_Idx==3 && SERVO4_ON) output_low(rueda_sup_izq);
    if(Servo_Idx==4 && SERVO5_ON) output_low(rueda_inf_der);
    if(Servo_Idx==5 && SERVO6_ON) output_low(rueda_inf_izq);
    if(Servo_Idx==6 && SERVO7_ON) output_low(mano_der);
    if(Servo_Idx==7 && SERVO8_ON) output_low(mano_izq);

    Ticks4NextInterrupt = 65535 - Ticks4Window + Servo_PWM[Servo_Idx];
    set_timer1(Ticks4NextInterrupt);

    if(++Servo_Idx>7) Servo_Idx=0; //contador para idx q es una variable que
define
                                   //el servo
   }
   ++flag_Phase;
 }
```

```
void angular(int8 servo, int8 angulo){
int16 pulso;
int16 tcm;
pulso=(11.11*angulo)+500.1; //1500 ejm
tcm=(pulso/0.2);//7500 ejm "ticks"
Servo_PWM[servo]=tcm;

}



void main(){
 int8 h;
 int8 x;
 int8 i;
 int8 valor;
 int8 valor_2;
 int1 car;
 int1 bipedo;

  disable_interrupts(global);
  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
```

```
   setup_timer_1(T1_INTERNAL | T1_DIV_BY_1);
   port_b_pullups(FALSE);

//inicializo xbee
   for (i=0;i<10;i++){
   putc(5);
   delay_ms(500);
   }
//////////////////////////

output_high(led_amarillo);
delay_ms(500);
output_low(led_amarillo);
delay_ms(500);



inicial();
while(input(man_auto)==1){ //modo manual
bipedo=1; //indico q esta en modo humano
car=0;
   disable_interrupts(global);
```

```
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_timer_1(T1_INTERNAL | T1_DIV_BY_1);
   port_b_pullups(FALSE);

//robot listo para usar

output_high(led_blanco);
delay_ms(500);
output_low(led_blanco);
delay_ms(500);
putc(2);
delay_ms(500);
////////////////////////////////////////////////////////////////////////////
//////////////////////recepcion de datos serial//////////////////
////////////////////////////////////////////////////////////////////////////
while(1){
valor=getc();
////////////////////////////////////////////////////////////////////////////
/////////////////Transformar de humano a carro///////////////
////////////////////////////////////////////////////////////////////////////
if((valor==4)&& (bipedo==1)){ //esta en modo humano? 0 es no, 1 es si
output_high (led_blanco);
```

```
delay_ms(500);
output_low (led_blanco);
output_high(enable);
delay_ms(1000);
output_low(in_a);
output_high (in_b);
output_low(enable);
delay_ms(3000);
output_high(enable);
   enable_interrupts(int_rda);
   set_timer1(Ticks4NextInterrupt);
   enable_interrupts(int_timer1);
   enable_interrupts(global);
angular(0,90);//coello
angular(1,180);//cabeza
angular(6,90);// mano_der
angular(7,90);// mano_izq
carro();
angular(6,40);// mano_der
angular(7,40);// mano_izq hd
delay_ms(2000);
carro_22();
```

```
angular(6,90);// mano_der
angular(7,90);// mano_izq
carro_33();
angular(6,5);// mano_der *0
angular(7,5);// mano_izq *180 hd
car=1; //bandera para indicar que esta en modo auto o carro
bipedo=0;
delay_ms(100);
while(1){
  valor_2=getc();
  if(valor_2==3){ //adelante
  for(h=0;h<50;h++){
     output_high (led_blanco);
     output_low (led_amarillo);
output_high(rueda_sup_izq); //gira izq-hitec
output_high(rueda_sup_der); //gira_der
output_high(rueda_inf_der); //gira_der
delay_us(500);//der
output_low(rueda_inf_der);
output_low(rueda_sup_der);
output_low(rueda_sup_izq);
output_high(rueda_inf_izq); //gira_izq
```

```
delay_us(2500);// izq
output_low(rueda_inf_izq);
delay_ms(17);
}
}
 if(valor_2==7){ //atras
for(h=0;h<100;h++){
output_high (led_blanco);
output_low (led_amarillo);
output_high(rueda_sup_izq);
output_high(rueda_sup_der);
output_high(rueda_inf_der);
delay_us(2500);//izq
output_low(rueda_inf_der);
output_low(rueda_sup_der);
output_low(rueda_sup_izq);
output_high(rueda_inf_izq);
delay_us(500);// der
output_low(rueda_inf_izq);
delay_ms(17);
}
     }
```

```
if(valor_2==8){ //g_derecha
for(h=0;h<100;h++){
      output_high (led_blanco);
      output_high (led_amarillo);
output_high(rueda_sup_izq); //gira izq hitec (delante)
delay_us(500);//der
output_low(rueda_sup_izq);
output_high(rueda_inf_izq); //gira izq
output_high(rueda_sup_der); //gira_izq
output_high(rueda_inf_der); //gira_izq
delay_us(2500);// izq
output_low(rueda_inf_izq);
output_low(rueda_sup_der);
output_low(rueda_inf_der);

delay_ms(17);
}
      }
 if(valor_2==9){ //g_izquierda
for(h=0;h<100;h++){
      output_high (led_blanco);
      output_high (led_amarillo);
```

```
output_high(rueda_inf_izq); //gira der
output_high(rueda_sup_der); //gira_der
output_high(rueda_inf_der); //gira_der
delay_us(500);//der
output_low(rueda_inf_izq);
output_low(rueda_sup_der);
output_low(rueda_inf_der);
output_high(rueda_sup_izq); //gira der hitec (delante)
delay_us(2500);//der
output_low(rueda_sup_izq);
delay_ms(17);
}
      }
  if(valor_2==0){ //salir
    output_low (led_blanco);
    output_low (led_amarillo);
  break;
  }
}
}
else if((valor==4)&& (bipedo==0)){ //esta en modo humano? 0 es no, 1 es si
while(1){
```

```
   valor_2=getc();
   if(valor_2==3){ //adelante
   for(h=0;h<50;h++){
      output_high (led_blanco);
      output_low (led_amarillo);
output_high(rueda_sup_izq); //gira izq-hitec
output_high(rueda_sup_der); //gira_der
output_high(rueda_inf_der); //gira_der
delay_us(500);//der
output_low(rueda_inf_der);
output_low(rueda_sup_der);
output_low(rueda_sup_izq);
output_high(rueda_inf_izq); //gira_izq
delay_us(2500);// izq
output_low(rueda_inf_izq);
delay_ms(17);
}
}
 if(valor_2==7){ //atras
for(h=0;h<100;h++){
output_high (led_blanco);
output_low (led_amarillo);
```

```
output_high(rueda_sup_izq);
output_high(rueda_sup_der);
output_high(rueda_inf_der);
delay_us(2500);//izq
output_low(rueda_inf_der);
output_low(rueda_sup_der);
output_low(rueda_sup_izq);
output_high(rueda_inf_izq);
delay_us(500);// der
output_low(rueda_inf_izq);
delay_ms(17);
}
     }
 if(valor_2==8){ //g_derecha
for(h=0;h<100;h++){
     output_high (led_blanco);
     output_high (led_amarillo);
output_high(rueda_sup_izq); //gira izq hitec (delante)
delay_us(500);//der
output_low(rueda_sup_izq);
output_high(rueda_inf_izq); //gira izq
output_high(rueda_sup_der); //gira_izq
```

```
output_high(rueda_inf_der); //gira_izq
delay_us(2500);// izq
output_low(rueda_inf_izq);
output_low(rueda_sup_der);
output_low(rueda_inf_der);
delay_ms(17);
}
    }
if(valor_2==9){ //g_izquierda
for(h=0;h<100;h++){
    output_high (led_blanco);
    output_high (led_amarillo);
output_high(rueda_inf_izq); //gira der
output_high(rueda_sup_der); //gira_der
output_high(rueda_inf_der); //gira_der
delay_us(500);//der
output_low(rueda_inf_izq);
output_low(rueda_sup_der);
output_low(rueda_inf_der);
output_high(rueda_sup_izq); //gira der hitec (delante)
delay_us(2500);//der
output_low(rueda_sup_izq);
```

```
delay_ms(17);
}
}
  if(valor_2==0){ //salir
    output_low (led_blanco);
    output_low (led_amarillo);
  break;
  }
}
}
////////////////////////////////////////////////////////////////////////////
//////////////////transformar de carro a humano///////////////
////////////////////////////////////////////////////////////////////////////
else if((valor==5) && (car==1)){
output_high (led_blanco);
delay_ms(500);
output_low (led_blanco);
humano();
x=0;
while(x<20){
angular(6,110);// mano_der *0
angular(7,110);// mano_izq *180 hd
```

```
x+=1;
}
delay_ms(2000);
angulo(11,90);//130homb der 90
angulo(12,90);//45homb izq 90
delay_ms(2000);
output_high(led_blanco);
delay_ms(500);
output_low(led_blanco);
humano_2();
angular(1,90);//cabeza
inicial();
angular(6,90);// mano_der
angular(7,90);// mano_izq
car=0; //bandera indica que dejo de ser carro
bipedo=1; //bandera indica que ahora esta en modo humano
}
//////////////////////modo humano,caminar//////////////////////
////////////////////////////////////////////////////////////////////////////
else if((valor==5) && (car==0)){ //0 ya no esta en modo carro

  output_high (led_blanco);
```

```
 delay_ms(500);
 output_low (led_blanco);
       while(1){
       valor=getc();
       if(valor==1){  //envia dato "caminar"
       camina_2(); //se entiende que esta en modo humano//////////////////////////////
          }
       else if (valor==0){ //envia dato "menu"
         output_high (led_blanco);
         delay_ms(500);
         output_low (led_blanco);
         break;
       }
       }
     }
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
else if(valor==6){
output_high (led_blanco);
delay_ms(500);
output_low (led_blanco);
```

```
medicion();//mido sensores, revisar respuesta de datos
}
}
}
while(input(man_auto)==0){ //modo automatico
//////////////////////modo automático "DEMO"/////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////llamada a subrutinas//////////////////////////
////////////////////////////////////////////////////////////////////////////
camina_2(); //en rutina defino numero de pasos
//*solo para manual mode
//medicion();//mido sensores, revisar respuesta de datos
////////////////////////////////////////////////////////////////////////////
//////////////////rutina de transformacion a carro//////////////
////////////////////////////////////////////////////////////////////////////
///habla trans carro
output_high(enable);
delay_ms(1000);
output_low(in_a);
output_high (in_b);
output_low(enable);
delay_ms(3000);
```

```
output_high(enable);
////////////////////////////////////////////////////////////////////////////
//////////////////////transformacion carro/////////////////////////
////////////////////////////////////////////////////////////////////////////
   enable_interrupts(int_rda);
   set_timer1(Ticks4NextInterrupt);
   enable_interrupts(int_timer1);
   enable_interrupts(global);
angular(0,90);//coello
angular(1,180);//cabeza
angular(6,90);// mano_der
angular(7,90);// mano_izq
carro();
angular(6,40);// mano_der
angular(7,40);// mano_izq hd
delay_ms(2000);
carro_22();
angular(6,90);// mano_der
angular(7,90);// mano_izq
carro_33();
angular(6,5);// mano_der *0
angular(7,5);// mano_izq *180 hd
```

```
rodar();
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////llamada a rutina tranformacion humano////
humano();
x=0;
while(x<20){
angular(6,110);// mano_der *0
angular(7,110);// mano_izq *180 hd
x+=1;
}
delay_ms(2000);
angulo(11,90);//130homb der 90
angulo(12,90);//45homb izq 90
delay_ms(2000);
output_high(led_blanco);
delay_ms(500);
output_low(led_blanco);
humano_2();
angular(1,90);//cabeza
inicial();
```

```
angular(6,90);// mano_der

angular(7,90);// mano_izq

car=0; //bandera indica que dejo de ser carro

bipedo=1; //bandera indica que ahora esta en modo humano

////////////////////////////////////////////////////////////////////////////

}

}
```

3.9.3 Algoritmo de programación del robot transformer hexápodo – auto

```
#include <16f877a.h>
 #fuses hs,nowdt,noprotect,nolvP
 #use delay(clock=20000000)
 #use rs232(baud=9600,xmit=pin_a1,rcv=pin_a0,stream=sensores)
 #use rs232(baud=9600,xmit=pin_e0,rcv=pin_e1,stream=xbee)
// #use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7, stream=COM1)
//#use rs232(baud=19200, xmit=PIN_B1, rcv=PIN_B0, stream=COM2)

 #use standard_io(a)
 #use standard_io(b)
 #use standard_io(c)
 #use standard_io(d)
 #use standard_io(e)
 #include <pulsoss.c>
 #include <INICIAL.c>
 #include <CARRO_ADELANTE.c>
 #include <CARRO_ATRAS.c>
 #include <PARADO.c>
 #include <CAMINA.c>
 #include <CAMINA1.c>
 #include <TRANSFORMACION.c>
```

```
#include <CARRO_DERECHA.c>

#include <CARRO_IZQUIERDA.c>


void main(){

INT8 REP,X;

int8 temp , dist , valor , valor2, aracnido,car;

REP=0;

X=0;

  setup_adc_ports(no_analogs);

  setup_adc(adc_off);

   port_b_pullups(false);

for(x=0;x<10;x++){

fputc(8,xbee);

delay_ms(500);

}

output_high(pin_e2);

delay_ms(500);

output_low(pin_e2);

delay_ms(5000);

WHILE(1){

while(INPUT(PIN_A2)== 0){

CAR=1;
```

```
ARACNIDO=0;
fputc(2,xbee);
delay_ms(1000);
INICIAL();
DELAY_MS(10);
while(1){
valor=fgetc(xbee);
if((valor==4)&&(aracnido==0)){
while(1){
valor2=fgetc(xbee);
delay_ms(100);
if(valor2==7){
CARRO_ADELANTE();
DELAY_MS(100);
}
if (valor2==3){
CARRO_ATRAS();
DELAY_MS(100);
//break;
}
 if (valor2==8){
```

```
CARRO_DERECHA();
DELAY_MS(100);}
 if (valor2==9){
CARRO_IZQUIERDA();
DELAY_MS(100);}
if(valor2==0){
//CAR=1;
break;}
}
}
 else IF((VALOR==4)&& (ARACNIDO==1)){
TRANSFORMACION();
DELAY_MS(100);
while(1){
valor2=fgetc(xbee);
delay_ms(100);
if(valor2==7){
CARRO_ADELANTE();
DELAY_MS(100);
}
if (valor2==3){
CARRO_ATRAS();
```

```
DELAY_MS(100);
}
 if (valor2==8){

CARRO_DERECHA();
DELAY_MS(100);}
 if (valor2==9){
CARRO_IZQUIERDA();
DELAY_MS(100);}
if(valor2==0){
break;}
}
}
if((valor==5)&&(car==1)){
car=0;
aracnido=1;
parado();
delay_ms(100);
while(1){
valor2=fgetc(xbee);
delay_ms(100);
IF(VALOR2==1){
```

```
CAMINA();
DELAY_MS(100);
CAMINA1();
DELAY_MS(100);
}
eLSE if (valor2==0){ //envia dato "menu"
        break;
 }
}
}
}
}

while(INPUT(PIN_A2)== 1){
inicial();
delay_ms(10);
CARRO_ADELANTE();
delay_ms(10);
CARRO_ATRAS();
delay_ms(10);
CARRO_DERECHA();
delay_ms(10);
```

```
CARRO_IZQUIERDA();
delay_ms(10);
PARADO();
DELAY_MS(10);
REP=0;
WHILE(REP<3){
DELAY_MS(20);
CAMINA();
DELAY_MS(20);
CAMINA1();
REP=REP+1;
}
DELAY_MS(10);
TRANSFORMACION();
DELAY_MS(100);
}
}
}
```

3.9.4 Algoritmo de programación del control remoto

```
#include <16f877.h>

#device adc=8

#fuses hs,nowdt,noprotect,nolvp

#use delay(clock=20000000)

#use rs232(baud=9600,xmit = pin_e0 , rcv = pin_e1, bits=8)

#include <hdm64gs12.c>

#include <graphics.c>

#include <stdio.h>

#use standard_io(b)

#use standard_io(c)

#use standard_io(d)

#use standard_io(e)

#byte trisa =0x85

#byte porta =0x05

#define driver_a pin_c1

#define driver_b pin_c2

int8 valor_X;

int8 valor_y;

int8 temperatura=0;

int8 distancia=0;
```

```
int8 iniciar=0;
int8 tactil (int8 dato){
int8 x;
int8 y;
output_low(driver_a);
output_low(driver_b);
if (dato == 0){
while (dato == 0){
output_high(driver_a);
output_low(driver_b);
setup_adc_ports(an0_an1_an3);
setup_adc(adc_clock_internal);
      set_adc_channel(0);
      delay_ms(5);
      x=read_adc();
      if (x>10) {
      delay_ms(20);
      return(x);
     }
}
}
   else if (dato == 1) {
```

```
    while (dato == 1){
      output_low(driver_a);
      output_high(driver_b);
setup_adc_ports(an0_an1_an3);
setup_adc(adc_clock_internal);
      set_adc_channel(1);
      delay_ms(5);
      y=read_adc();
      if (y>10) {
      delay_ms(20);
      return(y);
}
}
   }
}
void bip(){
   output_high(pin_c0);
   delay_ms(200);
   output_low(pin_c0);
   delay_ms(200);
}
void pantalla_1(){
```

```
char a0[ ]="Universidad";
char a1[ ]="Politecnica Salesiana";
char a2[ ]="2014-2014";
char a3[ ]="UPS-GYQ";
glcd_init(on);
glcd_text57(25,10,a3,2,1); //25,10,a3,2,1
glcd_text57(30,30,a0,1,1);//30,30,a0,1,1
glcd_text57(0,40,a1,1,1);//0,40,a1,1,1
glcd_text57(35,50,a2,1,1);//35,50,a2,1,1
delay_ms(2000);
}
void pantalla_2(){
char a0[ ]="Inicializando";
char a1[ ]="Esperando robot";
glcd_init(on);
glcd_text57(25,20,a0,1,1);
glcd_text57(20,30,a1,1,1);
delay_ms(1000);
}
void carro(){
char a5[ ]="ADELANTAR";
char a6[ ]="RETROCEDER";
```

```
char a7[ ]="G-DERECHA";

char a8[ ]="G-IZQUIERDA";

char a9[ ]="  **MENU**  ";

char a10[ ]="Auto";

char a11[ ]="Humano";

char a12[ ]="Sensores";

    glcd_init(on);

    glcd_text57(25,5,a5,1,1);

    glcd_text57(25,20,a6,1,1);

    glcd_text57(25,35,a7,1,1);

    glcd_text57(25,50,a8,1,1);

    glcd_rect(95,5,115,10,0,1);

    glcd_rect(95,20,115,25,0,1);

    glcd_rect(95,35,115,40,0,1);

    glcd_rect(95,50,115,55,0,1);

    DELAY_MS(1000);

while(1){

valor_x=tactil(0);

delay_ms(50);

valor_y=tactil(1);

delay_ms(50);

//adelantar *120
```

```
if((valor_x >140 && valor_x < 200) && (valor_y >180 && valor_y <230)){
  bip();
  putc(3);
  delay_ms(100);
}
//retroceder
if((valor_x >140 && valor_x < 200) && (valor_y > 120 && valor_y < 170)){
  bip();
  bip();
  putc(7);
  delay_ms(100);
}
//g-derecha
if((valor_x >140 && valor_x < 200) && (valor_y > 60 && valor_y < 110)){
  bip();
  bip();
  bip();
  putc(8);
  delay_ms(100);
  }
//g-izquierda
if((valor_x >140 && valor_x < 200) && (valor_y > 15 && valor_y < 50)){
```

```
   bip();
   bip();
   bip();
   bip();
   putc(9);
   delay_ms(100);
}
//voy a menu
if((valor_x >11 && valor_x < 50) && (valor_y > 11 && valor_y < 50)){
   bip();
   putc(0);
   delay_ms(100);
   break;
}
   }
      glcd_init(on);
      glcd_text57(0,0,a9,2,1);
      glcd_text57(25,20,a10,1,1);
      glcd_text57(25,35,a11,1,1);
      glcd_text57(25,50,a12,1,1);
      glcd_rect(80,20,100,25,0,1);
      glcd_rect(80,35,100,40,0,1);
```

```
      glcd_rect(80,50,100,55,0,1);
      delay_ms(1000);
break;
}
void humano(){
char a0[ ]="MODO HUMANO";
char a1[ ]=" Menu  ";
char a2[ ]="Caminar";
char a9[ ]="  **MENU**  ";
char a10[ ]="Auto";
char a11[ ]="Humano";
char a12[ ]="Sensores";
//////////////////////////////////////////
    glcd_init(on);
    glcd_text57(0,10,a0,2,1);
    glcd_text57(10,50,a1,1,1);
    glcd_text57(75,50,a2,1,1);
    glcd_rect(20,30,40,45,1,1);
    glcd_rect(85,30,105,45,1,1);
    delay_ms(500);
while(1){
valor_x=tactil(0);
```

```
delay_ms(50);
valor_y=tactil(1);
delay_ms(50);
//menu x 10 y 50 ----- 20 30 40 45
if((valor_x >20 && valor_x < 120) && (valor_y >70 && valor_y <200)){
  bip();
  putc(0);
  delay_ms(100);
  break;
}
//caminar x 75 y 50 ----- 85,30,105,45
if((valor_x >135 && valor_x < 215) && (valor_y > 70 && valor_y < 200)){
  bip();
  putc(1);
  delay_ms(100);
 // break;
}
}
     glcd_init(on);
     glcd_text57(0,0,a9,2,1);
     glcd_text57(25,20,a10,1,1);
     glcd_text57(25,35,a11,1,1);
```

```
    glcd_text57(25,50,a12,1,1);
    glcd_rect(80,20,100,25,0,1);
    glcd_rect(80,35,100,40,0,1);
    glcd_rect(80,50,100,55,0,1);
    delay_ms(1000);
    break;
}
void sensor(){
char a1[ ]="*SENSORES*";
char a2[ ]="TEMP";
char a3[ ]="DIST";
char a4[ ]="oC";
char a5[ ]="cm";
char a9[ ]="  **MENU**  ";
char a10[ ]="Auto";
char a11[ ]="Humano";
char a12[ ]="Sensores";
char dist[3];
char temp[2];
    glcd_init(on);
    glcd_text57(0,0,a1,2,1);
    glcd_text57(35,50,a2,1,1); //20
```

```
      glcd_text57(85,50,a3,1,1);  //35
      delay_ms(500);
   sprintf(temp,"%2u",temperatura);
   glcd_text57(15,30,temp,2,1);
   glcd_text57(40,30,a4,1,1);
   sprintf(dist,"%3u",distancia);
   glcd_text57(65,30,dist,2,1);
   glcd_text57(100,30,a5,1,1);
   delay_ms(5000);
////////////////////////////////////////////////////////////////////////////
      glcd_init(on);
      glcd_text57(0,0,a9,2,1);
      glcd_text57(25,20,a10,1,1);
      glcd_text57(25,35,a11,1,1);
      glcd_text57(25,50,a12,1,1);
      glcd_rect(80,20,100,25,0,1);
      glcd_rect(80,35,100,40,0,1);
      glcd_rect(80,50,100,55,0,1);
      delay_ms(1000);
break;
}
int8 serial(){
```

```
int8 dato;
        bip();
            while(1){
            //recepcion esperar???
            dato=getc();
            if (dato !=0x0){
            bip();
            break;
            }
      }
return(dato);
}
void main(){
char a0[ ]="  **MENU**  ";
char a1[ ]="Auto";
char a2[ ]="Humano";
char a3[ ]="Sensores";
char a4[ ]="  ** robot **  ";
char a5[ ]="Listo para usar";
setup_adc_ports(an0_an1_an3);
setup_adc(adc_clock_internal);
output_low(driver_a);
```

```
output_low(driver_b);
glcd_init(on);
pantalla_1(); //llama funcion pantalla_1
pantalla_2() ;
while(1){
   iniciar=serial();
   delay_ms(10);
   if (iniciar == 2){ //recibo en int; 0x25 recibo en hex
   bip();
   break;
   }
}
//robt listo para usar
  glcd_init(on);
  glcd_text57(20,20,a4,1,1); //30
  glcd_text57(20,30,a5,1,1); //40
  delay_ms(1000);
////////////////////////////////////////////////////////////////////////////
/////////muestro menú de selección, modo auto, humano, sensores////////////////
////////////////////////////////////////////////////////////////////////////
     glcd_init(on);
     glcd_text57(0,0,a0,2,1);
```

```
    glcd_text57(25,20,a1,1,1);
    glcd_text57(25,35,a2,1,1);
    glcd_text57(25,50,a3,1,1);
    glcd_rect(80,20,100,25,0,1);
    glcd_rect(80,35,100,40,0,1);
    glcd_rect(80,50,100,55,0,1);
    delay_ms(2000);
while(1){
valor_x=tactil(0);
delay_ms(50);
valor_y=tactil(1);
delay_ms(50);

if((valor_x >125 && valor_x < 200) && (valor_y >150 && valor_y <200)){
  bip();
  putc(4);  //transforma a carro el robot       HUMANO-CARRO
  carro();
}
if((valor_x >125 && valor_x < 200) && (valor_y > 90 && valor_y < 140)){
  bip();
  putc(5);  //transforma a humano el robot        CARRO-HUMANO
  humano();
```

```
 }
 if((valor_x > 125 && valor_x < 200) && (valor_y > 30 && valor_y < 80)){
  bip();
  putc(6);  //pide valores de sensores
  bip();
  delay_ms(500);
 while(1){
   temperatura=serial();
   delay_ms(10);
   distancia=serial();
   delay_ms(10);
   if (temperatura != 0){ //recibo en int; 0x25 recibo en hex
   bip();
   break;
  }
 }
   sensor();
}
else {
output_low(pin_c0);
}
 }
```

}

3.9.5 Algoritmo de programación para la tarjeta de control de los sensores

```
INCLUDE "modedefs.bas"
adcon1=0
define OSC 4
DEFINE ADC_BITS 8       ' Set number of bits in result
DEFINE ADC_CLOCK 3     ' Set clock source (rc = 3)
DEFINE ADC_SAMPLEUS 50 ' Set sampling time in microseconds
a VAR PORTC.3
b VAR PORTC.4
c VAR PORTB.5  ;16
d VAR PORTC.2
ping var portb.2   ;23  TRIGGER
ping1 var portc.0   ;11  ECO
Scale var word                          'raw x 10.00 =uS
Cm var word
CEMT VAR BYTE                           '1/29.034(ancho**)
tem var word
ldr var word
Dist var word                           'raw medida
cem var word
in var byte
```

```
;cemt var byte

led var portb.0

led1 var portb.1

Scale = 10

cm = 2257

 MAIN:

high b

high a

high c

high d

serin portc.7,t9600,in

if in=1 then gosub mensaje1

if in=2 then gosub mensaje2

if in=3 then gosub mensaje3

if in=5 then gosub mensaje4

if in=4 then goTO DATOS_1

goto main

ilu:

adcin 0,ldr

ldr=ldr*100

ldr=ldr/255

return
```

```
ENSAJE1:
high led
pause 1000
low led
pause 200
low a
high b
high c
high d
PAUSE 1000
RETURN
MENSAJE2:
high led
pause 1000
low led
pause 200
low b
high a
high c
high d
PAUSE 1000
RETURN
```

```
MENSAJE3:
high led
pause 1000
low led
pause 200
low c
high b
high a
high d
PAUSE 1000
RETURN
MENSAJE4:
high led
pause 1000
low led
pause 200
low d
high c
high a
high b
PAUSE 1000
RETURN
```

```
MENSAJE0:
high c
high b
high a
high d
RETURN
DATOS_1:
high b
high C
low A
high d
high led
pause 1000
low led
 PULSOUT Ping, 5             ' Send short pulse to Ping
 PULSIN Ping, 1, dist        ' Wait for echo
 cem = 890 ** dist  ' Convert to inches
' Display result
 PAUSE 200
 CEMT=CEM*10
  adcin 1,tem
 tem=tem*2
```

```
if cemt < 50 then cemt=cemt+10
if (cemt > 50) and (cemt < 100) then cemt=cemt+20
 serout portc.6,T9600,[CEMT]
 PAUSE 1500
 serout portc.6,T9600,[TEM]
 PAUSE 1500
   if CEM < 5 THEN  GOTO ENVIO
Serout portc.6,T9600,[2]
goto main
ENVIO:
high led1
pause 5000
low led1
pause 500
serout portc.6,T9600,[5]
GOTO MAIN
 end
```

CONCLUSIONES

- La programación de microcontroladores PIC en lenguaje C permite el desarrollo de aplicaciones más complejas, optimizando mayor cantidad de recursos de memoria que utilizar lenguajes de programación en Basic.

- El diseño estructural de cualquier tipo de robot, representa una variable muy importante de considerar al momento de realizar un diseño.

- La comunicación inalámbrica por Xbee, es de forma muy segura y de largo alcance que permite realizar comunicaciones seguras.

- El avance académico que se puede tener al experimentar con robots de esta categoría.

- Se ha podido cumplir con el proyecto propuesto en su totalidad, logrando desarrollar dos robots que tengan la capacidad de modificar su estructura sin perder la motricidad, teniendo sus limitaciones como las que fueron mencionadas con anterioridad.

RECOMENDACIONES

- Dimensionar muy bien los servomotores según su capacidad de torque antes de colocarlos sobre una estructura.

- Seleccionar adecuadamente el tipo de caja de engranaje que el servo vaya a tener, ya que no siempre es suficiente el torque del mismo.

- Utilizar materiales rigidos y livianos como el aluminio cuando se vaya a construir un robot que se exponga a golpes externos.

- Realizar una tabla de variables o periféricos que se vayan a medir o utilizar en el diseño antes de seleccionar cualquier tipo de microcontrolador.

- Utilizar microcontroladores con salidas de HPWM, para simplificar el algoritmo de programación.
- Probar nuevas rutinas de programación y nuevas formas de transformaciones con los robots desarrollados.

- No sobre utilizar a los robots, debido a que podrían causar daños en sus servomotores.

- En caso de daño de algunos de sus servomotores, cambiarlos con los mismos especificados, ya que de lo contrario el algoritmo de programación no los reconocerá.

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Barrientos, A., Peñín, L. F., Balaguer, C., & Aracil, R. (1997). *Fundamentos de Robótica.* Madrid, España: McGraw-Hill.

Breijo, E. G. (2008). *Compilador C CCS y Simulador PROTEUS para Microcontroladores PIC.* (Alfaomega, Ed.) Barcelona, México: Marcombo.

Carlos, R. (2008). *Microcontroladores PIC Programación en Basic* (Vol. 1). Quito: rispergraf.

Cortés, F. R. (2010). *Matlab aplicado a Robótica y Mecatrónica.* (Marcombo, Ed.) Alfaomega.

Craig, J. J. (2006). *Robótica* (Tercera ed.). Pearson.

electronics, m. (2009). *Xbee.cl Módulos de Transmisión Inalámbrica.* Recuperado el 10 de Enero de 2014, de http://www.xbee.cl/

Esteva, C. B. (s.f.). *Conexión electrónica.* Obtenido de www.conexionelectronica.com

HobbyKing. (s.f.). Recuperado el 5 de Diciembre de 2013, de http://www.hobbyking.com/hobbycity/store/uh_listCategoriesAndProducts.asp?catname=HD+Servo&idCategory=281&ParentCat=189&gclid=COTu1463jcECFajm7AodoV8Axg

MicrochipTechnology. (2003). alldatasheet. Recuperado el 21 de Junio de 2014, de http://pdf1.alldatasheet.com/datasheet-pdf/view/560054/MICROCHIP/PIC16F877A-E.html

Mikroelectrónica. (s.f.). Recuperado el 2014, de http://www.mikroe.com/

Mikroelectrónica. (s.f.). Mikroelectrónica. Obtenido de http://www.mikroe.com/

Parallax. (s.f.). Obtenido de http://www.parallax.com/product/28015

Semiconductor, D. (s.f.). alldatasheet. Recuperado el 14 de Julio de 2014, de http://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html

Semiconductor, N. (Noviembre de 2000). alldatasheet. Recuperado el 20 de Febrero de 2014, de http://pdf1.alldatasheet.com/datasheet-pdf/view/8866/NSC/LM35.html

ServoDatabase. (s.f.). Recuperado el 6 de Enero de 2014, de http://www.servodatabase.com/servos/hitec

Superrobótica. (s.f.). Obtenido de http://www.superrobotica.com/Servosrc.htm

System, A. M. (s.f.). alldatasheet. Recuperado el 2 de Enero de 2014, de http://pdf1.alldatasheet.com/datasheet-pdf/view/8866/NSC/LM35.html

tiposde. (s.f.). Recuperado el 10 de Julio de 2014, de http://www.tiposde.org/general/460-tipos-de-robots/

ANEXOS

ANEXO 1: ENCUESTA REALIZADA A ESTUDIANTES DE CICLOS SUPERIORES (8VO – 9NO – 10MO)

Datos preliminares:

Población total encuestada:

Modelo de Encuesta:

Nombre:	
Carrera:	
Semestre:	

ROBOT CON APLICACIONES DE TRANSFORMACION Y MOVIMIENTO BAJO UNA MISMA ESTRUCTURA

Según lo expuesto sobre este tema de tesis por favor resaltar las siguientes observaciones pertinentes.

Observaciones relacionadas al Robot Humanoide - Auto	
Observaciones relacionadas al Robot Hexápodo - Auto	
Cree que los robots le ayudarían a fortalecer los conocimientos teóricos con la práctica sobre ellos en la materia de Robótica?	Si ( ) No ( )
Cree que los robots le ayudarían a fortalecer los conocimientos teóricos con la práctica sobre ellos en la materia de Microcontroladores II?	Si ( ) No ( )
Cree que los robots le ayudarían a fortalecer los conocimientos teóricos con la práctica sobre ellos en la materia de Teoría del Diseño?	Si ( ) No ( )

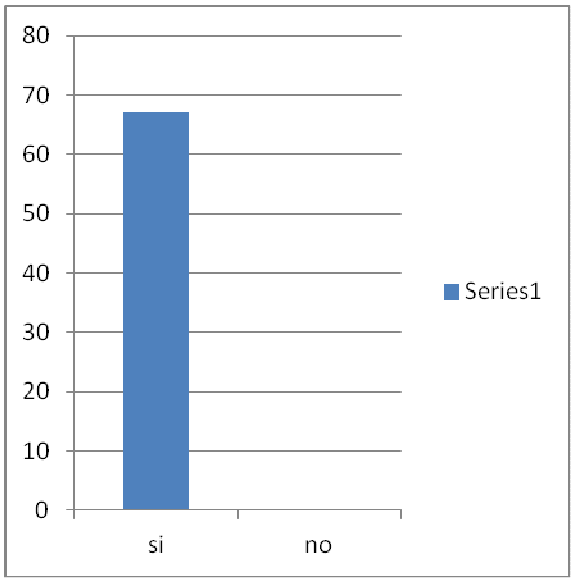
Cree usted que los Robots están aptos para participar en algún concurso de robótica?	Si ( ) No ( )
Le parecen los robots didácticos?	Si ( ) No ( )
Había experimentado alguna vez de forma personal con robots de ésta índole?	Si ( ) No ( )
Le gustaría aprender a desarrollar robots de esta categoría?	Si ( ) No ( )
Adicionaría algo al proyecto? Si la respuesta es afirmativa, ¿Qué sería?	Si ( ) No ( )

Tabulación de datos.

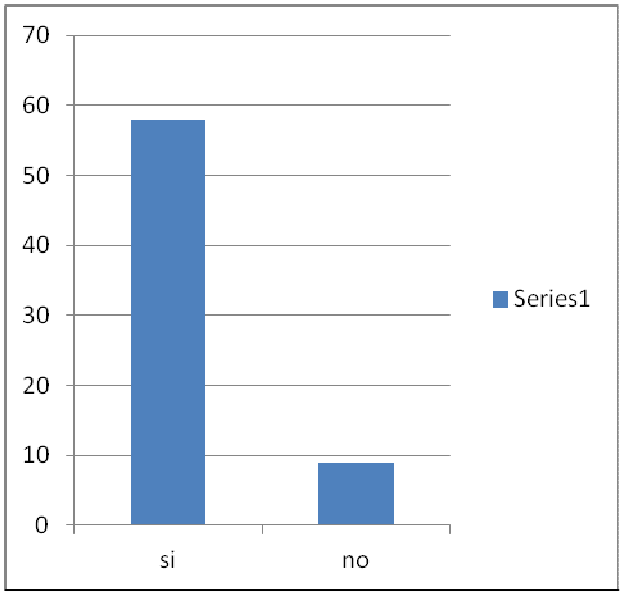
Entre las observaciones realizadas por los encuestados resaltamos las siguientes:

- Realizar más rutinas de movimientos para los dos robots.
- Aumentar la estabilidad.
- Aumentarle la velocidad al momento de caminar en los dos robots.

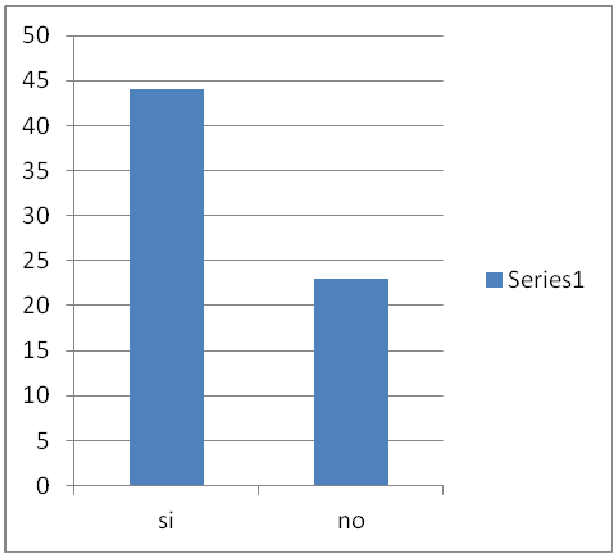
- Cree que los robots le ayudarían a fortalecer los conocimientos teóricos con la práctica sobre ellos en la materia de Robótica?



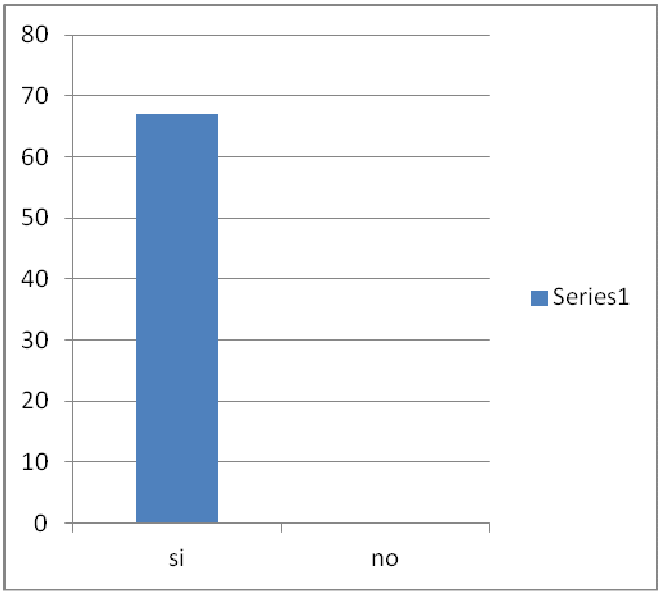
- Cree que los robots le ayudarían a fortalecer los conocimientos teóricos con la práctica sobre ellos en la materia de Microcontroladores II?



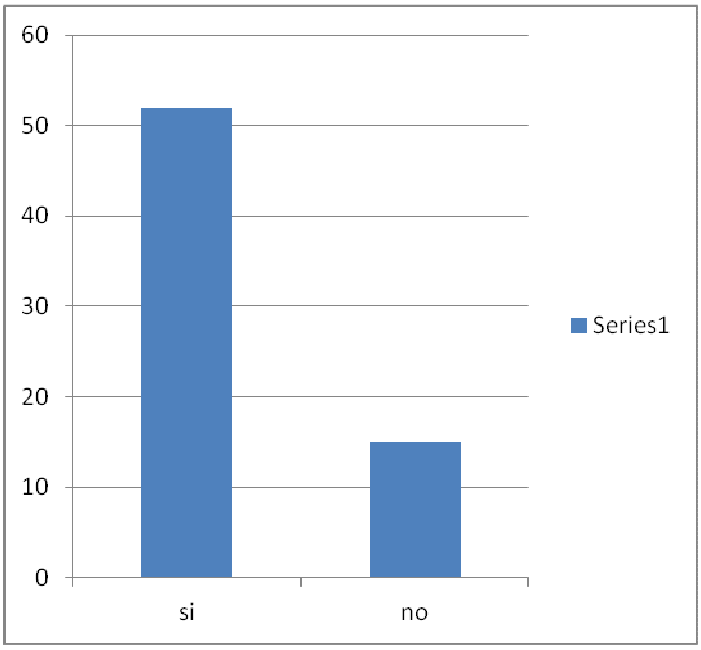
- Cree que los robots le ayudarían a fortalecer los conocimientos teóricos con la práctica sobre ellos en la materia de Teoría del Diseño?



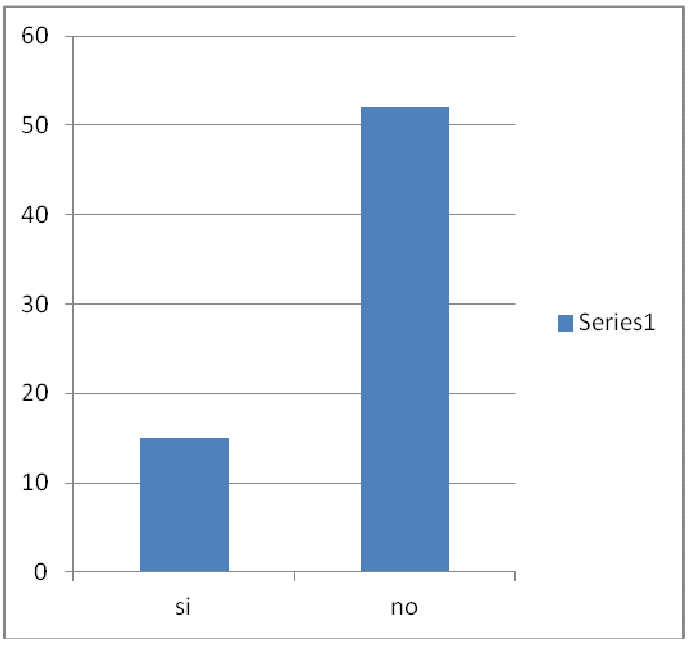
- Cree usted que los Robots están aptos para participar en algún concurso de robótica?



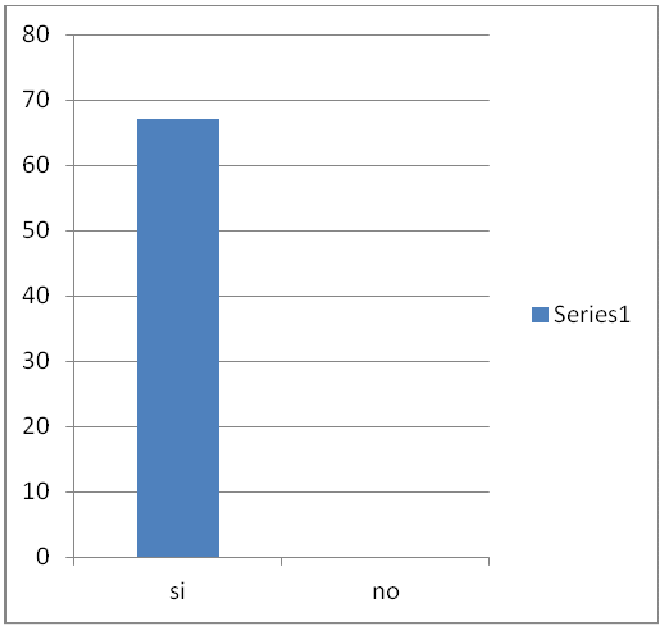
- Le parecen los robots didácticos?



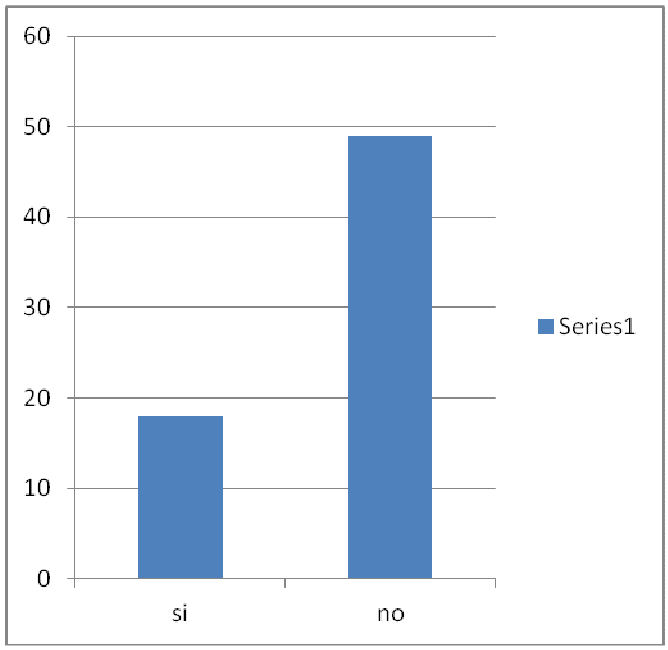
- Había experimentado alguna vez de forma personal con robots de ésta índole?



- Le gustaría aprender a desarrollar robots de esta categoría?



- Adicionaría algo al proyecto? Si la respuesta es afirmativa, ¿Qué sería?



Entre las respuestas de las personas encuestadas, resaltamos las siguientes:

- Insertarle más cantidad de sensores.
- Realizar una aplicación para controlar los robots mediantes un Smart Phone.
- Adicionarle un GPS.
- Adicionarle una cámara de video.

ANEXO 2: SUBRUTINAS DE PROGRAMACIÓN DE LAS TARJETAS DE CONTROL PRINCIPAL

SUBRUTINAS DE PROGRAMACIÓN DEL ROBOT TRANSFORMER HEXÁPODO - AUTO

- **Subrutina para generación de pulsos del robot Hexápodo – Auto**

```
//funcion de generacion de pulsos
#define HOMBRO_1 PIN_B5
 #define CODO_1 PIN_B4
 #define PIE_1 PIN_B3
 #define RUEDA_1 PIN_B2

 #define HOMBRO_2 PIN_A5
 #define CODO_2 PIN_B7
 #define PIE_2 PIN_A3
 #define RUEDA_2 PIN_C7

 #define HOMBRO_3 PIN_B0
 #define CODO_3 PIN_B1
 #define PIE_3 PIN_D6
 #define RUEDA_3 PIN_D5

 #define HOMBRO_4 PIN_C6
 #define CODO_4 PIN_C5
 #define PIE_4 PIN_C4
 #define RUEDA_4 PIN_C3
```

```
#define HOMBRO_5 PIN_D4
#define CODO_5 PIN_D3
#define PIE_5 PIN_D2
#define RUEDA_5 PIN_D1

#define HOMBRO_6 PIN_C2
#define CODO_6 PIN_C1
#define PIE_6 PIN_C0
#define RUEDA_6 PIN_D0
#DEFINE DOMO PIN_B6

VOID pulsoss (int8 servo, int8 valor){
int8  serv;
int16 pulso;
INT8   BAJO;
if (servo==1) serv = HOMBRO_1;
if (servo==2) serv = CODO_1;
if (servo==3) serv = PIE_1;
if (servo==4) serv = RUEDA_1;
if (servo==5) serv = HOMBRO_2;
if (servo==6) serv = CODO_2;
if (servo==7) serv = PIE_2;
if (servo==8) serv = RUEDA_2;
if (servo==9) serv = HOMBRO_3;
if (servo==10) serv = CODO_3;
if (servo==11) serv = PIE_3;
```

```
if (servo==12) serv = RUEDA_3;
if (servo==13) serv = HOMBRO_4;
if (servo==14) serv = CODO_4;
if (servo==15) serv = PIE_4;
if (servo==16) serv = RUEDA_4;
if (servo==17) serv = HOMBRO_5;
if (servo==18) serv = CODO_5;
if (servo==19) serv = PIE_5;
if (servo==20) serv = RUEDA_5;
if (servo==21) serv = HOMBRO_6;
if (servo==22) serv = CODO_6;
if (servo==23) serv = PIE_6;
if (servo==24) serv = RUEDA_6;
if (servo==25) serv = DOMO;
pulso=0;
   output_low(serv);
//delay_ms(199);

   pulso=(11.11*valor)+500.2; //angulo inicial en valores eje 1500
   BAJO=18-(PULSO/1000);
   output_high(serv);//rodi der
   delay_us(pulso);
   output_low(serv);
  // DELAY_MS(BAJO);
   BREAK;
```

```
}
```

- **Subrutina de posición inicial**

```
INT8 V;
int8 I;
int8 W,temp,dist,lux;
//INT8 VALOR;
VOID INICIAL(){
I=0;
while(I<1){
W=2;
PUTC(W);
DELAY_MS(4000);
I=I+1;
}
I=0;
while(I<50){
 pulsoss(1,63);  // hombro_1
  pulsoss(3,0);   // pata_1
  pulsoss(2,90);  // codo_1
 // pulsoss(4,75);// rueda_1

   pulsoss(5,107); // hombro_2
   pulsoss(7,180);// pata_2
   pulsoss(6,90); // codo_2
 //  pulsoss(8,75);// rueda_2
```

```
   pulsoss(9,94); // hombro_3
   pulsoss(10,90);// codo_3
   pulsoss(11,0);// pata_3
   // pulsoss(12,75);// rueda_3

   pulsoss(13,85); // hombro_4
   pulsoss(14,90); // codo_4 20
   pulsoss(15,170); // pata_4
  // pulsoss(16,75);// rueda_4

   pulsoss(17,95); // hombro_5
   pulsoss(19,0); // pata_5
   pulsoss(18,90); // codo_5
// pulsoss(20,75);// rueda_5

   pulsoss(21,70); // hombro_6
   pulsoss(23,180); // pata_6
   pulsoss(22,90); // codo_6
  // pulsoss(24,75);// rueda_6

I=I+1;
}
I=0;
WHILE(I<50){
  pulsoss(1,63);  // hombro_1
```

```
 pulsoss(3,0);   // pata_1
 pulsoss(2,120);  // codo_1
// pulsoss(4,75);// rueda_1

  pulsoss(5,107); // hombro_2
  pulsoss(7,180);// pata_2
  pulsoss(6,45); // codo_2
//  pulsoss(8,75);// rueda_2

  pulsoss(9,94); // hombro_3
  pulsoss(10,140);// codo_3
  pulsoss(11,0);// pata_3
  // pulsoss(12,75);// rueda_3

  pulsoss(13,85); // hombro_4
  pulsoss(14,40); // codo_4 20
  pulsoss(15,170); // pata_4
 // pulsoss(16,75);// rueda_4

  pulsoss(17,95); // hombro_5
  pulsoss(19,0); // pata_5
  pulsoss(18,135); // codo_5
// pulsoss(20,75);// rueda_5

  pulsoss(21,70); // hombro_6
  pulsoss(23,180); // pata_6
```

```
   pulsoss(22,60); // codo_6
  // pulsoss(24,75);// rueda_6
I=I+1;
}
I=0;
//WHILE(I<4){
W=4;
PUTC(w);
dist=GETC();
output_high(PIN_E2);

while( dist!= 50){
W=4;
PUTC(w);
dist==GETC();

V=100;
 DELAY_MS(v);
 pulsoss(25,0); // DOMO
 DELAY_MS(v);
 pulsoss(25,10); // DOMO
 DELAY_MS(V);
 pulsoss(25,20); // DOMO
 DELAY_MS(V);
  pulsoss(25,30); // DOMO
 DELAY_MS(V);
```

```
 pulsoss(25,40); // DOMO
DELAY_MS(V);
 pulsoss(25,50); // DOMO
DELAY_MS(V);
 pulsoss(25,60); // DOMO
DELAY_MS(V);
 pulsoss(25,70); // DOMO
DELAY_MS(V);
 pulsoss(25,80); // DOMO
DELAY_MS(V);
pulsoss(25,90); // DOMO
DELAY_MS(V);
pulsoss(25,100); // DOMO
DELAY_MS(V);
 pulsoss(25,110); // DOMO
DELAY_MS(V);
 pulsoss(25,120); // DOMO
DELAY_MS(V);
 pulsoss(25,130); // DOMO
DELAY_MS(V);
 pulsoss(25,140); // DOMO
DELAY_MS(V);
 pulsoss(25,150); // DOMO
DELAY_MS(V);
 pulsoss(25,160); // DOMO
DELAY_MS(V);
```

```
 pulsoss(25,180); // DOMO
 DELAY_MS(100);
 I=I+1;
}
while(dist==50){
I=0;

WHILE(I<10){
W=4;
PUTC(w);
dist==GETC();

output_LOW(PIN_E2);
delay_ms(300);
output_high(PIN_E2);
delay_ms(300);
//V=100;
// pulsoss(25,60); // DOMO
 //DELAY_MS(100);
 I=I+1;
}
break;
}
}
```

- **Subrutina de movimiento delantero en modo Auto**

```
int8 Z;
VOID CARRO_ADELANTE(){
Z=0;
WHILE(Z<150){
  pulsoss(4,180);// rueda_1
  pulsoss(8,0);// rueda_2
  pulsoss(12,180);// rueda_3
  pulsoss(16,0);// rueda_4
  pulsoss(20,180);// rueda_5
  pulsoss(24,0);// rueda_6

Z=Z+1;
}
BREAK;
}
```

- **Subrutina de movimiento hacia atrás en modo Auto**

```
int8 Y;
VOID CARRO_ATRAS(){
Y=0;
WHILE(Y<150){
  pulsoss(4,0);// rueda_1
  pulsoss(8,180);// rueda_2
  pulsoss(12,0);// rueda_3
```

```
  pulsoss(16,180);// rueda_4
  pulsoss(20,0);// rueda_5
  pulsoss(24,180);// rueda_6
Y=Y+1;
}
BREAK;
}
```

- **Subrutina de movimiento hacia la derecha en modo Auto**

```
VOID CARRO_DERECHA(){
int16 Z;
Z=0;
WHILE(Z<250){
  pulsoss(4,0);// rueda_1
  pulsoss(8,0);// rueda_2
  pulsoss(12,0);// rueda_3
  pulsoss(16,0);// rueda_4
  pulsoss(20,0);// rueda_5
  pulsoss(24,0);// rueda_6
Z=Z+1;
}
BREAK;
}
```

- **Subrutina de movimiento hacia la izquierda en modo Auto**

```
VOID CARRO_IZQUIERDA(){
int16 G;
G=0;
WHILE(G<250){
  pulsoss(4,180);// rueda_1
  pulsoss(8,180);// rueda_2
  pulsoss(12,180);// rueda_3
   pulsoss(16,180);// rueda_4
  pulsoss(20,180);// rueda_5
  pulsoss(24,180);// rueda_6
G=G+1;
}
BREAK;
}
```

Subrutina de transformación de modo Auto a modo Hexápodo

```
int16 X;
int16 A;
VOID PARADO(){
X=0;
while(X<1){
A=3;
PUTC(A);
DELAY_MS(4000);
```

```
X=X+1;
}
X=0;
while(X<1){
A=1;
PUTC(A);
DELAY_MS(4000);
X=X+1;
}



X=0;
WHILE(X<50){
  pulsoss(1,63);  // hombro_1
  pulsoss(3,0);   // pata_1
  pulsoss(2,90);  // codo_1
  pulsoss(5,107); // hombro_2
   pulsoss(7,180);// pata_2
   pulsoss(6,90); // codo_2
 //  pulsoss(8,75);// rueda_2
   pulsoss(9,94); // hombro_3
   pulsoss(10,90);// codo_3
   pulsoss(11,0);// pata_3
   // pulsoss(12,75);// rueda_3
   pulsoss(13,85); // hombro_4
```

```
   pulsoss(14,90); // codo_4 20
   pulsoss(15,170); // pata_4
  // pulsoss(16,75);// rueda_4
  pulsoss(17,95); // hombro_5
   pulsoss(19,0); // pata_5
   pulsoss(18,90); // codo_5
// pulsoss(20,75);// rueda_5
   pulsoss(21,70); // hombro_6
   pulsoss(23,180); // pata_6
   pulsoss(22,90); // codo_6
  // pulsoss(24,75);// rueda_6
X=X+1;
}
X=0;
WHILE(X<10){
   pulsoss(1,85);  // hombro_1
   pulsoss(2,50);  // codo_1
   pulsoss(3,0);   // pata_1
  // pulsoss(4,75);// rueda_1
   pulsoss(5,90); // hombro_2
   pulsoss(6,135); // codo_2
   pulsoss(7,175);// pata_2
 //  pulsoss(8,75);// rueda_2
   pulsoss(9,94); // hombro_3
   pulsoss(10,55);// codo_3
   pulsoss(11,0);// pata_3
```

```
  // pulsoss(12,75);// rueda_3
   pulsoss(13,85); // hombro_4
   pulsoss(14,135); // codo_4
   pulsoss(15,180); // pata_4
  // pulsoss(16,75);// rueda_4
   pulsoss(17,80); // hombro_5
   pulsoss(18,45); // codo_5
   pulsoss(19,0); // pata_5
  // pulsoss(20,75);// rueda_5
   pulsoss(21,90); // hombro_6
   pulsoss(22,135); // codo_6
   pulsoss(23,180); // pata_6
  // pulsoss(24,75);// rueda_6

  X=X+1;
}
X=0;
WHILE(X<30){
   pulsoss(1,85);  // hombro_1
   pulsoss(2,15);  // codo_1
   pulsoss(3,180);   // pata_1
  // pulsoss(4,75);// rueda_1
   pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,30);// pata_2
 //  pulsoss(8,75);// rueda_2
```

```
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
  // pulsoss(12,75);// rueda_3
  pulsoss(13,85); // hombro_4
  pulsoss(14,160); // codo_4
  pulsoss(15,40); // pata_4
 // pulsoss(16,75);// rueda_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,10); // codo_5
  pulsoss(19,135); // pata_5
 // pulsoss(20,75);// rueda_5
  pulsoss(21,90); // hombro_6
  pulsoss(22,160); // codo_6
  pulsoss(23,10); // pata_6
 X=X+1;
}
 BREAK;
}
```

- **Subrutina para caminar en modo Hexápodo**

```
VOID CAMINA(){
int16 B;
int8 C;
int16 J;
int16 K;
```

```
//int16 L;
int8 O ;
int8 P ;
int8 Q ;
int8 R ;
INT8 REP;
INT8 TIME;
REP=0;
//WHILE(REP<5){
B=0;
C=0;
O=0;
J=0;
K=0;
P=0;
Q=0;
R=0;
TIME=10;
//WHILE(R<20){
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////
////////////////////////////////////////////////////////////// RUTINA PATA 1//
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
WHILE(C<TIME){
  pulsoss(2,90);  // codo_1 45
  pulsoss(1,45);  // hombro_1 135
```

```
   pulsoss(3,180);   // pata_1
// pulsoss(4,75);// rueda_1
      pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,30);// pata_2
 //  pulsoss(8,75);// rueda_2
   pulsoss(9,94); // hombro_3
   pulsoss(10,20);// codo_3
   pulsoss(11,145);// pata_3
   // pulsoss(12,75);// rueda_3
   pulsoss(13,85); // hombro_4
   pulsoss(14,160); // codo_4
   pulsoss(15,40); // pata_4
  // pulsoss(16,75);// rueda_4

   pulsoss(17,80); // hombro_5
   pulsoss(18,10); // codo_5
   pulsoss(19,135); // pata_5
  // pulsoss(20,75);// rueda_5
   pulsoss(21,90); // hombro_6
   pulsoss(22,160); // codo_6
   pulsoss(23,10); // pata_6
  // pulsoss(24,75);// rueda_6
  C=C+1;
}
WHILE(B<TIME){
```

```
  pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1 135
  pulsoss(3,180);   // pata_1
 // pulsoss(4,75);// rueda_1
  pulsoss(5,90); // hombro_2
  pulsoss(6,160); // codo_2
  pulsoss(7,30);// pata_2
//  pulsoss(8,75);// rueda_2
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
  // pulsoss(12,75);// rueda_3
  pulsoss(13,85); // hombro_4
  pulsoss(14,160); // codo_4
  pulsoss(15,40); // pata_4
 // pulsoss(16,75);// rueda_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,10); // codo_5
  pulsoss(19,135); // pata_5
 // pulsoss(20,75);// rueda_5
  pulsoss(21,90); // hombro_6
  pulsoss(22,160); // codo_6
  pulsoss(23,10); // pata_6
 // pulsoss(24,75);// rueda_6
 B=B+1;
}
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////

//////////////////////////////////////////////////////////// RUTINA PATA 6 //
////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

B=0;

WHILE(B<TIME){

  pulsoss(2,15);  // codo_1 45

  pulsoss(1,45);  // hombro_1 135

  pulsoss(3,180);   // pata_1

 // pulsoss(4,75);// rueda_1

  pulsoss(5,90); // hombro_2

  pulsoss(6,160); // codo_2

  pulsoss(7,30);// pata_2

//  pulsoss(8,75);// rueda_2



  pulsoss(9,94); // hombro_3

  pulsoss(10,20);// codo_3

  pulsoss(11,145);// pata_3

  // pulsoss(12,75);// rueda_3

  pulsoss(13,85); // hombro_4

  pulsoss(14,160); // codo_4

  pulsoss(15,40); // pata_4

 // pulsoss(16,75);// rueda_4

  pulsoss(17,80); // hombro_5

  pulsoss(18,10); // codo_5
```

```
  pulsoss(19,135); // pata_5
 // pulsoss(20,75);// rueda_5
  pulsoss(22,90); // codo_6 165
  pulsoss(21,110); // hombro_6
  pulsoss(23,10); // pata_6
 // pulsoss(24,75);// rueda_6
 B=B+1;
}
B=0;
WHILE(B<TIME){

  pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1 135
  pulsoss(3,180);   // pata_1
 // pulsoss(4,75);// rueda_1
  pulsoss(5,90); // hombro_2
  pulsoss(6,160); // codo_2
  pulsoss(7,30);// pata_2
//  pulsoss(8,75);// rueda_2
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
  // pulsoss(12,75);// rueda_3
  pulsoss(13,85); // hombro_4
  pulsoss(14,160); // codo_4
  pulsoss(15,40); // pata_4
```

```
  // pulsoss(16,75);// rueda_4
   pulsoss(17,80); // hombro_5
   pulsoss(18,10); // codo_5
   pulsoss(19,135); // pata_5
  // pulsoss(20,75);// rueda_5
   pulsoss(22,160); // codo_6
   pulsoss(21,110); // hombro_6
   pulsoss(23,10); // pata_6
  // pulsoss(24,75);// rueda_6
  B=B+1;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
////////////////////////////////////////////////////////////// RUTINA PATA 4//
//////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
B=0;


J=0;
WHILE(J<TIME){
   pulsoss(2,15);  // codo_1 45
   pulsoss(1,45);  // hombro_1
   pulsoss(3,180);   // pata_1
  // pulsoss(4,75);// rueda_1
   pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,30);// pata_2
```

```
//  pulsoss(8,75);// rueda_2
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
  // pulsoss(12,75);// rueda_3
  pulsoss(14,90); // codo_4 135
  pulsoss(13,135); // hombro_4
  pulsoss(15,15); // pata_4
 // pulsoss(16,75);// rueda_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,10); // codo_5
  pulsoss(19,135); // pata_5
 // pulsoss(20,75);// rueda_5
  pulsoss(22,160); // codo_6
  pulsoss(21,110); // hombro_6
  pulsoss(23,10); // pata_6
 // pulsoss(24,75);// rueda_6

 J=J+1;
}
J=0;
WHILE(K<TIME){
  pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1
  pulsoss(3,180);   // pata_1
 // pulsoss(4,75);// rueda_1
```

```
   pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,30);// pata_2
 //  pulsoss(8,75);// rueda_2
  pulsoss(9,94); // hombro_3
   pulsoss(10,20);// codo_3
   pulsoss(11,145);// pata_3
   // pulsoss(12,75);// rueda_3
    pulsoss(14,160); // codo_4 135
   pulsoss(13,135); // hombro_4
   pulsoss(15,40); // pata_4
  // pulsoss(16,75);// rueda_4
   pulsoss(17,80); // hombro_5
   pulsoss(18,10); // codo_5
   pulsoss(19,135); // pata_5
  // pulsoss(20,75);// rueda_5
   pulsoss(22,160); // codo_6
   pulsoss(21,110); // hombro_6
   pulsoss(23,10); // pata_6
  // pulsoss(24,75);// rueda_6
 K=K+1;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
////////////////////////////////////////////////////////////// RUTINA PATA 5//
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
```

```
WHILE(O<TIME){
   pulsoss(2,15);  // codo_1 45
   pulsoss(1,45);  // hombro_1
   pulsoss(3,180);   // pata_1
  // pulsoss(4,75);// rueda_1
   pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,30);// pata_2
 //  pulsoss(8,75);// rueda_2
   pulsoss(9,94); // hombro_3
   pulsoss(10,20);// codo_3
   pulsoss(11,145);// pata_3
   // pulsoss(12,75);// rueda_3
    pulsoss(14,160); // codo_4 135
   pulsoss(13,135); // hombro_4
   pulsoss(15,40); // pata_4
  // pulsoss(16,75);// rueda_4
    pulsoss(18,90); // codo_5  0
    pulsoss(17,60); // hombro_5
    pulsoss(19,165); // pata_5
  // pulsoss(20,75);// rueda_5
  pulsoss(22,160); // codo_6
   pulsoss(21,110); // hombro_6
   pulsoss(23,10); // pata_6
  // pulsoss(24,75);// rueda_6
  O=O+1;
```

```
}
WHILE(P<TIME){
  pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1
  pulsoss(3,180);   // pata_1
 // pulsoss(4,75);// rueda_1
  pulsoss(5,90); // hombro_2
  pulsoss(6,160); // codo_2
  pulsoss(7,30);// pata_2
 //  pulsoss(8,75);// rueda_2
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
  // pulsoss(12,75);// rueda_3
   pulsoss(14,160); // codo_4 135
  pulsoss(13,135); // hombro_4
  pulsoss(15,40); // pata_4
 // pulsoss(16,75);// rueda_4
  pulsoss(18,10); // codo_5  0
  pulsoss(17,60); // hombro_5
  pulsoss(19,135); // pata_5
 // pulsoss(20,75);// rueda_5
  pulsoss(22,160); // codo_6
  pulsoss(21,110); // hombro_6
  pulsoss(23,10); // pata_6
 // pulsoss(24,75);// rueda_6
```

```
 P=P+1;
}
 BREAK;
}
```

Subrutina Camina 1 en modo Hexapodo

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////
/////////////////////////////////////////////////////////// RUTINA PATAS 2,
//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////VOID CAMINA1(){
VOID CAMINA1(){
int8 Q ;
INT8 TIME;
TIME=10;
Q=0;
WHILE(Q<TIME){
   pulsoss(2,15);  // codo_1 45
   pulsoss(1,45);  // hombro_1
   pulsoss(3,180);   // pata_1
  // pulsoss(4,75);// rueda_1


   pulsoss(6,90); // codo_2 165
   pulsoss(5,135); // hombro_2 45
   pulsoss(7,5);// pata_2
 //  pulsoss(8,75);// rueda_2
   pulsoss(9,94); // hombro_3
```

```
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
  // pulsoss(12,75);// rueda_3
   pulsoss(14,160); // codo_4 135
  pulsoss(13,135); // hombro_4
  pulsoss(15,40); // pata_4
 // pulsoss(16,75);// rueda_4
  pulsoss(18,10); // codo_5  0
  pulsoss(17,60); // hombro_5
  pulsoss(19,135); // pata_5
  pulsoss(22,160); // codo_6
  pulsoss(21,110); // hombro_6
  pulsoss(23,10); // pata_6
 Q=Q+1;
}
Q=0;
WHILE(Q<TIME){
    pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1
  pulsoss(3,180);   // pata_1
  pulsoss(6,160); // codo_2 165
  pulsoss(5,135); // hombro_2
  pulsoss(7,30);// pata_2
 pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,145);// pata_3
```

```
   pulsoss(14,160); // codo_4 135
  pulsoss(13,135); // hombro_4
  pulsoss(15,40); // pata_4
  pulsoss(18,10); // codo_5  0
  pulsoss(17,60); // hombro_5
  pulsoss(19,135); // pata_5
  pulsoss(22,160); // codo_6
  pulsoss(21,110); // hombro_6
  pulsoss(23,10); // pata_6
 Q=Q+1;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
////////////////////////////////////////////////////////////// RUTINA PATA 3//
//////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
Q=0;
WHILE(Q<TIME){
  pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1
  pulsoss(3,180);   // pata_1
  pulsoss(6,160); // codo_2 165
  pulsoss(5,135); // hombro_2
  pulsoss(7,30);// pata_2
 pulsoss(10,90);// codo_3 15
  pulsoss(9,40); // hombro_3
  pulsoss(11,165);// pata_3
```

```
   pulsoss(14,160); // codo_4 135
  pulsoss(13,135); // hombro_4
  pulsoss(15,40); // pata_4
    pulsoss(18,10); // codo_5  0
  pulsoss(17,60); // hombro_5
  pulsoss(19,135); // pata_5
    pulsoss(22,160); // codo_6
  pulsoss(21,110); // hombro_6
  pulsoss(23,10); // pata_6
 Q=Q+1;
}
Q=0;
WHILE(Q<TIME){
   pulsoss(2,15);  // codo_1 45
  pulsoss(1,45);  // hombro_1
  pulsoss(3,180);   // pata_1
 pulsoss(6,160); // codo_2 165
  pulsoss(5,135); // hombro_2
  pulsoss(7,30);// pata_2
   pulsoss(10,20);// codo_3 15
  pulsoss(9,40); // hombro_3
  pulsoss(11,145);// pata_3
 pulsoss(14,160); // codo_4 135
  pulsoss(13,135); // hombro_4
  pulsoss(15,40); // pata_4
 pulsoss(18,10); // codo_5  0
```

```
   pulsoss(17,60); // hombro_5
   pulsoss(19,135); // pata_5
  pulsoss(22,160); // codo_6
   pulsoss(21,110); // hombro_6
   pulsoss(23,10); // pata_6
    Q=Q+1;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////
////////////////////////////////////////////////////////////// RUTINA PATA //
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////
Q=0;
while(Q<60){
   pulsoss(1,85);  // hombro_1
   pulsoss(2,15);  // codo_1
   pulsoss(3,180);   // pata_1
    pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,30);// pata_2
   pulsoss(9,94); // hombro_3
   pulsoss(10,20);// codo_3
   pulsoss(11,145);// pata_3
  pulsoss(13,85); // hombro_4
   pulsoss(14,160); // codo_4
   pulsoss(15,40); // pata_4
    pulsoss(17,80); // hombro_5
```

```
   pulsoss(18,10); // codo_5
   pulsoss(19,135); // pata_5
  pulsoss(21,90); // hombro_6
   pulsoss(22,160); // codo_6
   pulsoss(23,10); // pata_6
Q+=1;
}
 BREAK;
}////////////////////////////////////////////////////////////////////////
```

- **Subrutina de transformación de modo Hexápodo a modo Auto**

```
VOID TRANSFORMACION(){
int8 Q ;
INT8 TIME;
INT8 A;
TIME=50;
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
/
Q=0;
while(Q<1){
A=3;
PUTC(A);
DELAY_MS(4000);

Q=Q+1;
}
Q=0;
```

```
WHILE(Q<5){
   pulsoss(1,85);  // hombro_1
   pulsoss(2,15);  // codo_1
   pulsoss(3,170);   // pata_1
     pulsoss(5,90); // hombro_2
   pulsoss(6,160); // codo_2
   pulsoss(7,40);// pata_2
    pulsoss(9,94); // hombro_3
   pulsoss(10,20);// codo_3
   pulsoss(11,125);// pata_3
      pulsoss(13,85); // hombro_4
   pulsoss(14,160); // codo_4
   pulsoss(15,50); // pata_4
   pulsoss(17,80); // hombro_5
   pulsoss(18,10); // codo_5
   pulsoss(19,125); // pata_5
     pulsoss(21,90); // hombro_6
   pulsoss(22,160); // codo_6
   pulsoss(23,20); // pata_6
    Q=Q+1;
}
Q=0;
WHILE(Q<5){
   pulsoss(1,85);  // hombro_1
   pulsoss(2,15);  // codo_1
   pulsoss(3,150);   // pata_1
```

```
  pulsoss(5,90); // hombro_2
  pulsoss(6,160); // codo_2
  pulsoss(7,60);// pata_2
   pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,105);// pata_3
  pulsoss(13,85); // hombro_4
  pulsoss(14,160); // codo_4
  pulsoss(15,70); // pata_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,10); // codo_5
  pulsoss(19,105); // pata_5
    pulsoss(21,90); // hombro_6
  pulsoss(22,160); // codo_6
  pulsoss(23,40); // pata_6
 Q=Q+1;
}
Q=0;
WHILE(Q<5){
  pulsoss(1,85);  // hombro_1
  pulsoss(2,15);  // codo_1
  pulsoss(3,140);   // pata_1
    pulsoss(5,90); // hombro_2
  pulsoss(6,160); // codo_2
  pulsoss(7,70);// pata_2
```

```
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,95);// pata_3
     pulsoss(13,85); // hombro_4
  pulsoss(14,160); // codo_4
  pulsoss(15,80); // pata_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,10); // codo_5
  pulsoss(19,95); // pata_5
    pulsoss(21,90); // hombro_6
  pulsoss(22,160); // codo_6
  pulsoss(23,50); // pata_6
 Q=Q+1;
}
Q=0;
WHILE(Q<5){
  pulsoss(1,85);  // hombro_1
  pulsoss(2,15);  // codo_1
  pulsoss(3,130);   // pata_1
  pulsoss(5,90); // hombro_2
  pulsoss(6,160); // codo_2
  pulsoss(7,80);// pata_2
  pulsoss(9,94); // hombro_3
  pulsoss(10,20);// codo_3
  pulsoss(11,85);// pata_3
  pulsoss(13,85); // hombro_4
```

```
  pulsoss(14,160); // codo_4
  pulsoss(15,90); // pata_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,10); // codo_5
  pulsoss(19,85); // pata_5
    pulsoss(21,90); // hombro_6
  pulsoss(22,160); // codo_6
  pulsoss(23,60); // pata_6
   Q=Q+1;
}
Q=0;
while(Q<60){
  pulsoss(1,85);  // hombro_1
  pulsoss(2,90);  // codo_1
  pulsoss(3,180);   // pata_1
    pulsoss(5,90); // hombro_2
  pulsoss(6,90); // codo_2
  pulsoss(7,30);// pata_2
   pulsoss(9,94); // hombro_3
  pulsoss(10,90);// codo_3
  pulsoss(11,145);// pata_3
  pulsoss(13,85); // hombro_4
  pulsoss(14,90); // codo_4
  pulsoss(15,40); // pata_4
  pulsoss(17,80); // hombro_5
  pulsoss(18,90); // codo_5
```

```
  pulsoss(19,135); // pata_5
    pulsoss(21,90); // hombro_6
  pulsoss(22,90); // codo_6
  pulsoss(23,10); // pata_6
Q+=1;
}
Q=0;
while(Q<TIME){
 pulsoss(1,63);  // hombro_1
  pulsoss(3,0);   // pata_1
  pulsoss(2,90);  // codo_1
    pulsoss(5,107); // hombro_2
  pulsoss(7,180);// pata_2
  pulsoss(6,90); // codo_2
  pulsoss(9,94); // hombro_3
  pulsoss(10,90);// codo_3
  pulsoss(11,0);// pata_3
  pulsoss(13,85); // hombro_4
  pulsoss(14,90); // codo_4 20
  pulsoss(15,170); // pata_4
    pulsoss(17,95); // hombro_5
  pulsoss(19,0); // pata_5
  pulsoss(18,90); // codo_5
  pulsoss(21,70); // hombro_6
  pulsoss(23,180); // pata_6
  pulsoss(22,90); // codo_6
```

```
  Q+=1;
}
Q=0;
while(Q<TIME){
 pulsoss(1,63);  // hombro_1
  pulsoss(3,0);   // pata_1
  pulsoss(2,120);  // codo_1
   pulsoss(5,107); // hombro_2
   pulsoss(7,180);// pata_2
   pulsoss(6,45); // codo_2
    pulsoss(9,94); // hombro_3
   pulsoss(10,140);// codo_3
   pulsoss(11,0);// pata_3
   pulsoss(13,85); // hombro_4
   pulsoss(14,40); // codo_4 20
   pulsoss(15,170); // pata_4
   pulsoss(17,95); // hombro_5
   pulsoss(19,0); // pata_5
   pulsoss(18,135); // codo_5
   pulsoss(21,70); // hombro_6
   pulsoss(23,180); // pata_6
   pulsoss(22,60); // codo_6
Q+=1;
}
 BREAK;
```

}

SUBRUTINAS DE PROGRAMACIÓN DEL ROBOT TRANSFORMER HUMANOIDE - AUTO

- **Rutina para control de servos**

```
#define PIE_DER PIN_B0
 #define TOBI_DER PIN_B1
 #define RODI_DER PIN_B2
 #define CAD_DER PIN_B3
 #define PIE_IZQ PIN_D0
 #define TOBI_IZQ PIN_D1
 #define RODI_IZQ PIN_D2
 #define CAD_IZQ PIN_D3

 #define HOMB_DER PIN_B4
 #define CODO_DER PIN_B5
 #define MANO_DER PIN_B6
 #define HOMB_IZQ PIN_D4
 #define CODO_IZQ PIN_D5
 #define MANO_IZQ PIN_D6

 #define COELLO PIN_C0
 #define CABEZA PIN_C1
 #define GIRA_DER PIN_C2
 #define GIRA_IZQ PIN_C3
 #define RUEDA_SUP_DER PIN_C4
```

```
#define RUEDA_SUP_IZQ PIN_C5
#define RUEDA_INF_DER PIN_C6
#define RUEDA_INF_IZQ PIN_C7


#define ENABLE PIN_A2
#define IN_A PIN_A0
#define IN_B PIN_A1

#define led_blanco   pin_e2
#define led_amarillo pin_b7


void angulo (int16 servo,angulo){
int16 pulso=0;
int16 bajo=0;
int8 serv;//int16

if (servo==1) serv = pie_der;
if (servo==2) serv = tobi_der;
if (servo==3) serv = rodi_der;
if (servo==4) serv = cad_der;

if (servo==5) serv = pie_izq;
if (servo==6) serv = tobi_izq;
if (servo==7) serv = rodi_izq;
```

```
if (servo==8) serv = cad_izq;

if (servo==9) serv = codo_der;
if (servo==10) serv = codo_izq;

if (servo==11) serv = homb_der;
if (servo==12) serv = homb_izq;

if (servo==13) serv = gira_der;
if (servo==14) serv = gira_izq;

if (servo==15) serv = rueda_sup_der;
if (servo==16) serv = rueda_sup_izq;

if (servo==17) serv = rueda_inf_der;
if (servo==18) serv = rueda_inf_izq;

output_low(serv);
pulso=((11.11*angulo)+500.2)-20; //angulo inicial en valores eje 1500,el -20 no
bajo=20-(pulso/1000);            //es parte de la formula,solo para compensar

    output_high(serv);
    delay_us(pulso);
    output_low(serv);
    delay_ms(bajo);
    break;
```

```
}


void angulo_2 (int16 servo,angulo){
int16 pulso;
int16 serv;
int8 x;
x=0;
if (servo==1) serv = coello;
if (servo==2) serv = cabeza;
if (servo==3) serv = gira_der;
if (servo==4) serv = gira_izq;
if (servo==5) serv = homb_der;
if (servo==6) serv = homb_izq;
if (servo==7) serv = mano_der;
if (servo==8) serv = mano_izq;

output_low(serv);
pulso=((11.11*angulo)+500.2)-20; //angulo inicial en valores eje 1500,el -20 no
                                 //es parte de la formula,solo para compensar
while(x<30){
    output_high(serv);
    delay_us(pulso);
    output_low(serv);
    delay_ms(18);
    x+=1;
```

```
    }
    break;
}
```

- **Rutina para posición inicial (modo humano)**

```
void inicial(){

output_low(homb_der);
output_low(codo_der);
output_low(mano_der);
output_low(homb_izq);
output_low(codo_izq);
output_low(mano_izq);

  output_low(coello);
  output_low(cabeza);
  output_low(gira_der);
  output_low(gira_izq);
  output_low(homb_der);
  output_low(homb_izq);
  output_low(mano_der);
  output_low(mano_izq);

  delay_ms(1000);
```

```
////////////////////////////////////////////////////////////////////////////

angulo(1,90); //pie der

angulo(2,30); //tobi der

angulo(3,90); //rodi der

angulo(4,70); //60cad der

angulo(5,90); //pie izq

angulo(6,150); //tobi izq

angulo(7,90); //rodi izq

angulo(8,110); //120cad izq

angulo(11,85); //*110homb_der 90

angulo(12,95); //*70homb_izq 90

angulo(9,10); //codo der

angulo(10,170); //codo izq

angulo_2(1,90); //coello

angulo_2(2,90); //cabeza

angulo(13,30); //gira_der

angulo(14,150); //gira_izq
```

```
angulo_2(7,90); //mano_der
angulo_2(8,90); //mano_izq



delay_ms(1000);

}
```

- **Subrutina para transformación de modo Auto a Humano**

```
void humano (){
////////////////////////////////////////////////////////////////////////////
//////////////habla modo humano////////////////////////////////////////////////

output_high(enable);
delay_ms(500);

output_low(in_a);
output_low(in_b);
output_low(enable);
delay_ms(3500);

output_high(enable);
delay_ms(100);
```

```
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

angulo(11,180);//homb der 90
angulo(12,0);//homb izq 90
delay_ms(500);

angulo(9,180); //codo der 0
angulo(10,0); //codo izq 180
delay_ms(1000);


angulo(13,30);//gira der *110
angulo(14,150);//gira izq *10
delay_ms(500);


//normalmente van las caderas aqui
////////////////////////////////////////////////////////////////////////////

angulo(2,10); //*0tobi der
angulo(6,170); //*180tobi izq
delay_ms(500);

angulo(3,120); //rodi der 40
angulo(7,60); //rodi izq 140
```

```
delay_ms(500);

angulo(4,35);//cad der 95
angulo(8,145);//cad izq 75
delay_ms(500);

angulo(11,145);//homb der
angulo(12,30);//homb izq
delay_ms(500);


angulo(3,105); //*120rodi der
angulo(7,75); //*60rodi izq
angulo(4,20);//cad der
angulo(8,160);//cad izq


delay_ms(500);


angulo(3,45); //*45-105rodi der
angulo(7,135); //*135-75rodi izq
delay_ms(500);

//hasta aqui el robot esta inclinado falta un impulso que le voy a dar con
//homros o mano
```

```
angulo(11,130);//*145homb der
angulo(12,45);//*30homb izq
delay_ms(500);

angulo(3,30); //*rodi der
angulo(7,150); //*rodi izq
delay_ms(500);
}


void humano_2 (){
angulo(11,85); //*110homb_der 90
angulo(12,95); //*70homb_izq 90

angulo(9,10); //codo der
angulo(10,170); //codo izq
delay_ms(1000);

////////////////////doble de tobillos/////////////////////////////////////

//tobillos de 10 a 30 der y de 170 a 150 izq
//rodillas de 30 a 90 der y de 150 a 90
//pos inicial tobi der 30, izq 150, rodi der 90, 90 , cad der 70,110
//angulo(4,20);//cad der
//angulo(8,160);//cad izq
```

```
angulo(3,40); //rodi der
angulo(7,140); //rodi izq
delay_ms(100);
angulo(2,20); //tobi der
angulo(6,160); //tobi izq
delay_ms(100);

angulo(3,50); //rodi der
angulo(7,130); //rodi izq
delay_ms(100);
angulo(2,30); //tobi der
angulo(6,150); //tobi izq
delay_ms(100);


angulo(3,60); //rodi der
angulo(7,120); //rodi izq
delay_ms(100);
angulo(2,40); //tobi der
angulo(6,140); //tobi izq
delay_ms(100);


angulo(3,70); //rodi der
```

```
angulo(7,110); //rodi izq
delay_ms(100);
angulo(2,50); //tobi der
angulo(6,130); //tobi izq
delay_ms(100);

angulo(3,80); //rodi der
angulo(7,100); //rodi izq
delay_ms(100);
angulo(4,30); //60cad der
angulo(8,150); //120cad izq
delay_ms(100);
//angulo(2,40); //tobi der
//angulo(6,140); //tobi izq
//delay_ms(500);

angulo(3,90); //rodi der
angulo(7,90); //rodi izq
delay_ms(100);
angulo(4,40); //60cad der
angulo(8,140); //120cad izq
delay_ms(100);

}
```

- **Subrutina para transformación de modo Humano a Auto**

```
void carro() {
int8 n;
int16 pulso_1;
int16 pulso_2;
int8 valor; //der
int8 valor_2; //izq
int8 numero;
//int8 z;

angulo(13,170);//*110 gira_Der
angulo(14,10);// gira_izq
angulo(9,120); //codo der *90
angulo(10,60); //codo izq *90
angulo(11,90); //homb der
angulo(12,90); //homb izq

delay_ms(3000);

////////////////////////////////////////////////////////////////////////////
//////////////////////doble de rodillas/////////////////////////////////////
n=1;
pulso_1=0;
pulso_2=0;
```

```
output_low(rodi_der);
output_low(rodi_izq);

//rodillas de 90 a 60 der y de 90 a 120 izq
valor=90; //der
valor_2=90; //izq

  while (valor >= 60){  //
   pulso_1=(11.11*valor)+500.2; //angulo inicial en valores eje 1500
   pulso_2=(11.11*valor_2)+500.2; //angulo inicial en valores eje 1500

     output_high(rodi_der);
     delay_us(pulso_1);
     output_low(rodi_der);

     output_high(rodi_izq);
     delay_us(pulso_2);
     output_low(rodi_izq);

     n+=0.5;
     valor=valor-n;
     valor_2=valor_2+n;

     delay_ms(20);
  }
```

```
valor=0; //der
valor_2=0; //izq
n=1;
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
delay_ms(1000);
////////////////////////////////////////////////////////////////////////////
//////////////////////doble de tobillos/////////////////////////////////////
n=1;
pulso_1=0;
pulso_2=0;

output_low(tobi_der);
output_low(tobi_izq);

//tobillos de 30 a 0 der y de 150 a 180 izq
valor=30; //der
valor_2=150; //izq

numero=60; //se resta hasta 30 normalmente

   while (numero >= 35){ //uso numero para restar 30 (60-30)y ni llegar a 0
   pulso_1=(11.11*valor)+500.2; //angulo inicial en valores eje 1500
   pulso_2=(11.11*valor_2)+500.2; //angulo inicial en valores eje 1500

     output_high(tobi_der);
```

```
      delay_us(pulso_1);
      output_low(tobi_der);

      output_high(tobi_izq);
      delay_us(pulso_2);
      output_low(tobi_izq);

      n+=0.5;
      valor=valor-n; //tobi der
      valor_2=valor_2+n; //tobi izq

      numero=numero-n;

      delay_ms(20);
      }

n=1;
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

}

void carro_22(){
int8 n=1;
int16 pulso_1=0;
int16 pulso_2=0;
```

```
int8 valor= 0; //der
int8 valor_2=0; //izq
//int8 numero;
//int8 z=0;
////////////////////////////////////////////////////////////////////////////
////////////////////doble de rodillas hacia adelante//////////////////////////
pulso_1=0;
pulso_2=0;

output_low(rodi_der);
output_low(rodi_izq);

//rodillas de 60 a 120 der y de 120 a 60 izq
valor=60; //der
valor_2=120; //izq
  while (valor <= 120){   //

   pulso_1=(11.11*valor)+500.2; //angulo inicial en valores eje 1500
   pulso_2=(11.11*valor_2)+500.2; //angulo inicial en valores eje 1500

     output_high(rodi_der);
     delay_us(pulso_1);
     output_low(rodi_der);

     output_high(rodi_izq);
     delay_us(pulso_2);
```

```
      output_low(rodi_izq);

      n+=0.5;
      valor=valor+n;
      valor_2=valor_2-n;

      delay_ms(20);
      }

n=1;
////////////////////////////////////////////////////////////////////////////
//////////////////////doble de codos para posicion//////////////////////////////
pulso_1=0;
pulso_2=0;

output_low(codo_der);
output_low(codo_izq);

//rodillas de 90 a 0 der y de 90 a 180 izq
valor=120; //der
valor_2=60; //izq

   while (valor_2 <= 180){  //

    pulso_1=(11.11*valor)+500.2; //angulo inicial en valores eje 1500
    pulso_2=(11.11*valor_2)+500.2; //angulo inicial en valores eje 1500
```

```
      output_high(codo_der);
      delay_us(pulso_1);
      output_low(codo_der);

      output_high(codo_izq);
      delay_us(pulso_2);
      output_low(codo_izq);


      n+=0.5;
      valor=valor-n;
      valor_2=valor_2+n;

      delay_ms(20);
      }

n=1;
delay_ms(500);
}
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
void carro_33(){
int8 z=0;

angulo(9,0); //codo der 0
```

```
angulo(10,175); //codo izq 180

angulo(3,40); //rodi der 120
angulo(7,140); //rodi izq 60
delay_ms(500);

angulo(4,90); //cad der
angulo(8,90); //cad izq
delay_ms(500);

while (z<10){

angulo(2,35); //*10 tobi der 0
angulo(6,145); //*160 tobi izq 180
delay_ms(500);

angulo(4,95); //cad der 60-90 *105
angulo(8,75); //cad izq 120-90

z+=1;
}



}

void  rodar (){
```

```
int8 h;

for(h=0;h<50;h++){
output_high (led_blanco);
output_low (led_amarillo);

output_high(rueda_sup_izq); //gira izq-hitec
output_high(rueda_sup_der); //gira_der
output_high(rueda_inf_der); //gira_der
delay_us(500);//der
output_low(rueda_inf_der);
output_low(rueda_sup_der);
output_low(rueda_sup_izq);

output_high(rueda_inf_izq); //gira_izq
delay_us(2500);// izq
output_low(rueda_inf_izq);

delay_ms(17);
}
h=0;
delay_ms(1000);


//atras
for(h=0;h<100;h++){
```

```
output_high (led_blanco);
output_low (led_amarillo);

output_high(rueda_sup_izq);
output_high(rueda_sup_der);
output_high(rueda_inf_der);
delay_us(2500);//izq
output_low(rueda_inf_der);
output_low(rueda_sup_der);
output_low(rueda_sup_izq);


output_high(rueda_inf_izq);
delay_us(500);// der
output_low(rueda_inf_izq);

delay_ms(17);
}
h=0;
delay_ms(1000);

//g_derecha
for(h=0;h<100;h++){
output_high (led_blanco);
output_high (led_amarillo);
```

```
output_high(rueda_sup_izq); //gira izq hitec (delante)
delay_us(500);//der
output_low(rueda_sup_izq);

output_high(rueda_inf_izq); //gira izq
output_high(rueda_sup_der); //gira_izq
output_high(rueda_inf_der); //gira_izq
delay_us(2500);// izq
output_low(rueda_inf_izq);
output_low(rueda_sup_der);
output_low(rueda_inf_der);



delay_ms(17);
}
h=0;
delay_ms(1000);

//g_izquierda
for(h=0;h<100;h++){
output_high (led_blanco);
output_high (led_amarillo);

output_high(rueda_inf_izq); //gira der
output_high(rueda_sup_der); //gira_der
output_high(rueda_inf_der); //gira_der
```

```
delay_us(500);//der
output_low(rueda_inf_izq);
output_low(rueda_sup_der);
output_low(rueda_inf_der);

output_high(rueda_sup_izq); //gira der hitec (delante)
delay_us(2500);//der
output_low(rueda_sup_izq);

delay_ms(17);
}
h=0;
delay_ms(1000);



}
```

- **Subrutina para lectura de sensor DS18S20**

```
// (C) copyright 2003 j.d.sandoz / jds-pic !at! losdos.dyndns.org

 // released under the GNU GENERAL PUBLIC LICENSE (GPL)
 // refer to http://www.gnu.org/licenses/gpl.txt

 // This program is free software; you can redistribute it and/or modify
 // it under the terms of the GNU General Public License as published by
 // the Free Software Foundation; either version 2 of the License, or
```

```
// (at your option) any later version.

// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.

// You should have received a copy of the GNU General Public License
// along with this program; if not, write to the Free Software
// Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

/**********************1Wire Class**********************/
/*Description: This class handles all communication */
/* between the processor and the 1wire */
/* sensors.
/*************************************************************/

/*-------1-wire definitions-------*/
#define ONE_WIRE_PIN PIN_a4

/********************1-wire communication functions********************/

/************onewire_reset*****************************************
*******/
/*This function initiates the 1wire bus */
```

```
 /* */
 /*PARAMETERS: */
 /*RETURNS: */

/*****************************************************************
****/

 void onewire_reset()  // OK if just using a single permanently connected device
 {
  output_low(ONE_WIRE_PIN);
  delay_us( 500 ); // pull 1-wire low for reset pulse
  output_float(ONE_WIRE_PIN); // float 1-wire high
  delay_us( 500 ); // wait-out remaining initialisation window.
  output_float(ONE_WIRE_PIN);
 }

 /************************                                    onewire_write()
*********************************/
 /*This function writes a byte to the sensor.*/
 /* */
 /*Parameters: byte - the byte to be written to the 1-wire */
 /*Returns: */

/*****************************************************************
****/

 void onewire_write(int data)
 {
```

```
 int count;

 for (count=0; count<8; ++count)
 {
  output_low(ONE_WIRE_PIN);
  delay_us( 2 ); // pull 1-wire low to initiate write time-slot.
  output_bit(ONE_WIRE_PIN, shift_right(&data,1,0)); // set output bit on 1-wire
  delay_us( 60 ); // wait until end of write slot.
  output_float(ONE_WIRE_PIN); // set 1-wire high again,
  delay_us( 2 ); // for more than 1us minimum.
 }
}

/************************                                    read1wire()
*********************************/
/*This function reads the 8 -bit data via the 1-wire sensor. */
/* */
/*Parameters: */
/*Returns: 8-bit (1-byte) data from sensor */

/*****************************************************************
****/

int onewire_read()
{
 int count, data;

 for (count=0; count<8; ++count)
```

```
 {
  output_low(ONE_WIRE_PIN);
  delay_us( 2 ); // pull 1-wire low to initiate read time-slot.
  output_float(ONE_WIRE_PIN); // now let 1-wire float high,
  delay_us( 8 ); // let device state stabilise,
  shift_right(&data,1,input(ONE_WIRE_PIN)); // and load result.
  delay_us( 120 ); // wait until end of read slot.
 }

 return( data );
}
```

- **Subrutina de lectura de dato de sensro DS18B20**

```
float ds1820_read()
 {
  int8 busy=0, temp1, temp2;
  signed int16 temp3;
  float result;

  onewire_reset();
  onewire_write(0xCC);
  onewire_write(0x44);

  while (busy == 0)
  busy = onewire_read();
```

```
 onewire_reset();
 onewire_write(0xCC);
 onewire_write(0xBE);
 temp1 = onewire_read();
 temp2 = onewire_read();
 temp3 = make16(temp2, temp1);

// result = (float) temp3 / 2.0;     //Calculation for DS18S20 with 0.5 deg C resolution
 result = (float) temp3 / 16.0;  //Calculation for DS18B20 with 0.1 deg C resolution
// result = ((float) temp3 / 16.0)-2.0;  //0.1 calib en real, no proteus

 delay_ms(200);
 return(result);
}
```

- **Subrutina para medir sensores**

```
#define SIG   pin_a3
#define led_blanco   pin_e2
#define led_amarillo pin_b7

float distancia(void) {          //funcion que mide la distancia por TIMER1

  long pulso;                 //variable de 16bits
  float distancia;            //variable de 32bits
  output_high(SIG);        //ponemos en alto el SIG
```

```
  delay_us(10);                //aplicando un pulso de 10uS
  output_low(SIG);         //ponemos en bajo el pin SIG
  set_tris_a(0x08);        //configuramos el pin SIG "a3" como entrada
  output_high(enable);      //para que no hable
  //debo considerar configuracion memoria de voz porq al configurar trisa
  //se pone en 0 los puertos de voz y habla el robot

  while(!input_state(SIG)){}//se queda preguntando por el inicio del eco
  set_timer1(0);               //4m inicio el temporizador TIMER1, lleva la cuenta del tiempo incrementando cada 1us
  while(input_state(SIG)){}  //pregunto si termina el eco
  pulso=get_timer1();          //cuando finaliza, obtengo el dato del TIMER1 que equivale al tiempo
  pulso=pulso/5;
  delay_ms(30);              //espero 30ms para una nueva medida
  distancia=(pulso/58.12);   // distacia en cm.
  return distancia;
}



void medicion(){
 float temperature;
 float temperature_2;
 float dist_cm=0;
 int8 temp;
 int8 dist;
```

```
 setup_adc_ports(NO_ANALOGS);
 setup_adc(ADC_OFF);
 setup_psp(PSP_DISABLED);
 setup_spi(FALSE);
 setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
 setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);//configuro el timer1
 setup_timer_2(T2_DISABLED,0,1);
 setup_comparator(NC_NC_NC_NC);
 setup_vref(VREF_LOW|-2);

   set_tris_b(0x00);
   output_b(0);

delay_ms(2000);

while(1){
   dist_cm=distancia();
   temperature = ds1820_read();//la primera lectura sabe fallar
   delay_ms(100);
   temperature_2=ds1820_read();
   delay_ms(100);

 temp=(int)temperature; //convierte de flotante a entero
 temp=(int)temperature_2; //convierte de flotante a entero
 dist=(int)dist_cm;
```

```
//envio valores

delay_ms(1000);
output_high(led_amarillo);
delay_ms(500);
output_low(led_amarillo);
delay_ms(500);

output_high(led_blanco);
delay_ms(500);
output_low(led_blanco);
delay_ms(500);

putc(temp);
delay_ms(2000);

output_high(led_blanco);
delay_ms(500);
output_low(led_blanco);
delay_ms(500);

putc(dist);
delay_ms(500);
break;
```

```
}
  }
```

- **Subrutina para caminar en modo Humano**

```
void camina_2(){

int8 pasos;
int8 y;
int8 n;
int8 paso;
int16 espera;
int16 espera_2;

pasos=0;
y=0;
n=0;
paso=3; //numero de pasos que el robot va a dar
espera=100;
espera_2=100;

//habla caminar//////////////////////////////////////////////////////////////
output_high(enable);
delay_ms(500);

output_high(in_a);
output_low(in_b);
```

```
output_low(enable);
delay_ms(3500);

output_high(enable);
delay_ms(100);
////////////////////////////////////////////////////////////////////////////
//////////////////////****posicion inicial****////////////////////////////////
////////***se modifico posicion inicial de cad der-izq y pie izq***/////////////
delay_ms(500);

while(pasos < paso){ //numero de pasos

output_low(homb_der);
output_low(codo_der);
output_low(mano_der);
output_low(homb_izq);
output_low(codo_izq);
output_low(mano_izq);


angulo(1,90); //pie der
angulo(2,30); //tobi der
angulo(3,90); //rodi der
angulo(4,70); //60cad der

angulo(5,90); //pie izq
```

```
angulo(6,150); //tobi izq
angulo(7,90); //rodi izq
angulo(8,110); //120cad izq

angulo(9,10); //codo der
angulo(10,170); //codo izq

delay_ms(100);//stop//////////////////////////

////////////////////////////////////////////////////////////////////////////
/////////////////primer paso/////////////////////////////////////////////////

//levanto pie derecho

angulo(5,105);//pie izq inclino
angulo(1,105);//pie der recto
delay_ms(500);

angulo(4,15);//60-30cad der levanto pie der
angulo(2,60);//30-50tobi der levanto pie de
angulo(3,70);//rodi_der nuevo
delay_ms(espera);

angulo(3,90);//rodi_der nuevo
delay_ms(espera);
```

```
//pongo recto los pies base
angulo(1,90);//pie der  inclinado, recto 105
angulo(5,90);//pie izq recto
delay_ms(espera);

//acomodo pie derecho
angulo(2,45); //tobi der
angulo(4,35); //cad der
delay_ms(espera);//corregir

////////////////////////////////////////////////////////////////////////////

//pongo recto pie izq y levanto pie derecho
angulo(1,75);//pie der inclino mas
angulo(4,70);//60cad der/////////////////////////////////////
angulo(3,110);//rodi der
delay_ms(espera);

////////////////////////////////////////////////////////////////////////////
//////////////////////perfecto/////////////////////////////////////////////////
angulo(1,90);//pie der inclinado inicial
//delay_ms(espera);//150

//acomoda en "posicion inicial"
angulo(7,90); //rodi izq
angulo(6,150); //*150tobi izq
```

```
angulo(8,110);//120cad izq
delay_ms(espera);//150

angulo(3,90); //rodi der
angulo(2,30); //*30tobi der
angulo(4,70);//60cad der
delay_ms(1200);

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

//levanto pie izquierdo
angulo(5,80);//90*75pie izq recto
angulo(1,75);//*75pie der inclino
delay_ms(500);


angulo(8,165);//*165
angulo(6,120);//*120
angulo(7,110);//*110
delay_ms(espera_2);//150

angulo(7,90);//rodi_der nuevo
delay_ms(espera_2);//150

////pongo recto los pies base
```

```
angulo(5,95);//105*90pie izq mas inclinado-75
angulo(1,105);//*90pie der recto
delay_ms(espera_2);//150

///acomodo pie derecho
angulo(6,135); //*135tobi der
angulo(8,145); //*145cad der
delay_ms(espera_2);//150



////////////////////////////////////////////////////////////////////////////

//pongo recto pie izq y levanto pie derecho
angulo(5,90);//*105pie izq
angulo(8,110);//*110cad izq/////////////////////////////////////
angulo(7,70);//*70rodi der
delay_ms(espera_2);//150

////////////////////////////////////////////////////////////////////////////
//////////////////////perfecto/////////////////////////////////////////////////
angulo(5,90);//pie izq
delay_ms(espera_2);//150

//acomoda en "posicion inicial"
angulo(3,90); //*90rodi izq
```

```
angulo(2,30); //*30tobi izq
angulo(4,70);//*70cad izq
angulo(1,90);//*90pie der recto
delay_ms(espera_2);//150

angulo(7,90); //*90rodi der
angulo(6,150); //*150tobi der
angulo(8,110);//*110cad der
delay_ms(1200);
//delay_ms(espera);

pasos+=1;
  }
}
```