



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**DISEÑO Y CONSTRUCCIÓN DE UN
MINIROBOT CON SISTEMA DE VISIÓN**

T E S I S

PARA OBTENER EL TITULO DE:

**INGENIERO EN COMPUTACIÓN
INGENIERO ELÉCTRICO ELECTRÓNICO**

PRESENTAN:

**ALEJANDRA CECILIA SÁNCHEZ BERMEJO
FRANCISCO DORANTES ÁNGELES**

DIRECTOR DE TESIS:

DR. JESÚS SAVAGE CARMONA



Ciudad Universitaria, México D.F. Mayo 2010

AGRADECIMIENTOS

A MIS PAPÁS:

JUAN SÁNCHEZ RUÍZ Y GRACIELA BERMEJO HERNÁNDEZ

Agradezco a la vida por haberlos tenido de padres y que se nos haya dado la oportunidad de ser una familia.

Les agradezco por todo lo que me han dado en la vida, y gracias a ustedes puedo presentar esta tesis que es para ustedes, gracias por todo su apoyo por estar en cada proyecto que he estado, desde que empecé a estudiar, por su paciencia, por estar conmigo de arriba para abajo en lo que he necesitado, por sus consejos, por compartir sus experiencias que son grandes enseñanzas, por desvelarse al parejo de mi cuando lo necesite, por aguantarnos cuando se hacían trabajos de equipo en la casa, por escuchar, por defenderme de todo y contra todos, por aceptarme así como soy, y sobre todo por creer en mí y darme el amor que necesito para poder seguir adelante.

Mami me has dado una gran lección de fortaleza por todo lo que has pasado, espero algún día ser tan fuerte como tú y poner ese esfuerzo en todo lo que haga, eres un gran ejemplo de vida y por ser la amiga que esta las 24 horas del día conmigo.

Papi te agradezco todo lo que has hecho por nosotras, por todo lo que nos has dado que afortunadamente nunca nos ha faltado nada, y por ser el fuerte en las pruebas que nos han puesto la vida, y también por ser el amigo de 24 horas.

Espero les guste este trabajo el cual les dedico y espero que sea una parte del futuro que nos espera juntos y así poder gratificarles todo lo que han hecho por mí y así seguir creciendo junto a ustedes.

Los quiero y los amo

Alejandra Cecilia Sánchez Bermejo

A Francisco Dorantes Ángeles:

Te agradezco el apoyo incondicional desde que nos conocimos, en toda la carrera, sin el cual no sé si estaría presentando este proyecto, gracias por estar ahí echándole ganas junto conmigo en las materias, así como las desveladas que nos pusimos para sacar los proyectos, por tenerme mucha paciencia y enseñarme lo que no entendía por lo que fuiste mi gran maestro, eres una persona que admiro mucho, por todo lo que haces y eres, te agradezco el haberme dado la oportunidad de estar a tu lado en todos los proyectos que hemos emprendido.

A mis maestros y sinodales:

Dr. Jesús Savage y sinodales gracias por compartir sus conocimientos y experiencias, y por todo el apoyo recibido durante los años que estuve en la Facultad, y por ayudarme a culminar esta etapa de mi vida al estar pendientes de esta tesis, gracias por el apoyo que hemos recibido en las competencias en las que hemos participado.

Julio y Edith:

Gracias por estar conmigo desde que nos conocimos, en verdad apreció lo que han hecho por mí al darme consejos, resolverme dudas en algunas de mis materias, y dedicarme tiempo, gracias sobre todo por estar pendientes de mis papás, el hecho de saber que están ahí es un gran apoyo, ahora sé que mis hermanos llegaron cuando los conocí, espero que así como se que cuento con ustedes sepan que siempre podrán contar conmigo al igual que sus niñas.

A mi familia:

A las dos familias Sánchez y Bermejo, gracias por el apoyo recibido en toda mi vida, por compartir las alegrías y tristezas conmigo, por vivir las tensiones de las competencias y por todos sus consejos gracias. Denisse, Carlitos, Ángel y Alex, solo quiero que sepan, que al igual que yo tuve el apoyo de muchas personas, espero que sepan que pueden contar conmigo para lo que necesiten, aunque no lo crean me han dado mucho apoyo, al estar conmigo los fines de semana porque a veces cuando más pesadas se vuelven las cosas, una sonrisa de ustedes, bajaba el stress que traía, y me inyectaban energía, y espero que sigan echándole ganas a la escuela, gracias por ser la alegría de la casa, los quiero mucho

A mis amigos:

Ángeles, Ana, Carlos, Gaby, Janet, Beto, Misael, Jonathan, Gracias por estar ahí porque compartimos momentos buenos y malos durante nuestra formación, gracias por ser compañeros de equipo, de estudio, de diversión y por compartir momentos de su vida conmigo ya que todas esas experiencias me han ayudado en mi vida diaria.

AGRADECIMIENTOS

Ante todo, quiero agradecer a todas aquellas personas que han hecho posible la culminación de mis estudios y la elaboración de esta tesis.

En este apartado, quiero agradecer a las instituciones y a los profesores que han forjado mi camino con su apoyo y dedicación constante.

Un apoyo fundamental, ha sido el doctor Jesús Savage, quien dirigió esta tesis, así como los miembros del jurado que la evaluaron. Gracias por sus comentarios y consejos, sus aportaciones fueron de gran ayuda.

A mis amigos, con los cuales he disfrutado todo este camino y me han ayudado a crecer y madurar como persona.

A mis hermanos Irma, Guillermina, Alicia y Oscar por haber crecido a mi lado y ser un apoyo constante en cada uno de mis días

En lo personal, agradezco especialmente el apoyo obtenido de mi novia Alejandra, por su motivación e implicación en los retos que a nivel personal, me propongo y que día a día me hace querer ser mejor, muchas gracias por ser mi compañera en todo momento.

Agradezco enormemente el apoyo incansable de mis padres, Francisco y Juana, a quien dedico esta tesis, que con su cariño, su confianza, sus regaños, su amor han logrado hacer de mi el hombre que ahora soy, nunca podre pagarles todo lo que han hecho por mí, este es un logro más del que les estaré siempre agradecido.

PORTADA	
AGRADECIMIENTOS	
ÍNDICE DE CONTENIDO	
LISTA DE FIGURAS	
LISTA DE TABLAS	
RESUMEN	

Capítulo 1 Introducción

1.1	Robótica -----	1
1.2	Robot -----	1
1.3	Breve historia y evolución de los robots -----	1
1.4	Línea del tiempo -----	2
1.5	Robots móviles -----	4
1.6	Robots autónomos -----	5
1.7	Clasificación de los tipos de robots -----	5
1.7.1	Clasificación por Generación -----	5
1.7.2	Clasificación por su Arquitectura -----	6
1.7.3	Clasificación por su nivel de inteligencia -----	7
1.7.4	Clasificación por su nivel de control -----	7
1.7.5	Clasificación por lenguaje de programación -----	7
	8Planteamiento del problema	
2.1	Robots desactivadores de bombas -----	9
2.1.1	Robot Asendro Eod -----	9
2.1.2	Robot Packbot 510 -----	10
2.1.3	Robot Wolverine -----	10
2.1.4	Robot Israelita -----	11
2.1.5	Robot de Tlaxcala -----	11
2.2	Proyecto Robot que simula la desactivación de bombas -----	11
2.3	Descripción de la categoría open -----	12
2.4	Alcance del proyecto -----	15
2.5	Estrategia que siguió para la desactivación de bombas -----	15

Capítulo 3 Marco teórico

3.1.	Microprocesadores -----	19
3.2.	Microcontroladores -----	19
3.2.1.	Arquitectura interna del microcontrolador -----	20
3.2.2.	Estructura básica de un microcontrolador -----	21
3.2.3.	Familias de microcontroladores -----	22
3.3.	Diferencia entre microcontrolador y microprocesador -----	23
3.4.	Microcontroladores PIC -----	24
3.4.1.	Reseña histórica de los PIC -----	24
3.4.2.	Gama de los microcontroladores PIC -----	24
3.5.	Microcontrolador PIC18f452 -----	25
3.5.1.	Definición de los pines y sus funciones -----	26

3.5.2.	Puertos digitales de entrada y salida -----	29
3.5.3.	Interrupciones del sistema -----	29
3.5.4.	Puerto de comunicación USART -----	29
3.5.4.1.	Transferencia síncrona -----	29
3.5.4.2.	Transferencia asíncrona -----	30
3.5.4.3.	Modulo SCI en transmisión asíncrona -----	30
3.5.5.	Módulo CCP (PWM/CAPTURA/COMPARACION) -----	31
3.5.6.	Timers -----	31
3.5.7.	Módulo Convertidor analógico/digital -----	31
3.6.	Comunicación serie -----	32
3.6.1.	Protocolo transmisión RS232 -----	33
3.6.1.1.	Puerto serie en el PC -----	35
3.6.2.	Circuito MAX232 -----	36
3.7.	Teoría de motores -----	36
3.7.1.	Motores de corriente directa o continua -----	37
3.7.1.1.	Principio de funcionamiento -----	39
3.7.2.	Puente H -----	40
3.7.2.1.	Circuito integrado L298 -----	42
3.7.2.2.	Sistema de protección -----	42
3.7.2.2.1.	Diodo rectificador como elemento de protección -----	43
3.7.3.	Servomotores -----	45
3.7.4.	Teoría PWM -----	45
3.7.4.1.	Ciclo de trabajo -----	45
3.8.	Sistemas mecánicos -----	45
3.8.1.	Tipos de plataformas móviles -----	45
3.8.1.1.	Configuración diferencial -----	46
3.8.1.2.	Configuración en triciclo -----	47
3.8.1.3.	Configuración Ackerman -----	47
3.8.1.4.	Configuración de dirección sincronizada -----	48
3.8.1.5.	Configuración omnidireccional -----	48
3.8.1.6.	Configuración mediante orugas -----	49
3.8.2.	Brazos mecánicos -----	49
3.8.2.1.	Grados de libertad -----	49
3.8.2.1.1.	Determinación de los G.de libertad --	49
3.8.3.	Gripper -----	50
3.9.	Sistemas de visión artificial -----	51
3.9.1.	Teoría de colores -----	52
3.9.1.1.	Sistema de visión RGB -----	53
3.9.2.	Cámara CMUCAM V3 -----	54
3.10.	Sensores como elementos de señalización -----	56
3.11.	Sistemas embebidos -----	56
3.12.	Máquina de estados o máquinas secuenciales -----	57
3.13.	Fuentes de energía (pilas) -----	58

Capítulo 4 Desarrollo

4.1.	Construcción del Hardware -----	59
4.1.1.	Diseño del sistema embebido -----	59
4.1.2.	Diseño del driver para los motores (Puente H) -----	62
4.2.	Construcción del robot -----	63
4.2.1.	Construcción de la base móvil -----	65

4.2.2.	Implementación de la tarjetas en el cuerpo -----	68
4.2.2.1.	Conexiones eléctricas de la base móvil -----	69
4.2.3.	Construcción y Montaje del mecanismo de la cámara y gripper -----	70
4.2.3.1.	Conexiones eléctricas de la cámara y gripper --	72
4.2.4.	Construcción y Montaje del brazo mecánico -----	73
4.2.4.1.	Conexiones eléctricas del brazo mecánico -----	74
4.3.	Estructura terminada -----	74
4.4.	Implementación del software -----	76
4.4.1.	Movimiento del robot -----	76
4.4.1.1.	Diagrama de flujo -----	77
4.4.1.2.	Explicación movimiento del robot -----	79
4.4.1.3.	Explicación Librería creada para el robot -----	81
4.4.2.	Software del sistema de visión -----	83
4.4.2.1.	Diagrama de flujo -----	85
4.4.2.2.	Explicación del sistema de visión -----	86
Capítulo 5 Pruebas y resultados		
5.1.	Prueba de movimiento de la base móvil -----	87
5.2.	Prueba de reconocimiento de colores -----	88
5.3.	Prueba de movimiento del robot y del sistema de visión -----	89
5.3.1.	Prueba del movimiento del robot con visión simulada -----	89
5.3.2.	Prueba del sistema de visión con movimiento simulado -----	90
5.3.3.	Prueba de consumo de energía -----	90
5.4.	Resultados -----	91
Capítulo 6 Conclusiones y trabajos futuros		
6.1.	Conclusiones -----	93
6.2.	Trabajos futuros -----	94
APENDICE A		
A.	Código del movimiento del robot -----	95
APENDICE B		
B.	Librería para el robot -----	103
APENDICE C		
C.	Código del sistema de visión -----	109
APENDICE D		
D.	Código prueba de movimiento -----	112

APENDICE E

E. Eagle	
i. Instalación del software para Windows -----	117
ii. Creación de un proyecto -----	117
iii. Desarrollo del diagrama esquemático -----	118
iv. Creación de la placa (board) -----	119

APENDICE F

F. MPLAB Y COMPILADOR CCS PARA WINDOWS	
i. Instalación del MPLAB -----	120
ii. Instalación del CCS y PLUG IN para MPLAB -----	121
iii. Creación de un proyecto -----	124
iv. Compilación, errores y depuración de un programa -----	129

APENDICE G

G. Software para el uso de la cámara CMUCAM V3	
i. Instalación del CYGWIN -----	132
ii. Instalación del GNU ARM GCC -----	135
iii. Instalación del LPC210x FLASH UTILITY -----	135
iv. Instalación del CC3 -----	136
v. Instalación del programa DEV-C++ para edición de programas -----	137
vi. Creación de un programa para la cámara -----	138
vii. Compilación, errores y depuración de un programa -----	139
viii. Descargar programa a la cámara -----	140

BIBLIOGRAFÍA -----	142
--------------------	-----

ÍNDICE DE TABLAS Y FIGURAS

LISTA DE FIGURAS

1.1	Puerta automática -----	2
1.2	Pato mecánico -----	2
2.1	Aseondro EOD -----	9
2.2	Packbot 510 -----	10
2.3	Wolverine -----	10
2.4	R. Israelita -----	11
2.5	R. Tlaxcala -----	11
2.6	Medidas de la pista -----	12
2.7	Vista de arriba de la pista -----	13
2.8	Posiciones de la bomba -----	13
2.9	Medidas de las bombas -----	14
2.10	Estrategia en la pista -----	15
2.11	Estrategia 2(uso del manipulador) -----	16
2.12	Búsqueda de los cables -----	16
2.13	Búsqueda de la Bomba azul -----	17
3.1	Microcontrolador -----	19
3.2	A. Von Neumann -----	20
3.3	A. Harvard -----	20
3.4	Diferencia entre microcontrolador y microprocesador -----	23
3.5	PIC 18F452 -----	26
3.6	Transferencia síncrona -----	30
3.7	Transferencia asíncrona -----	30
3.8	Ejemplo de comunicación PIC-PC mediante puerto serie (RS232) -----	30
3.9	Diagrama de bloques -----	32
3.10	Protocolo RS232 -----	34
3.11	Conectores DB25 Y DB9 -----	35
3.12	Circuito MAX 232 -----	36
3.13	Motor de corriente directa -----	37
3.14	Partes de un motor de corriente directa -----	38
3.15	Esquema interno de un motor de corriente continúa -----	38
3.16	Segunda ley de Lorentz 1 -----	39
3.17	Segunda ley de Lorentz 2 -----	39
3.18	Esquema puente H -----	41
3.19-A	Puente H -----	41
3.19-B	Puente H -----	41
3.20	Circuito integrado L298 -----	42
3.21	Esquemático L298 -----	42
3.22	Sistemas de protección en el puente H -----	43
3.23	Pulso y ángulo de un servomotor -----	44
3.24	Servomotor -----	44
3.25	Convertor de ancho de pulso -----	44
3.26	Ciclo de trabajo -----	45
3.27	Tracción errónea de un par diferencial -----	46
3.28	Tracción en un par diferencial -----	46
3.29	Configuración en triciclo -----	47
3.30	Configuración Ackerman -----	47
3.31	Configuración sincronizada -----	48
3.32	Configuración omnidireccional -----	48
3.33	Configuración mediante orugas -----	49
3.34	Gripper -----	50
3.35	Diagrama de visión artificial -----	51

3.36	Tipos de iluminación -----	51
3.37	Tipos de cámaras -----	52
3.38	Mezcla aditiva de colores -----	53
3.39	Cubo de color RGB -----	54
3.40	CMUCAM V3 -----	54
3.41	Sistema embebido de visión -----	55
3.42	Diagrama del chip de la cámara y el microprocesador -----	55
3.43	Diagrama del puerto para servomotores -----	56
3.44	Switch de contacto -----	56
3.45	Ejemplo de máquinas secuenciales y combinatorias -----	57
4.1	Diagrama eléctrico KITPIC -----	60
4.2	Vista por arriba KITPIC -----	60
4.3	Tarjeta KITPIC terminada -----	61
4.4	Descripción de la tarjeta de desarrollo -----	61
4.5	Esquemático del Puente H -----	62
4.6	Vista top Puente H -----	63
4.7	Perfiles de aluminio -----	64
4.8	Motor de la base móvil -----	64
4.9	Medidas físicas del motor -----	65
4.10	Robotshop-rover-arduino-tank-kit -----	65
4.11	Llantas -----	65
4.12	Estructura de la base móvil -----	66
4.13	Acoplamiento del motor con la llanta -----	66
4.14	Acoplamiento de las llantas con el eje -----	66
4.15	Acoplamiento de la oruga con las llantas -----	67
4.16	Implementación SW de contacto -----	67
4.17	Diagrama eléctrico de la conexión de las pilas -----	68
4.18	Implementación de las pilas en el robot -----	68
4.19	Implementación de las tarjetas de control en el robot -----	69
4.20	Diagrama eléctrico de la base móvil -----	69
4.21	Conexiones eléctricas de la etapa de potencia -----	70
4.22	Mecanismo interno de un reproductor de discos compactos -----	70
4.23	Adaptación del mecanismo CD y Gripper -----	71
4.24	Diagrama eléctrico del movimiento de la cámara -----	72
4.25	Diagrama eléctrico del sistema de visión CMUCAM -----	72
4.26	Mecanismo terminado -----	73
4.27	Montaje del brazo -----	73
4.28	Diagrama eléctrico del brazo mecánico -----	74
4.29	Estructura terminada vista 1 -----	74
4.30-A	Estructura terminada vista 2 -----	75
4.30-B	Estructura terminada robot real -----	75
4.31	Diagrama de flujo bomba azul -----	77
4.32	Diagrama de flujo bomba amarilla -----	78
4.33	Esquema de calibración CMUCAM -----	83
4.34	Software de calibración CMUCAM -----	83
4.35	Diagrama de flujo sistema de visión -----	85
4.36	Imagen dividida en tres -----	86
5.1	Hyperterminal movimiento del robot -----	88
5.2	Hyperterminal reconocimiento de colores -----	88
5.3	Imagen real del movimiento del robot -----	89
5.4	Hyperterminal movimiento del robot con visión simulada -----	90
5.5	Hyperterminal sistema de visión con movimiento simulado -----	90
5.6	Robot en resultados -----	92

LISTA DE TABLAS

1.1	Línea del tiempo -----	2
1.2	Clasificación por generación -----	5
1.3	Clasificación por arquitectura -----	6
1.4	Clasificación por nivel de inteligencia -----	7
1.5	Clasificación por nivel de control -----	7
1.6	Clasificación por su lenguaje de programación -----	8
3.1	Estructura del microcontrolador -----	21
3.2	Familias de microcontroladores -----	22
3.3	Designación del PIC -----	24
3.4	Gamas del PIC -----	25
3.5	Definición de los pines -----	26
3.6	Módulo CCP -----	31
3.7	Comunicación serie -----	33
3.8	Pines RS232 -----	34
3.9	DB25 Y DB9 -----	35
3.10	Clasificación de motores -----	37
3.11	Elementos de un motor de corriente continua -----	40
3.12	Tabla de verdad del Puente H -----	41
3.13	Polarización de un diodo -----	43
3.14	Ecuación de Ackerman -----	48
3.15	G. libertad en mecanismos planos -----	50
3.16	G. libertad en estructuras -----	50
3.17	Tipos de pilas -----	58
4.1	Características del motor -----	64
5.1	Consumo de energía -----	90

RESUMEN

En los últimos años los problemas de seguridad pública en México y en todo el mundo, se han convertido en el centro de la problemática nacional y por las características que han dado origen a esta problemática, no se percibe a corto ni a mediano plazo una solución por falta de un proyecto integral para frenar y resolver el problema.

La problemática a resolver es desarrollar un prototipo en forma de robot que posibilite la búsqueda y desactivación de bombas utilizando en su mayoría material reciclable o de rehúso y el material nuevo que sea de bajo costo pero de buena calidad.

El objeto de estudio abarca el reconocimiento de colores para la localización y desactivación de las bombas, junto con un microprocesador PIC que funciona como el cerebro de nuestro robot.

Con los resultados obtenidos se prevé establecer las bases para seguir con estudios que permitan resolver problemas de seguridad antibombas o explosivos.

CAPÍTULO 1

INTRODUCCIÓN

En este primer capítulo se darán algunas definiciones que se utilizan a través de todo el trabajo, así como una pequeña reseña histórica de la evolución de la robótica.

1.1. ROBÓTICA

Es la ciencia la cual se encarga del estudio, diseño, fabricación y utilización de máquinas programables que sean capaces de realizar tareas de los seres humanos.

1.2. ROBOT

“El término procede de la palabra checa “robota”, que significa “trabajo obligatorio”, y que es idéntico al término ruso que significa trabajo arduo, repetitivo y monótono, y lo usó por primera vez el escritor Karel Capek para referirse en sus obras a máquinas con forma humanoide. Deriva de "robotnik" que define al esclavo del trabajo” [1].

A grandes rasgos, se puede decir que es un manipulador multifuncional el cual puede ser diseñado para mover, ya sea materiales, piezas, herramientas o dispositivos especiales de diferentes tamaños y materiales, mediante movimientos programados.

1.3. BREVE HISTORIA Y EVOLUCIÓN DE LOS ROBOTS

Al igual que otras ciencias, la robótica nació llena de promesas y en pocos años se desarrolló rápida e intensamente ya que en pocos años, alcanzó metas que en aquellos tiempos correspondían a la ciencia ficción. Las ciencias que tuvieron grandes aportaciones, como la informática, en continuo adelanto, y inteligencia artificial, en sus metodologías permitían prever la disponibilidad, en pocos años, de robots dotados de una gran flexibilidad y capacidad de adaptación al entorno.

Herón, en la Grecia antigua, desarrolló un dispositivo similar a las puertas automáticas actuales. Dicho dispositivo era la puerta de un santuario, que se abría automáticamente cuando se encendía un fuego en un altar del santuario y se cerraba al extinguirse el fuego (FIGURA 1.1). El inventó también un dispositivo automático, desde el que salía agua bendita cuando era insertada una moneda. En el Siglo XVIII, un francés, Beaucerson, creó un pato artificial. Este pato era capaz de bañarse, comer, graznar y producir excrementos (FIGURA 1.2). Ha habido mucha más gente, incluyendo ingenieros e inventores, que han creado diversos tipos de muñecos que podían escribir, dibujar y tocar una corneta.

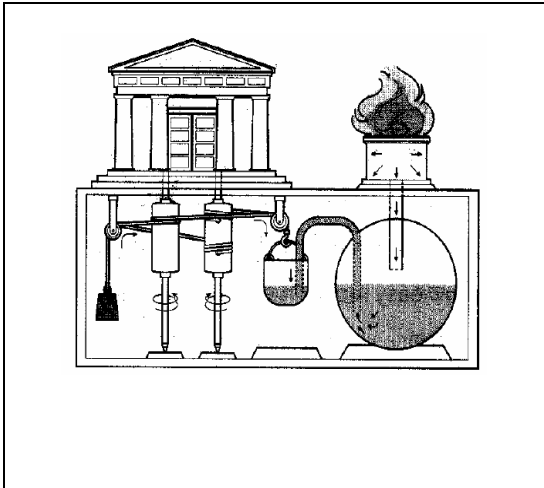


FIGURA 1.1 [2]
PUERTA AUTOMÁTICA

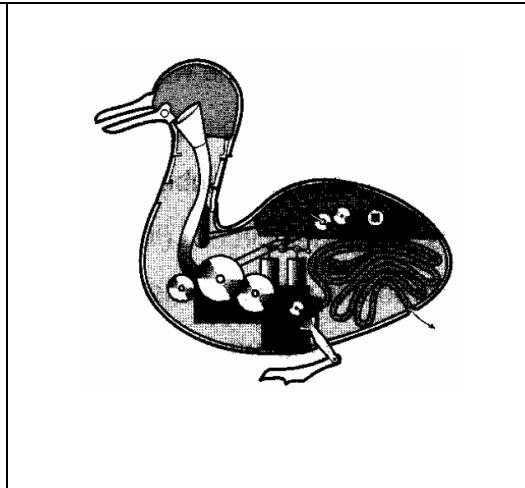


FIGURA 1.2 [2]
PATO MECÁNICO

En el Siglo XX, con el desarrollo de la ciencia y la tecnología, los entonces llamados muñecos automáticos fueron desarrollándose cada vez más para convertirse en sofisticados mecanismos. Los robots de principio de siglo fueron utilizados inicialmente en la tecnología mecánica y eléctrica conjuntamente. Fue hasta 1940 cuando surgió el problema para desarrollar robots como resultado del progreso de la ciencia y la tecnología y fue entonces que Isaac Asimov presentó sus “Tres Principios para la Ingeniería de Robots” o conocidas como las leyes que rigen la construcción de robots.

Hoy en día, no podrían existir robots sin el desarrollo de sistemas de computación y tecnología de control. Las computadoras y la tecnología de control experimentaron un fuerte desarrollo durante la segunda guerra mundial.

La investigación en materia de robots está avanzando rápidamente con el desarrollo de las tecnologías de computación, sensores y actuadores. Actualmente, no sólo se realizan investigaciones en los robots propiamente dichos, sino también en todos los campos relacionados con los robots, como los métodos de desplazamiento, agarre de objetos, comunicacines, inteligencia artificial y otras.

1.4 LÍNEA DEL TIEMPO

AÑO	ACONTECIMIENTO	DESARROLLO
1917	Karel Capek emplea por primera vez la palabra checa “robota” (“trabajo tedioso”) para referirse a un humanoide mecánico.	La palabra apareció en una obra de teatro en Londres donde un personaje del drama conocido como Karel Capek crea diversos hombres artificiales que reemplazan a humanos en sus puestos de trabajo.
1938	Los americanos Willard Pollard y Harold Roselund fabrican la primera máquina para pintar con spray.	La máquina fue fabricada para la empresa DeVilbiss que hasta nuestros días sigue laborando ya con ciertos toques de tecnología más avanzada.
1942	Isaac Asimov publica las tres leyes de la robótica. Estas leyes se ponen de moda especialmente a finales de siglo XX, al introducirse la robótica en los hogares y plantearse un problema ético y de seguridad civil.	<p>Primera ley: Un robot no puede hacerle daño a un ser humano, ni por omisión.</p> <p>Segunda Ley: Un robot debe obedecer a un ser humano siempre que sus órdenes no contradigan la Primera Ley.</p> <p>Tercera Ley: Un robot debe proteger su propia existencia siempre y cuando dicha protección no interfiera con la Primera o Segunda Ley” [3].</p>

1951	Raymond Goertz diseña el primer brazo mecánico manejado a distancia para la Comisión de la Energía Atómica.	Al final de la Segunda Guerra Mundial, donde la supremacía de los tanques en este conflicto puso de manifiesto la necesidad de llevar la revolución industrial del siglo XIX.
1954	George Devol diseña el primer robot programable comercial.	Empezó a comercializarse a partir de 1961. Poco tiempo después Devol fundaría "Unimation", la primera empresa de robótica de la historia.
1959	Se funda el Artificial Intelligence Laboratory en el MIT.	El MIT ha sido la piedra angular en robótica universitaria durante el siglo XX, solo comparable al Robotics Institute de la "Carnegie Melon University".
1959	Sale al mercado el primer robot comercial.	El robot se llamó "Versatran" donde sus siglas significan versatil-transfer.
1965	Se funda el Robotics Institute en la "Carnegie Melon University".	Hoy en día el "RI de la CMU" es como un "supermercado tecnológico", desarrollan cientos de robots gracias a una tremenda red de subvenciones, de donde surge la llamada cámara CMUCAM.
1973	Aparece el primer robot controlado por un mini-ordenador.	Fue el robot "T3". Aproximadamente los mini-ordenadores de esta época pesaban 30 kilos.
1974	Hay 3500 robots en uso en el mundo.	Surge el concepto de Robot de Servicio gracias a su utilización en la industria.
1976	El robot de la NASA "Viking II" aterriza en Marte.	Su característica principal es que disponía de un brazo robótico articulado.
1978	Empiezan a surgir numerosas empresas dedicadas a la fabricación de robots para la industria.	Solo en la década de los 80 y en EEUU surgen aproximadamente un poco más de 10 empresas de gran capital social en el sector Industrial, algunas siguen vigentes hasta estos días.
1986	HONDA, la empresa Japonesa inicia un proyecto para construir un robot humanoide. Su evolución y sus numerosos problemas se mantienen en secreto.	Muchos científicos consideraban que HONDA no iba a poder lograr su objetivo, tanto en Estados Unidos como en Japón.
1997	HONDA presenta P3, un enorme robot humanoide.	"Cuando los resultados se hicieron públicos muchos investigadores que trabajaban en proyectos similares se quedaron sorprendidos ya que sus proyectos estaban a años luz de los resultados obtenidos por HONDA" [4].
1999	SONY lanza "Aibo" un perro-robot.	Los resultados en forma de publicidad gratuita que HONDA ha obtenido provocan una carrera de marketing tecnológico entre empresas de tecnología en Japón.
2000	SONY presenta un pequeño humanoide en la "Robodex 2000".	"Mientras los EEUU miran a sus robots en Marte, Japón mira a sus robots a la cara. Cada uno en su terreno es el rey de la robótica" [4].
2003	Robot humanoide de SONY, Qrio.	HONDA sería el primero en caminar, pero SONY el primero en correr. La carrera está abierta y otras empresas anuncian su propósito de unirse.
2004	Primera edición del "Darpa Grand Challenge".	La guerra de Irak de 2003 y sus numerosas bajas puso en evidencia la necesidad de reducir las bajas militares estadounidenses en futuros conflictos.

2005	Hitachi presenta su robot humanoide	Este robot, llamado EMIEW (excellent mobility and interactive existence as workmate), a diferencia de otros robots tiene dos ruedas en lugar de piernas. Como si se tratara de un malabarista en un monociclo, EMIEW mantiene perfectamente el equilibrio y la vertical, aunque le empujes suavemente, pudiendo desplazarse a una velocidad moderada.
2006	La Academia China de Ciencias, presenta a “La bella señorita Cheng”, robot con características humanas.	Entre las cualidades de la señorita Rong Cheng se encuentran ser una buena conversadora, una excelente escucha y muy buena bailarina, Además, puede caminar a una velocidad de una milla por hora y subir escaleras y por ello, la señorita Cheng será la recepcionista y guía de turistas del Museo Sichuan de Ciencias en Chengdu.
2007	Científicos Europeos desarrollan robots enfermera	El proyecto IWARD, financiado por la UE, desarrolló tres robots enfermera que pueden realizar una serie de tareas de enfermeras, tales como comprobar la temperatura y la presión sanguínea y, en general, aligerar la carga del personal hospitalario para que éste pueda dedicar más tiempo a los pacientes.
2008	NEXI robot del MIT	Nexi es el nuevo robot del MIT Media Lab, un robot "emocional" al que llaman un robot MDS (Mobile/dexterous/Social), que puede mover su cuerpo, manos y cara de forma que refleja emociones humanas. Sus ojos, boca y párpados se mueven al estilo de un dibujo animado, y se mueve gracias a un par de ruedas auto-balanceadas como las del SegWay. Nexi puede crear mapas 3d del entorno gracias a su cámara a color de sus ojos y un sistema de infrarrojos en su nuca. Nexi ayuda al soporte de estudio de comunicación entre humanos y robots
2009	Crean el primer robot flexible en Norteamérica	Científicos norteamericanos de la "Agencia de Proyectos de Investigación Avanzada de la Defensa" (DARPA) y iRobot Coporation desarrollaron el primer robot flexible capaz de modificar su forma. Este prototipo posee una estructura suave que le permite moldear su estructura y adaptarse a casi cualquier superficie.

TABLA 1.1: LÍNEA DEL TIEMPO

1.5. ROBOTS MÓVILES

La idea de un robot móvil con lleva a su vez tres comportamientos claramente definidos, la capacidad de moverse, la capacidad de recabar información del medio y la capacidad de razonamiento para establecer su propio comportamiento. A finales del siglo XIX se presentan las primeras máquinas móviles, pero no será hasta la segunda guerra mundial cuando se realicen los primeros diseños de esta naturaleza.

Los robots móviles están provistos de patas, ruedas u orugas que los capacitan para desplazarse de acuerdo a su programación. Elaboran la información que reciben a través de sus propios sistemas de sensores y se emplean en determinado tipo de instalaciones industriales, sobre todo para el transporte de mercancías en cadenas de producción y almacenes. También se utilizan

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

Los robots móviles se componen de tres partes fundamentales, sistema de movimiento, sistema sensorial y sistema de razonamiento. Existe también, por lo tanto, un amplio espectro de clasificaciones de los mismos en función de diversos criterios (grados de libertad, actuadores usados en los diseños, sistemas sensoriales, número de extremidades, etc.).

1.6. ROBOTS AUTÓNOMOS

Un robot autónomo es aquel capaz de dirigir por sí mismo su comportamiento. Normalmente está dotado de un módulo sensorial completo mediante el cual recibe información del entorno.

La robótica inteligente autónoma es un enorme campo de estudio multidisciplinario, que se apoya esencialmente sobre la ingeniería (mecánica, eléctrica, electrónica e informática) y las ciencias (física, anatomía, psicología, biología, zoología, etología, etc.). Se refiere a sistemas automáticos de alta complejidad que presentan una estructura mecánica articulada gobernada por un sistema de control electrónico y características de autonomía, fiabilidad, versatilidad y movilidad.

En esencia, los “robots inteligentes autónomos” son sistemas dinámicos que consisten en un controlador electrónico acoplado a un cuerpo mecánico. Así, estas máquinas necesitan de sistemas sensoriales adecuados (para percibir el entorno en donde se desenvuelven), de una precisa estructura mecánica adaptable (a fin de disponer de una cierta destreza física de locomoción y manipulación), de complejos sistemas efectores (para ejecutar las tareas asignadas) y de sofisticados sistemas de control (para llevar a cabo acciones correctivas cuando sea necesario).

1.7. CLASIFICACIÓN DE LOS TIPOS DE ROBOTS

La potencia del software en el controlador determina la utilidad y flexibilidad del robot dentro de las limitantes del diseño mecánico y la capacidad de los sensores. De acuerdo a todas sus características, los robots pueden ser clasificados de acuerdo a su generación, a su nivel de inteligencia, a su nivel de control y a su nivel de lenguaje de programación.

1.7.1. CLASIFICACIÓN POR GENERACIÓN

La generación de un robot se determina por el orden histórico de desarrollos en la robótica, los cuales se basan en cinco generaciones.

GENERACIÓN	DESCRIPCIÓN
Primera generación	En ella el sistema de control se basa en paradas fijadas mecánicamente. Como ejemplo podemos encontrar mecanismos de relojería, mecanismos que mueven las cajas musicales o los simples juguetes de cuerda que se siguen usando hasta estos días.
Segunda generación	El movimiento se controla a través de una secuencia numérica almacenada en disco o cinta magnética. Estos son utilizados en la industria automotriz y son de gran tamaño.
Tercera generación	Utilizan las computadoras para su control y tienen cierta percepción de su entorno a través del uso de sensores. Con esta generación se inicia la era de los robots inteligentes y aparecen los lenguajes de programación para escribir los programas de control.

Cuarta generación	Se trata de robots altamente inteligentes con más y mejores extensiones sensoriales, para entender sus acciones y captar el mundo que los rodea. Incorporan conceptos “modélicos” de conducta.
Quinta generación	Actualmente en desarrollo. Esta nueva generación de robots basará su acción principalmente en modelos conductuales establecidos.

TABLA 1.2: CLASIFICACIÓN POR GENERACIÓN [5]

1.7.2. CLASIFICACIÓN POR SU ARQUITECTURA

ARQUITECTURA	DESCRIPCIÓN
Robots Play-back.	Los cuales regeneran una secuencia de instrucciones grabadas, como los utilizados para pintar carros con spray.
Robots controlados por sensores.	Estos tienen un control en lazo cerrado de movimientos manipulados y toman decisiones basados en datos obtenidos por sensores.
Robots controlados por visión.	Los robots pueden manipular un objeto al utilizar información desde un sistema de visión.
Robots controlados adaptablemente	Los robots pueden automáticamente reprogramar sus acciones tomando los datos obtenidos a partir de sus sensores.
Robots con Inteligencia Artificial	Utilizan técnicas de inteligencia artificial para hacer sus propias decisiones y resolver problemas.
Robots médicos	Ocupan fundamentalmente prótesis que se adaptan al cuerpo y están dotados de potentes sistemas de mando.
Androides	Robots que actúan como seres humanos pero su aspecto físico no es tan semejante.
Robots móviles	Provistos de patas, ruedas u orugas que los capacitan para desplazarse de acuerdo su programación. Elaboran la información que reciben a través de sus propios sistemas de sensores y se emplean en determinado tipo de instalaciones industriales, también se utilizan robots de este tipo para la investigación en lugares de difícil acceso o muy distantes, como es el caso de la exploración espacial y las investigaciones o rescates submarinos.

TABLA 1.3: CLASIFICACIÓN POR ARQUITECTURA [5]

1.7.3. CLASIFICACIÓN POR NIVEL DE INTELIGENCIA

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

NIVEL DE INTELIGENCIA	DESCRIPCIÓN
Dispositivos de manejo manual	Son los que están controlados por una persona.
Robots de secuencia variable	Donde un operador puede modificar la secuencia fácilmente
Robots regeneradores	El humano conduce el robot a través de la tarea que tiene que realizar.
Robots de control numérico	El operador alimenta la programación del movimiento, hasta que se enseñe manualmente la tarea.
Robots inteligentes	Los cuales pueden entender e interactuar con cambios en el medio ambiente.

TABLA 1.4: CLASIFICACIÓN POR NIVEL DE INTELIGENCIA [6]

1.7.4. CLASIFICACIÓN POR NIVEL DE CONTROL

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

NIVEL DE CONTROL	DESCRIPCIÓN
Nivel de inteligencia artificial	Donde el programa aceptará un comando como "levantar el producto" y descomponerlo dentro de una secuencia de comandos de bajo nivel basados en un modelo estratégico de las tareas.
Nivel de modo de control	Donde los movimientos del sistema son modelados, para lo que se incluye la interacción dinámica entre los diferentes mecanismos, trayectorias planeadas, y los puntos de asignación seleccionados.
Nivel de servosistemas	Donde los actuadores controlan los parámetros de los mecanismos con el uso de una retroalimentación interna de los datos obtenidos por los sensores, y la ruta es modificada sobre la base de los datos que se obtienen de sensores externos. Todas las detecciones de fallas y mecanismos de corrección son implementados en este nivel.

TABLA 1.5: CLASIFICACIÓN POR NIVEL DE CONTROL [6]

1.7.5. CLASIFICACIÓN POR LENGUAJE DE PROGRAMACIÓN

En la clasificación final se considerara el nivel del lenguaje de programación. La clave para una aplicación efectiva de los robots para una amplia variedad de tareas, es el desarrollo de lenguajes de alto nivel. Existen muchos sistemas de programación de robots, aunque la mayoría del software más avanzado se encuentra en los laboratorios de investigación. Los sistemas de programación de robots caen dentro de tres clases.

LENGUAJE DE PROGRAMACIÓN	DESCRIPCIÓN
Sistemas guiados	En el cual el usuario conduce el robot a través de los movimientos a ser realizados.
Sistemas de programación de nivel-robot	En los cuales el usuario escribe un programa de computadora para especificar el movimiento y el censado.
Sistemas de programación de nivel-tarea	En el cual el usuario especifica la operación por sus acciones sobre los objetos que el robot manipula.

TABLA 1.6: CLASIFICACIÓN POR SU LENGUAJE DE PROGRAMACIÓN [6]

CAPÍTULO 2

PLANTEAMIENTO DEL PROBLEMA

En este capítulo se describirán las características generales de los robots desactivadores de bombas que ya están a la venta y los que son todavía un proyecto por estar en pruebas. También se explicara la razón por la cual se decidió entrar al concurso y las reglas del mismo.

3.1. ROBOTS DESACTIVADORES DE BOMBAS.

En la actualidad existen robots, que ya están a la venta, los cuales están dedicados a la búsqueda de elementos explosivos. Las características generales con las que cuentan son: pueden estar controlados a distancia por un operador el cual está viendo en tiempo real lo que ve el robot ya que están dotados de cámaras las cuales pueden trabajar en terrenos oscuros o con luz; tienen un sistema de tracción de orugas para poder explorar diferentes terrenos así como poder subir escaleras, tienen sensores para poder evadir obstáculos, y para poder manipular los artefactos que puedan ser explosivos, tienen brazos mecánicos, dotados de la tecnología llamada telepresencia que permite sincronizar el movimiento del brazo con el brazo o la cabeza del operador, es decir es una simulación que permite actuar o trabajar al operador como si estuviera en el lugar de los hechos. Por otra parte sus manipuladores pueden extenderse o contraerse para poder llegar a diferentes alturas y terrenos. Algunos cuentan ya con sensores que determinan si los artículos sospechosos son peligrosos o inofensivos, es decir detectan vapores explosivos y que tan alta o baja es su concentración, otros pueden recoger muestras de aire e incluso detectar a cierta distancia la radiación que es emanada en las guerras.

3.1.1. ROBOT ASENDRO EOD

Este robot reconoce y desactiva explosivos a distancia, cuenta con sensores para evadir obstáculos y gracias a su sistema de tracción puede recorrer diferentes terrenos y subir escaleras.

Cuenta con un manipulador y una pinza, para poder recoger los objetos. El manipulador puede extenderse hasta 2.1 metros y cargar hasta 5 kilogramos. En el manipulador se cuenta con tecnología de telepresencia, lo cual va a permitir sincronizar el brazo del operador con el manipulador y no arriesgar la vida del operador.



FIGURA 2.1: ROBOT ASENDRO EOD [7]

3.1.2. PACKBOT 510

Este robot puede identificar explosivos. El sistema de tracción con el que cuenta lo hace muy rápido al avanzar, se adapta a cualquier terreno en donde lo pongan, pudiendo subir escaleras y evadir obstáculos. Cuenta con un manipulador y una pinza que gira 360° y levanta hasta 13 kilogramos.

Tiene un control como el del videojuego similar a un joystick para su fácil operación ya que es controlado a distancia, y para esto cuenta con cámaras para ver lo que pasa en tiempo real. Puede ser utilizado en zonas oscuras y de difícil acceso. Sus distribuidores son iRobot



FIGURA 2.2: ROBOT PACKBOT 510 [8]

3.1.3. WOLVERINE

Este robot es un prototipo no está a la venta. Y fue elaborado en los laboratorios Sandia, en Estados Unidos la noticia fue difundida el 30 de enero 2009. Fue denominado Wolverine y liberará al operador de algunas de las tareas que deben estar a su cargo en los momentos críticos de su labor. Su misión es realizar operaciones que de otro modo pondrían en peligro a un experto. Está equipado con cámaras, pinzas y otros sensores y herramientas, útiles para responder a una amenaza de bomba.

La estructura del Wolverine es diseño de la empresa Remotec, y se le ha agregado software de inteligencia de los laboratorios Sandia. El software se denomina SMART (por las siglas en inglés de Arquitectura Modular de Sandia para Robótica y Teleoperación).

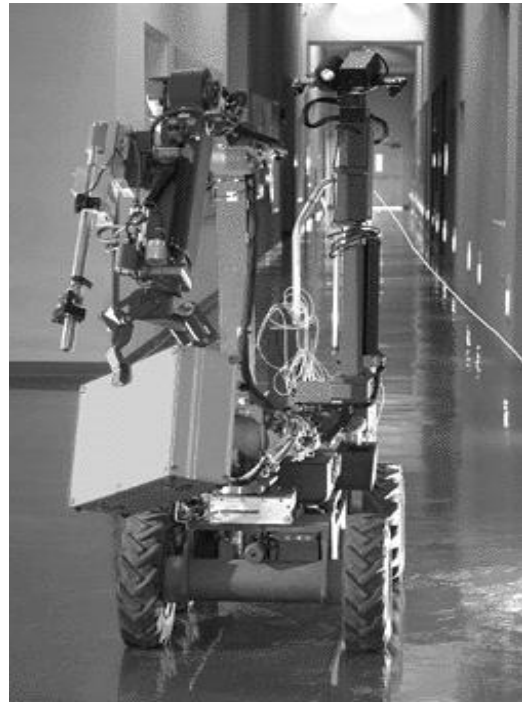


FIGURA 2.3: ROBOT WOLVERINE [9]

3.1.4. ROBOT Israelita

En Dimona, Israel, el 12 de febrero de 2008, donde hubo un ataque con dos terroristas suicidas, a los cuales la policía les disparo al ver que intentaban activar sus bombas. Inmediatamente se auxiliaron de uno de sus robots para ir a desactivar, el robot primero le quito la chamarra que traía el explosivo y en el mismo piso le paso por encima para evitar que explotara. Este robot cuenta con manipuladores, cámaras, teleoperación, orugas para explorar cualquier terreno y sobre todo tiene un peso capaz de aplastar y matar a un ser humano.



FIGURA 2.4: ROBOT ISRAELITA [10]

3.1.5. ROBOT DE TLAXCALA

En Calpulalpan, Tlaxcala, el 18 de marzo de 2008, los empleados de la gasolinera "López Calderón", detectaron una presunta bomba en la caja de registro del sanitario del restaurante ubicado en el interior de ese negocio. Los especialistas de la Policía Federal (PF) intentaban desactivar el artefacto que estaba atado a un teléfono celular. Para esa operación contaron con el apoyo de un robot antibombas, que no tenía manipuladores para desactivarla por sí mismo, sólo podía ir por ella vía control remoto; sin embargo, el operador no puede estar tan lejos del robot por lo que se sigue arriesgando la vida.

Los operadores tienen que quitarle la bomba al robot para llevarla a analizar a sus laboratorios; es decir, no cuenta con sensores para detectar algunas sustancias en el instante.



FIGURA 2.5: ROBOT. TLAXCALA [11]

2.2 PROYECTO: ROBOT QUE SIMULA LA DESACTIVACIÓN DE BOMBAS

En los últimos años los problemas de seguridad pública en materia de terrorismo en México y en todo el mundo, se han convertido en los principales puntos para resolver, ya que se ha incrementado el número de alertas de objetos explosivos en sitios públicos y en horas donde se encuentran más transitados.

Específicamente, en México no se percibe a corto ni a mediano plazo una solución por falta de un proyecto que se dedique a la construcción y/o estudio de este tipo de robots, el cual traería consigo el beneficio de un bajo costo con la misma calidad de los robots que venden en el extranjero, y el mantenimiento o soporte se daría en esta región, ya que se cuenta con tecnología que sigue poniendo en peligro las vidas de las personas e incluso de los operadores que manejan este tipo de robots ya que son controlados vía control remoto, donde el operador no puede estar tan

lejos, sus manipuladores no alcanzan grandes distancias y todavía no cuentan con telepresencia, sensores que puedan detectar sustancias y sobre todo pensar en semiautonomía o autonomía, ya que muchos a pesar de estar controlados por un operador cuando se encuentran con obstáculos son capaces de evadir y buscar nuevas rutas.

Tomando en cuenta los problemas de los que se han estado hablando, surge una convocatoria para entrar a un concurso que tiene varias categorías, entre ellas la categoría open en la cual la meta es desactivar dos simulaciones de bombas, que fue lo que nos llamó la atención porque con esos concursos surgen ideas de diseños, materiales y técnicas para la elaboración de este tipo de robots y así poder contribuir con algo a la sociedad.

2.3 DESCRIPCIÓN DE LA CATEGORÍA OPEN

En esta categoría el primer requisito es que tu robot sea construido desde cero, es decir que no sea un robot comercial, totalmente autónomo a la hora de realizar la tarea, el cual no exceda los 30 x 30 x 30 cm al estar apagado y una vez prendido puede excederse en las medidas que se piden.

Su tarea consiste en desactivar dos bombas, una azul y otra amarilla, en el orden que se desee. El procedimiento para desactivarla es quitar primero el cable rojo y después el verde.

El área de trabajo del robot es un tablero de fibra de madera, de densidad media, conocido como MDF, el cual tendrá una pared de 6 cm., 2 entradas y 2 rampas en una de las esquinas con una inclinación de 45°.

Descripción de la pista:

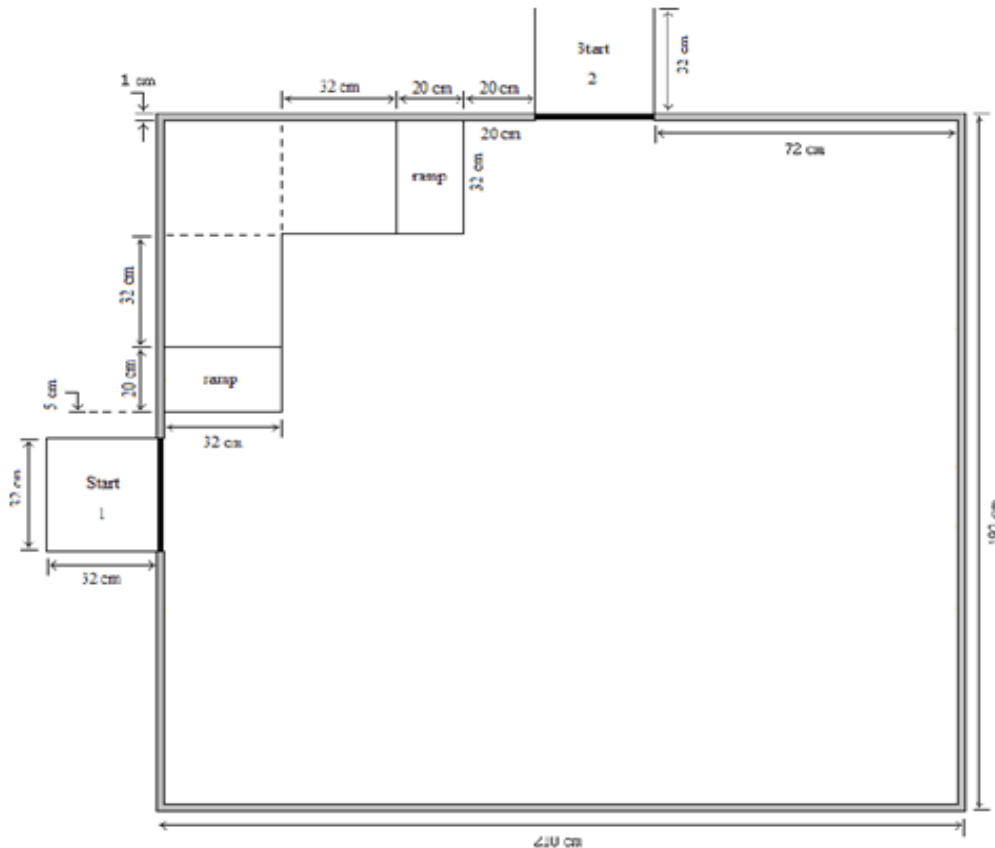


FIGURA 2.6: MEDIDAS DE LA PISTA [12]

Uno de los puntos que se piden es que el robot sea capaz de trabajar igual por cualquiera de las dos entradas que se ponga ya que la posición va a ser aleatoria.

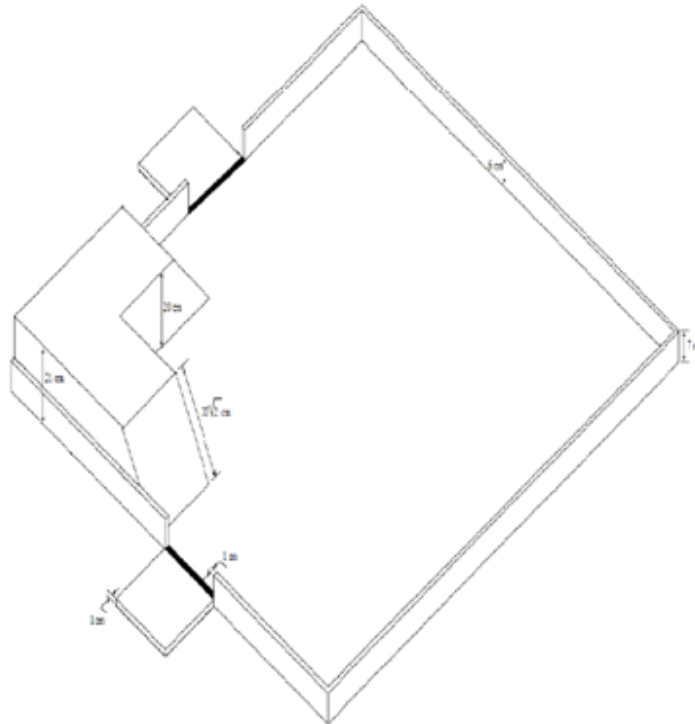


FIGURA 2.7: VISTA DE ARRIBA DE LA PISTA [12]

Las bombas se localizaran una en la parte de arriba de las rampas, y la otra que se encuentra en la parte de abajo podrá estar localizada en cualquiera de los 6 puntos que se muestran en la figura 2.3-C, dependiendo del criterio del juez. La bomba azul siempre va a estar colocada en la parte de arriba y la amarilla en la parte de abajo

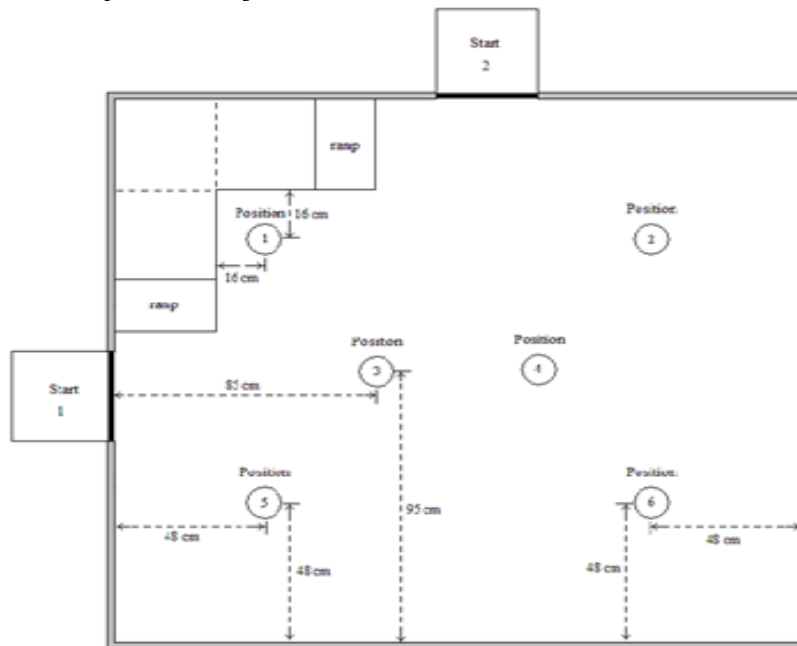


FIGURA 2.8: POSICIONES DE LA BOMBA [12]

Las características de las bombas, como se había mencionado antes, son: una de color azul y otra amarilla, en forma de cubo, las dos van a contar con tres cables de colores rojo, verde y negro. El material del que están hechas es de madera, sus dimensiones son de 10 x10 x 10 cm, colocadas en una base blanca también de madera de 10 x 10 x 5 cm, con un peso de 400 g ($\pm 10\%$) con los cables, las bases van a tener un peso de 230 g ($\pm 10\%$) y van a ser de color blanco. Los cables van a ser de 4 mm de diámetro, de calibre 18 AWG, con una longitud de 9 cm.

La bomba amarilla va a tener en una sola cara los cables que se van a desactivar, y la bomba azul va a tener en una cara 2 cables y en otra cara se colocara el tercer cable, la posición de los cables va a ser aleatoria de acuerdo al criterio del juez.

En cualquiera de las dos desactivaciones no es permitido usar objetos cortantes y/o sustancias peligrosas.

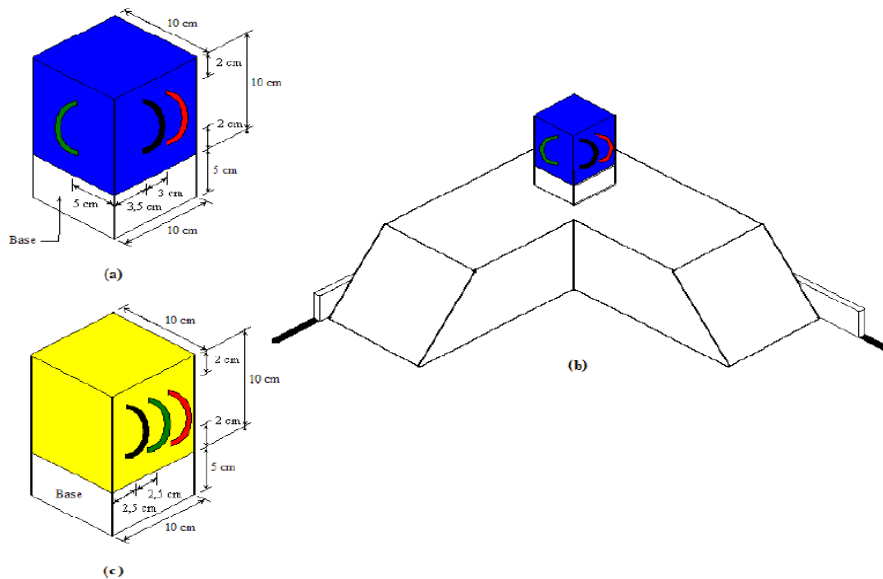


FIGURA 2.9: MEDIDAS DE LAS BOMBAS [12]

El puntaje para poder ganar la competencia es determinado por la acumulación de puntos, ya que por cada bomba desactivada son 3 puntos positivos y en caso de que varios equipos tengan la misma puntuación se decidirá por el que haya hecho menor tiempo.

El tiempo máximo que puede tener cada equipo son 10 minutos, excepto si se desactivaron una o las dos bombas. Es permitido que al desactivar una bomba el equipo pueda parar el tiempo y así dar por terminada su prueba, de acuerdo a su estrategia.

Cuando uno de los robots tira cualquiera de las dos bombas es merecedor a 3 puntos negativos, porque en esta simulación el tirarla equivale a que en la vida real la bomba explote. También tendrá tres puntos negativos si quita el cable negro antes de quitar los cables rojo y verde, tampoco se pueden quitar de su base en la que se encuentran colocadas. En este caso, si no se han consumido los 10 minutos que se dan en la prueba, el equipo puede decidir volver a empezar y en caso de que desarmen las dos bombas solo podrá tener un máximo de 3 puntos por la amonestación que recibió, y si solo desactiva una bomba quedará con cero puntos.

Durante la prueba el equipo puede pedir un reinicio, el cual es merecedor de un punto negativo. Este reinicio se pide cuando el equipo se percata que el robot no está haciendo la tarea que se desea y posiblemente consuma los 10 minutos que dan, o tuvo alguna falla en los componentes del robot y pueda ser reparado fácilmente, teniendo en cuenta que al pedir el reinicio el cronómetro sigue corriendo.

Mientras se realiza la prueba, ningún integrante del equipo podrá tocar o manipular al robot, en caso de hacerlo recibirá un punto negativo.

2.4 ALCANCE DEL PROYECTO

La problemática que nos planteamos fue desarrollar un prototipo de robot que posibilite la búsqueda y desactivación de bombas utilizando en su mayoría material reciclable o de rehúso y que el material nuevo sea de bajo costo pero de buena calidad el cual resista caídas, golpes y no dure solo la competencia, que no sea tan pesado, y que el material a pesar de que es reciclado sea de fácil reemplazo.

El estudio se centra en el desarrollo de software y hardware para hacerlo totalmente autónomo, permitiéndole evadir obstáculos, así como subir rampas y tratar de explorar cualquier terreno. Dicho estudio abarca reconocimiento de colores para la localización y desactivación de las bombas, un brazo mecánico para la manipulación de las mismas, un sistema de locomoción o base móvil con movimiento de par diferencial, diseño y construcción de un sistema embebido de control y otro de potencia para alimentar los motores, los cuales sean de uso general, es decir, que no se limiten sólo para ser usados en este robot si no que puedan ser ocupados en otras plataformas móviles.

Esto se realizará tratando de involucrar los conocimientos adquiridos en las materias que se llevaron a lo largo de toda la carrera.

2.5 ESTRATEGIA QUE SE SIGUIÓ PARA LA DESACTIVACIÓN DE BOMBAS

Cuando el robot es colocado en cualquiera de las dos entradas, primero busca enfrente de él la bomba, si en verdad la llega a ver se dirige hacia ella, en caso contrario va a avanzar cierto tiempo, y va a volver a localizar la bomba en caso de que no la vuelva a ver enfrente, procede a girar para buscarla a su alrededor. En cualquiera de las formas que localice la bomba al dirigirse a ella va a tratar de llegar de frente a una cara de la bomba, procurando llegar lo más centrado posible.

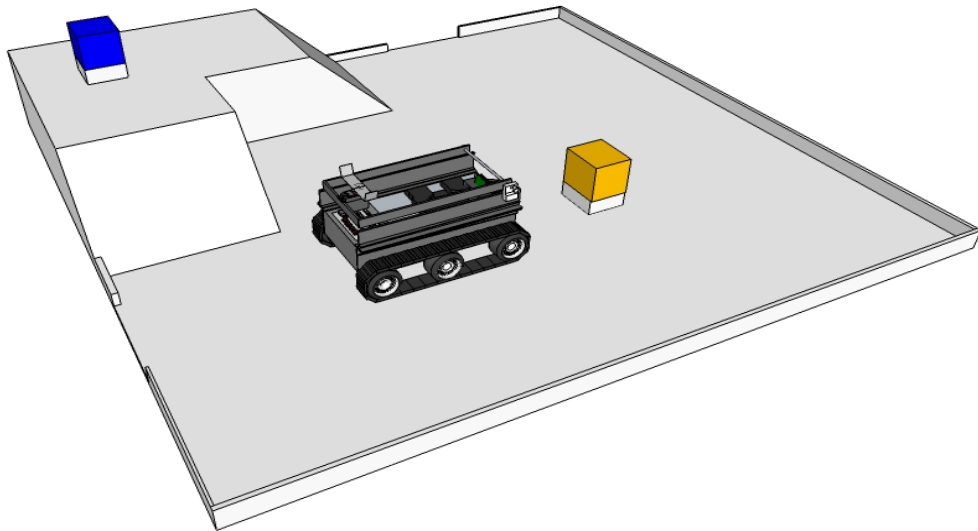


FIGURA 2.10: ESTRATEGIA EN LA PISTA

Una vez que se encuentra de frente a la bomba se asegura de que en verdad sea la bomba y no un objeto de color amarillo, después procede a buscar si en esa cara que quedo de frente se localizan los cables que tiene que quitar. En el caso de que no sea esa la cara de los cables, el robot acciona su brazo manipulador, colocándolo en la parte de arriba de la bomba para poder girarla, y en cada giro checa si esa cara contiene los cables.

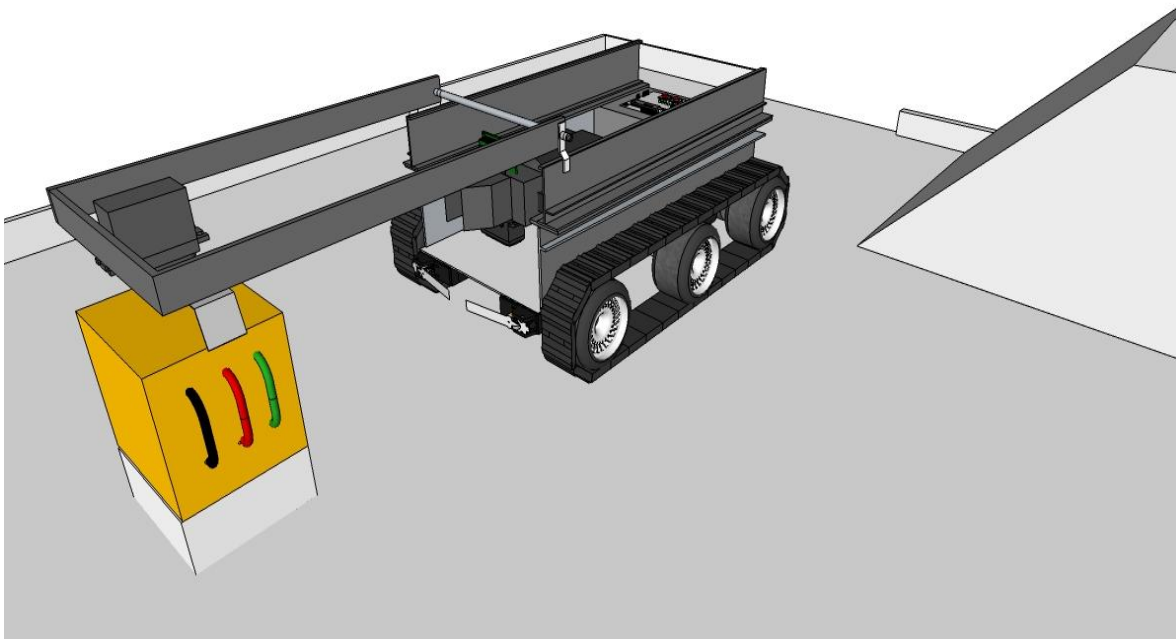


FIGURA 2.11: ESTRATEGIA 2 (USO DEL MANIPULADOR)

Durante la búsqueda de los cables, una vez que se aseguro de tener la cara de los cables en frente de él, empieza con un barrido de izquierda a derecha para empezar a localizar el cable rojo; una vez que lo localizó procede a quitarlo con un gripper, asegurándose de quitarlo totalmente. Posteriormente hace el mismo procedimiento pero ahora con el cable verde.

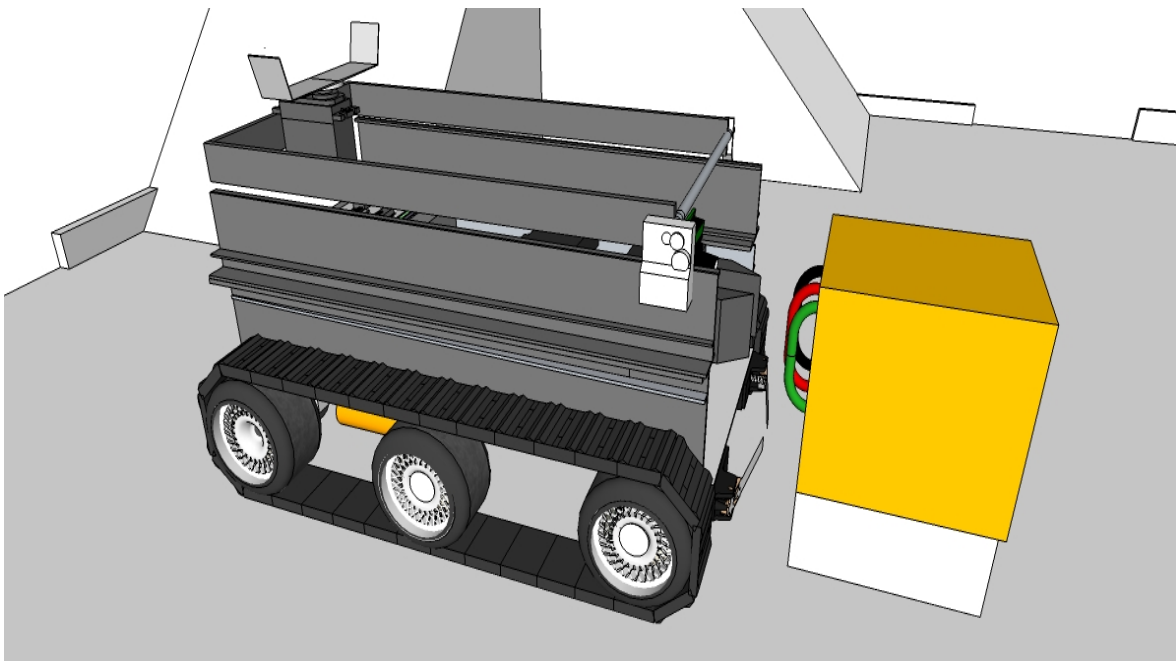


FIGURA 2.12: BÚSQUEDA DE LOS CABLES

Mientras hace la búsqueda puede chocar con obstáculos como la pared de la pista en la que se realiza la prueba y puede evadirla fácilmente.

Para la búsqueda de la bomba azul podemos elegir por medio del switches del sistema embebido en que entrada empezar y de acuerdo a esto el robot debe girar a la izquierda o la derecha, una vez que se encuentra de frente a la rampa, sube variando su velocidad y va tratándose de alinear para llegar al centro de la bomba, una vez que está en el centro procede a buscar los cables de la misma forma que lo hace con la bomba amarilla, rotándola hasta encontrar los cables.

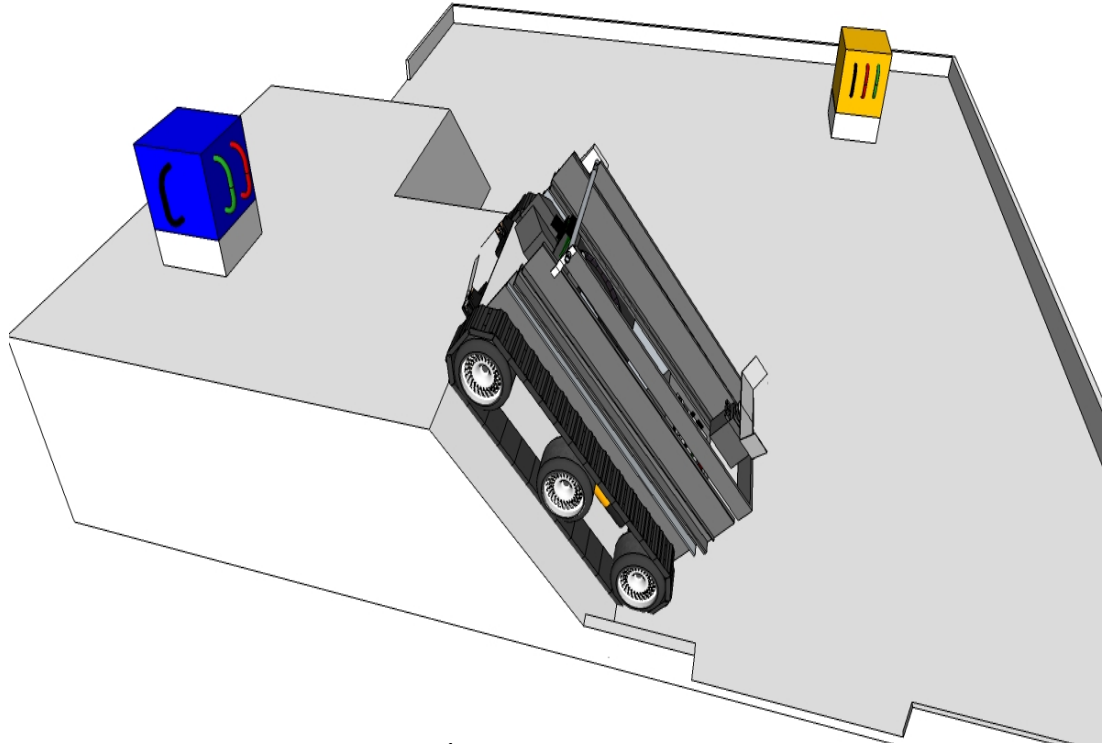


FIGURA 2.13: BÚSQUEDA DE LA BOMBA AZUL

CAPÍTULO 3

MARCO TEÓRICO

En este tercer capítulo se dará una reseña acerca de la teoría que se ocupó en todas las partes del robot.

3.1. MICROPROCESADORES

El microprocesador se puede ver como un chip o un tipo de componente electrónico en cuyo interior existen millones de elementos llamados transistores, cuya combinación permite realizar el trabajo que tenga encomendado dicho chip.

3.2. MICROCONTROLADORES

Un Microcontrolador es un circuito integrado que cumple las funciones de cerebro de cualquier aplicación, el cual incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y unidades de entrada y salida es decir, incorpora todos los bloques funcionales de un Sistema Microprocesador en un único encapsulado. Para funcionar sólo necesitan una tensión continua estable (por ejemplo 5V, 3.3V, 2.5V, 1.5V) y un oscilador. A grandes rasgos interpretan (decodifican) combinaciones de bits (instrucciones) y generan señales digitales internas y/o externas para “ejecutar” de manera continua una secuencia de instrucciones (programa) que permita controlar un sistema o subsistema electrónico.

Aunque sus prestaciones son limitadas si las comparamos con las de cualquier computadora personal. Estos dispositivos los podemos ver como las computadoras que están alrededor de todos los humanos, corriendo programas y haciendo cálculos silenciosamente sin la interacción de nadie. Estas las podemos ver en juguetes, automóviles, edificios inteligentes, incluso en aparatos electrodomésticos. Dichos microcontroladores, una vez conectados a una máquina, deben ser programados y finalmente pueden empezar a trabajar automáticamente.

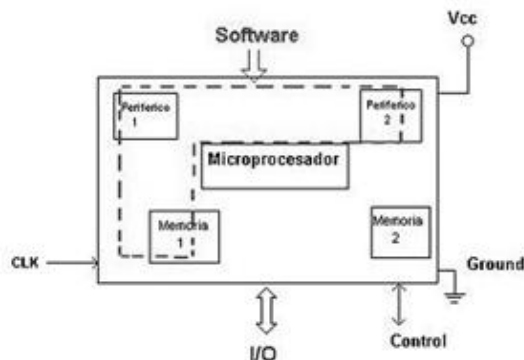


FIGURA 3.1: MICROCONTROLADOR [13]

El nombre se le da por Micro porque son pequeños, y controladores, porque controlan máquinas o incluso otros controladores, esto quiere decir que están diseñados para conectarse a máquinas.

Cientos de microcontroladores están disponibles algunos son programados una vez y producidos para aplicaciones específicas, tales como controlar su horno microondas. Otros son “reprogramables”, que quiere decir que pueden ser usados una y varias veces para diferentes aplicaciones como ejemplo tenemos los utilizados en proyectos escolares.

Generalmente son diseñados para disminuir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la CPU, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. Por ejemplo el control de un electrodoméstico sencillo como una batidora, utilizará un procesador muy pequeño (4 u 8 bit). En cambio un reproductor de música y/o vídeo digital (mp3 o mp4) requerirá de un procesador de 32 bit o de 64 bit y de uno o más Códec de señal digital (audio y/o vídeo).

3.2.1. ARQUITECTURA INTERNA DEL MICROCONTROLADOR

Como ya hemos visto, un microcontrolador es un dispositivo complejo, a continuación se analizan los más importantes:

PROCESADOR

Es la parte encargada del procesamiento de las instrucciones. Debido a la necesidad de conseguir elevados rendimientos en este proceso, se ha desembocado en el empleo generalizado de procesadores de arquitectura Harvard frente a los tradicionales que seguían la arquitectura de Von Neumann.

ARQUITECTURA VON NEUMANN

Esta se caracteriza porque la CPU se conectaba con una memoria única, donde coexistían datos e instrucciones, a través de un sistema de buses.

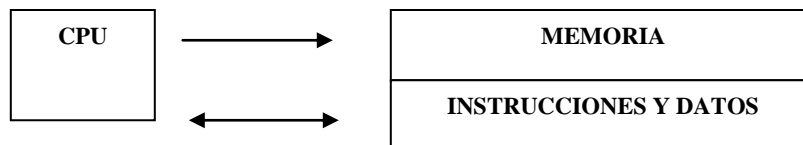


FIGURA 3.2: A. VON NEUMANN

ARQUITECTURA HARVARD

En la arquitectura Harvard son independientes la memoria de instrucciones y la memoria de datos y cada una dispone de su propio sistema de buses para el acceso. Esta dualidad, además de propiciar el paralelismo, permite la adecuación del tamaño de las palabras y los buses a los requerimientos específicos de las instrucciones y de los datos.

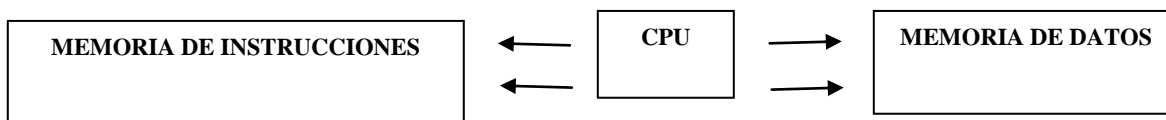


FIGURA 3.3: A. HARVARD

El procesador de los modernos microcontroladores responde a la arquitectura RISC (Computadores de Juego de Instrucciones Reducido), que se identifica por poseer un repertorio de instrucciones máquina pequeño y simple, de forma que la mayor parte de las instrucciones se ejecutan en un ciclo de instrucción.

Otra aportación frecuente que aumenta el rendimiento de la computadora es el fomento del paralelismo implícito, que consiste en la segmentación del procesador (pipe-line), descomponiéndolo en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.

Entonces podemos decir que estructura interna de los PIC 16C5XX se basa en registros con memoria y buses separados para las instrucciones y los datos, llamada arquitectura Harvard. La memoria y el bus de datos (RAM) son de 8 bits de ancho, mientras que la memoria EPROM y su bus tienen 12 bits.

Esta estructura emplea 2 espacios de memoria diferentes, uno para datos y otro para programas y además se utilizan 2 buses distintos: uno para el tráfico entre el CPU y los datos y otro para la comunicación entre la memoria de programa y el CPU. Esto permite que, mientras una instrucción se ejecuta utilizando el bus de datos (8 bits). La siguiente se está leyendo desde la memoria de programa y cargándose en el registro de instrucción utilizando el bus de instrucciones de 12 bits.

3.2.2. ESTRUCTURA BÁSICA DE UN MICROCONTROLADOR

Aún cuando el microcontrolador es una computadora embebida dentro de un circuito integrado, se compone de un núcleo y un conjunto de circuitos adicionales. Dentro del núcleo se encuentran el procesador y la memoria, todo ello estructurado de forma tal que conforme una arquitectura de computadora.

ESTRUCTURA	DESCRIPCIÓN
CPU o PROCESADOR	Es el cerebro del sistema que procesa todos los datos que viajan a lo largo del bus.
MEMORIAS	Están formadas por una no volátil (ROM, EEPROM, FLASH) donde se almacenan los programas y una volátil (RAM) donde se almacenan los datos.
RELOJ PRINCIPAL	Normalmente todos los microcontroladores tienen incorporados circuitos osciladores para el funcionamiento de éstos.
PUERTOS DE ENTRADA Y SALIDA	Soportan las líneas que comunican al microcontrolador con los periféricos externos.
WATCHDOG (PERRO GUARDIÁN)	Se emplea para provocar una reinicialización cuando el programa queda bloqueado.
PROTECCIÓN ANTE FALLO DE ALIMENTACIÓN O BROWNOUT	Circuito que resetea al microcontrolador cuando la tensión de alimentación baja de un cierto límite.
TEMPORIZADORES	Para controlar periodos de tiempo.
CONVERTIDORES A/D Y D/A. (ANALÓGICO/DIGITAL Y DIGITAL/ANALÓGICO)	Como es muy frecuente el trabajo con señales analógicas, éstas deben ser convertidas a digital y por ello muchos microcontroladores incorporan un conversor A/D, el cual se utiliza para tomar datos de varias entradas diferentes que se seleccionan mediante un multiplexor.
MODULADORES DE ANCHO DE PULSOS	Los PWM (Pulse Width Modulator) son periféricos muy útiles sobre todo para el control de motores, sin embargo hay un grupo de aplicaciones que pueden realizarse con este periférico.
PUERTOS DE COMUNICACIÓN SERIE	Este periférico está presente en casi cualquier microcontrolador, normalmente en forma de UART (Universal Asynchronous Receiver Transmitter) o USART (Universal Synchronous Asynchronous Receiver Transmitter) dependiendo de si permiten o no el modo sincrónico de comunicación. El destino común de este periférico es la comunicación con otro microcontrolador o con una PC y en la

	<p>mayoría de los casos hay que agregar circuitos externos para completar la interfaz de comunicación. La forma más común de completar el puerto serie es para comunicarlo con una PC mediante la interfaz EIA-232 (más conocida como RS-232), es por ello que muchas personas se refieren a la UART o USART como puerto serie RS-232, pero esto constituye un error, puesto que este periférico se puede utilizar para interconectar dispositivos mediante otros estándares de comunicación.</p>
<p>PUERTO COMUNICACIÓN SERIE SINCRÓNICO</p>	<p>Este tipo de periférico se utiliza para comunicar al microcontrolador con otros microcontroladores o con periféricos externos conectados a él, mediante las interfaces SPI (Serial Peripheral Interface) o I2C (Inter-Integrated Circuit). A pesar de que es también un tipo de puerto serie, es costumbre tratarlo de forma diferenciada respecto a la UART/USART porque las interfaces SPI e I2C aparecieron mucho después que la UART/USART, su carácter es únicamente sincrónico y no están diseñadas para interconectar al sistema con otros dispositivos independientes como una PC, sino para conectar al microcontrolador dispositivos tales como memorias, pantallas LCD, convertidores A/D o D/A.</p>
<p>OTROS PUERTOS DE COMUNICACIÓN</p>	<p>En un mundo cada vez más orientado a la interconexión de dispositivos, han aparecido muchas interfaces de comunicación y los microcontroladores no se han quedado atrás para incorporarlas, es por ello que podemos encontrar algunos modelos con puertos USB (Universal Serial Bus), CAN (Controller Area Network), Ethernet, puerto paralelo entre otros.</p>
<p>CONTROL DE INTERRUPCIONES</p>	<p>Las interrupciones constituyen quizá el mecanismo más importante para la conexión del microcontrolador con el mundo exterior, sincronizando la ejecución de programas con acontecimientos externos. El funcionamiento de las interrupciones es similar al de las subrutinas de las cuales se diferencian principalmente en los procedimientos que las ponen en marcha. Así como las subrutinas se ejecutan cada vez que en el programa aparece una instrucción CALL, las interrupciones se ponen en marcha al aparecer en cualquier instante un evento externo al programa, es decir por un mecanismo hardware. Las interrupciones son tan eficaces que permiten que el procesador actúe como si estuviese haciendo varias cosas a la vez cuando en realidad se dedica a la misma rutina de siempre, ejecutar instrucciones una detrás de la otra.</p>
<p>COMPARADORES</p>	<p>Son circuitos analógicos basados en amplificadores operacionales que tienen la característica de comparar dos señales analógicas y dar como salida los niveles lógicos '0' o '1' en dependencia del resultado de la comparación. Es un periférico muy útil para detectar cambios en señales de entrada de las que solamente nos interesa conocer cuando está en un rango determinado de valores.</p>

Tabla 3.1: ESTRUCTURA DEL MICROCONTROLADOR

3.2.3. FAMILIA DE MICROCONTROLADORES

Fabricante	8 bits	16 bits	32 bits
ATMEL AVR	ATmega8,89Sxxxx Familia similar 8051	ATmega16	
FREESCALE (antes Motorola)	68HC05, 68HC08, 68HC11, HCS08	68HC12, 68HCS12, 68HCSX12, 68HC16	683xx, PowerPC Architecture, ColdFire

HITACHI	H8		
HOLTEK	HT8		
INTEL	MCS-48 (familia 8048) MCS51 (familia 8051) 8xC251	MCS96, MXS296	
NATIONAL SEMICONDUCTOR	COP8		
MICROCHIP	Familia 10f2xx Familia 12Cxx Familia 12Fxx, 16Cxx y 16Fxx 18Cxx y 18Fxx	dsPIC30FXX y dsPIC33F de 16 bits	PIC32
NEC	78K		
PARALLAX			
ST	ST 62,ST 7		
TEXAS INSTRUMENTS	TMS370, MSP430		
ZILOG	Z8, Z86E02		
SILABS	C8051		

TABLA 3.2: FAMILIAS MICROCONTROLADORES [14]

3.3. DIFERENCIA ENTRE MICROCONTOROLADOR Y MICROPROCESADOR

A simple vista es fácil confundir los términos de microcontrolador y microprocesador empezando desde el nombre al llamarlos “micros”, pero hay una gran diferencia. Como un microcontrolador podemos entenderlo concretamente como un sistema completo, con unas prestaciones limitadas que no pueden modificarse y que puede llevar a cabo las tareas para las que ha sido programado de forma autónoma. Un microprocesador, en cambio, es simplemente un componente que conforma el microcontrolador, que lleva a cabo ciertas tareas y que, en conjunto con otros componentes, forman un microcontrolador.

Las principales características que diferencian a un microcontrolador de un microprocesador son:

1. Son sistemas cerrados, ya que contiene todos los elementos de un computador en un solo chip, frente a los microprocesadores que son sistemas abiertos, ya que sacan las líneas de los buses de datos, direcciones y control al exterior, para la conexión de memorias, interfaces de E/S, etc.
2. Son de propósito específico, es decir, son programados para realizar una única tarea, mientras que los microprocesadores son de propósito general.

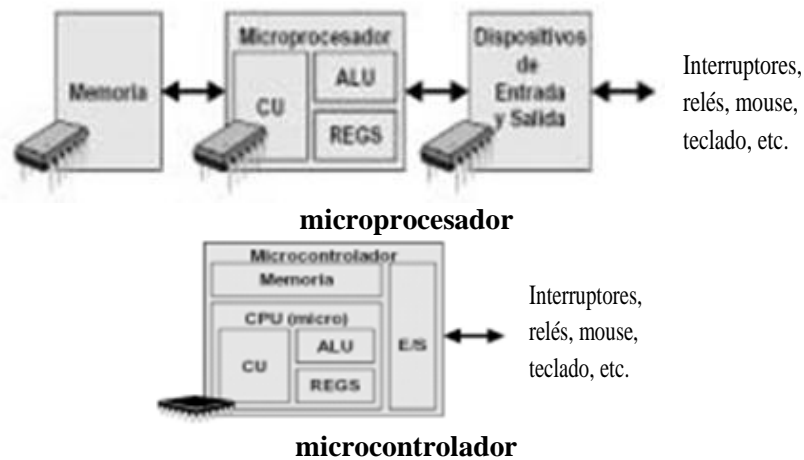


FIGURA 3.4: DIFERENCIA ENTRE MICROCONTROLADOR Y MICROPROCESADOR [15]

3.4. MICROCONTROLADORES PIC

Un PIC es un microcontrolador basado en memoria EPROM/FLASH desarrollado por Microchip Technology.

El nombre actual no es un acrónimo. En realidad, el nombre completo es PICmicro, aunque generalmente se utiliza como Peripheral Interface Controller (Controlador de Interfaz Periférico).

Son una familia de microcontroladores tipo RISC fabricados por Microchip Technology Inc. y derivados del PIC1650, originalmente desarrollado por la división de microelectrónica de General Instruments.

Los microcontroladores PIC fueron los primeros microcontroladores RISC, es decir, microcontroladores con un juego de instrucciones reducido. El hecho de ser procesadores de tipo RISC generalmente implica simplicidad en los diseños, permitiendo más características a bajo coste. Los principales beneficios de esta simplicidad en el diseño son que los microcontroladores se implementan en chip muy pequeños, con pocos pines, y tienen un consumo de potencia muy bajo.

3.4.1 RESEÑA HISTÓRICA DE LOS PIC

En 1965 GI formó una división de microelectrónica, destinada a generar las primeras arquitecturas viables de memoria EPROM y EEPROM. De forma complementaria GI Microelectronics Division fue también responsable de desarrollar una amplia variedad de funciones digitales y analógicas en las familias AY3-xxxx y AY5-xxxx.

GI también creó un microprocesador de 16 bit, denominado CP1600, a principios de los 70. Este fue un microprocesador razonable, pero no particularmente bueno manejando puertos de e/s. Para algunas aplicaciones muy específicas GI diseñó un Controlador de Interface Periférico (PIC) entorno a 1975. Fue diseñado para ser muy rápido, además de ser un controlador de e/s para una máquina de 16 bits pero sin necesitar una gran cantidad de funcionalidades, por lo que su lista de instrucciones es pequeña.

El PIC original se diseñó para ser usado con la nueva CPU de 16 bits CP16000. Siendo en general una buena CPU, ésta tenía malas prestaciones de E/S, y el PIC de 8 bits se desarrolló en 1975 para mejorar el rendimiento del sistema quitando peso de E/S a la CPU. El PIC utilizaba microcódigo simple almacenado en ROM para realizar estas tareas; y aunque el término no se usaba por aquel entonces, se trata de un diseño RISC que ejecuta una instrucción cada 4 ciclos del oscilador.

En 1980 aproximadamente, los fabricantes de circuitos integrados iniciaron la difusión de un nuevo circuito para control, medición e instrumentación al que llamaron microcomputador en un sólo chip o de manera más exacta MICROCONTROLADOR.

En 1985, dicha división de microelectrónica de General Instruments se convirtió en una filial y el nuevo propietario canceló casi todos los desarrollos, que para esas fechas la mayoría estaban obsoletos. El PIC, sin embargo, se mejoró con EPROM para conseguir un controlador de canal programable. Hoy en día multitud de PICs vienen con varios periféricos incluidos (módulos de comunicación serie, UARTs, núcleos de control de motores, etc.) y con memoria de programa desde 512 a 32.000 palabras (una palabra corresponde a una instrucción en ensamblador, y puede ser 12, 14 o 16 bits, dependiendo de la familia específica de PICmicro).

3.4.2 GAMAS DE LOS MICROCONTROLADORES PIC

La forma de designación de los PIC en general obedece a la siguiente estructura:

PIC nn LLL xxx	
Donde :	
nn	Es un número propio de la gama del PIC
LLL	Es el código de letras donde la primera indica la tensión de alimentación
xxx	Es el tipo de memoria que utiliza.

TABLA 3.3: DESIGNACIÓN DEL PIC

A continuación se muestra una tabla con las gamas que se les asigna a estos microcontroladores así como su descripción:

GAMA	DESCRIPCIÓN
Microcontroladores de arquitectura cerrada	Cada modelo posee una CPU, memoria, líneas de E/S y recursos auxiliares determinados. No admite ni variaciones ni ampliaciones.
Microcontroladores de arquitectura abierta	Además de disponer de una estructura interna determinada, utilizan sus líneas de E/S para sacar al exterior los buses de datos, dirección y control, con lo que se posibilita la ampliación de la memoria con circuitos integrados externos. Son básicamente microprocesadores.
Gama enana	PIC 12C(F)xxx de 8 pines, consumen 2mA a 5V y trabajan a 4MHz, poseen formato de instrucciones de 12 o 14 bits y su juego de instrucciones puede ser de 33 o 35, pueden disponer de 6 líneas de E/S ya que poseen un oscilador interno RC. Se utilizan para aplicaciones de control personal, sistemas de seguridad y en dispositivos de bajo consumo que gestionen receptores y transmisores de señales.
Gama baja	PIC16C5x de 18 o 28 pines, consumen 2mA y pueden alimentarse a partir de 2.5V, poseen 33 instrucciones de 12 bits, no admiten interrupciones y la pila es de 2 niveles (no admite más de 2 subrutinas). Se utilizan en dispositivos de alta velocidad de la industria de la automoción, control de receptores y transmisores.
Gama media	PIC16xxx de 18 hasta 68 pines, posee 35 instrucciones de 14 bits, habilita las interrupciones y posee una pila de 8 niveles que permite el anidamiento de subrutinas. Se utilizan para cargadores de baterías, gestión del consumo de la energía de alimentación, control remoto, tarjetas codificadas, etc.
Gama alta	PIC17Cxxx de 40 a 44 pines, poseen 58 instrucciones de 16 bits, son de arquitectura abierta, sacan al exterior buses de datos, dirección y control para conectar diversos periféricos, posee un sistema de interrupciones muy potente, un puerto paralelo y serie y mayor capacidad de memoria. Se utilizan para aplicaciones industriales que requieren alta velocidad y cálculos complejos.

TABLA 3.4: GAMAS DEL PIC

3.5. MICROCONTROLADOR PIC 18F452

El PIC 18f452 pertenece al tipo de los procesadores con arquitectura Harvard, es decir, la memoria de datos y de código separadas y arquitectura RISC (reduced instruction set computer). Cuenta con los siguientes elementos: memoria de programa de 32Kb del tipo flash, borrable y programable eléctricamente, 256 bytes de memoria EEPROM para el almacenamiento de datos permanentes en memoria, 1.5 Kb de RAM, en su memoria flash puede escribirse/borrarse 100,000 veces, en su memoria EEPROM puede escribirse/borrarse 1,000,000 veces, en sus dos memorias flash/eeprom puede durar su información hasta 40 años, contiene función de protección de código, contiene la función sleep para mínimo consumo, su tecnología CMOS con muy bajo consumo, su funcionamiento normal 2 [mA] a 5 [V], cuenta con 5 puertos de entrada y salida de datos, 3 interrupciones externas, 4 timers, y comunicaciones externas UART.

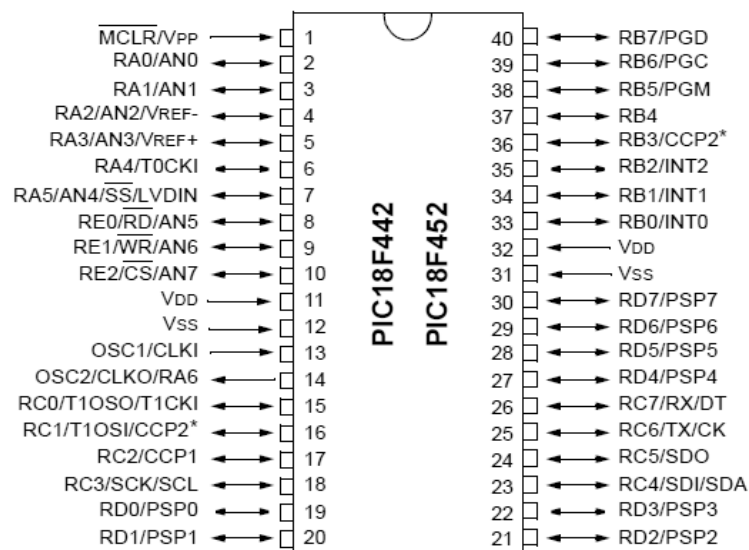


FIGURA 3.5: PIC 18F452 [16]

3.5.1. DEFINICIÓN DE LOS PINES Y SUS FUNCIONES

PIN	NOMBRE	FUNCION
1	MCLR/VPP MCLR VPP	Señal de reset externa. Pin de habilitación para programación ICSP alto voltaje.
2	RA0/AN0 RA0 AN0	Bit cero de entrada/salida del puerto A. O puede configurarse como una entrada analógica (bit cero).
3	RA1/AN1 RA1 AN1	Bit uno de entrada/salida del puerto A. O puede configurarse como una entrada analógica (bit 1).
4	RA2/AN2/V _{ref-} RA2 AN2 V _{ref-}	Bit dos de entrada/salida del puerto A. O puede configurarse como una entrada analógica (bit 2). U ocuparse como entrada de voltaje de referencia Vref negativo.
5	RA3/AN3/V _{ref+} RA3 AN3 V _{ref+}	Bit tres de entrada/salida del puerto A. O puede configurarse como una entrada analógica (bit 3). U ocuparse como entrada de voltaje de referencia (Vref) positivo.
6	RA4/T0CK1 RA4 T0CK1	Bit cuatro de entrada/salida del puerto A. O puede programarse para ser la entrada de reloj para el registro TMR0.
7	RA5/AN4/SS/LVDI N RA5 AN4 SS	Bit cinco de entrada/salida del puerto A. O puede configurarse como una entrada analógica (bit 4). Configurarlo como entrada esclava para el puerto serial. O detectar una caída de voltaje.
8	RE0/RD/AN5 RE0	Bit cero de entrada/salida del puerto E.

	RD AN5	Se puede leer un registro. O configurarse como entrada analógica (bit 5).
9	RE1/WR/AN6 RE1 WR AN6	Bit uno de entrada/salida del puerto E. Escribir en un registro. O configurarse como entrada analógica (bit 6).
10	RE2/CS/AN7 RE2 CS AN7	Bit dos de entrada/salida del puerto E. O configurarse como entrada analógica (bit 7).
11	VDD	Pin de entrada de voltaje positivo para los pines de entrada y salida lógicos
12	VSS	Pin de entrada de voltaje positivo para los pines de entrada y salida lógicos.
13	OSC1/CLKI OSC1 CLKI	Señal 1 de cristal oscilador. O configurarse como entrada de reloj externo.
14	OSC2/CLKO/RA6 OSC2 CLKO RA6	Señal 2 de cristal oscilador. O configurarse en modo RC en donde se genera una señal de salida (CLKOUT), que tiene ¼ de frecuencia de la señal en OSC1 y determina el ciclo de instrucción. Bit seis de entrada/salida del puerto A.
15	RC0/TIOS0/T1CLK RC0 TIOS0 T1CLK	Bit cero de entrada/salida del puerto C. Timer 1 salida del oscilador. Timer 1-timer entrada externa del reloj.
16	RC1/TIOS1/CCP2 RC1 TIOS1 CCP2	Bit uno de entrada/salida del puerto C. Timer 1 entrada del oscilador. Módulo 2 de entrada para captura y comparación de datos, y salida de PWM2.
17	RC2/CCP1 RC2 CCP1	Bit dos de entrada/salida del puerto C. Módulo 1 de entrada para captura(C) y comparación de datos(C), y salida de PWM2 (P1).
18	RC3/SCK/SCL RC3 SCK SCL	Bit tres de entrada/salida del puerto C. Entrada reloj (synchronous serial). Salida para modo SPI Modo I ² C
19	RD0/PSP0 RD0 PSP0	Bit cero de entrada/salida del puerto D. Puerto esclavo de datos.
20	RD1/PSP1 RD1 PSP1	Bit uno de entrada/salida del puerto D. Puerto esclavo de datos.
21	RD2/PSP2 RD2 PSP2	Bit dos de entrada/salida del puerto D. Puerto esclavo de datos.
22	RD3/PSP3 RD3	Bit tres de entrada/salida del puerto D.

	PSP3	Puerto esclavo de datos.
23	RC4/SDI/SDA RC4 SDI SDA	Bit cuatro de entrada/salida del puerto C. Entrada de datos SPI Entrada/salida de datos I ² C
24	RC5/SD0 RC5 SD0	Bit cinco de entrada/salida del puerto C. Salida de datos SPI
25	RC6/TX/CK RC6 TX CK	Bit seis de entrada/salida del puerto C. Transmisión USART (asíncrona). USART reloj síncrono.
26	RC7/RX/DT RC7 RX DT	Bit siete de entrada/salida del puerto C. Recepción USART (asíncrona). USART datos síncronos.
27	RD4/PSP4 RD4 PSP4	Bit cuatro de entrada/salida del puerto D. Puerto esclavo de datos
28	RD5/PSP5 RD5 PSP5	Bit cinco de entrada/salida del puerto D. Puerto esclavo de datos
29	RD6/PSP6 RD6 PSP6	Bit seis de entrada/salida del puerto D. Puerto esclavo de datos
30	RD7/PSP7 RD7 PSP7	Bit siete de entrada/salida del puerto D. Puerto esclavo de datos
31	VSS	Tierra de referencia para los bits de entrada/salida lógicos.
32	VDD	Fuente para los bits de entrada/salida lógicos.
33	RB0/INT0 RB0 INT0	Bit cero de entrada/salida del puerto B. Interrupción externa 0
34	RB1/INT1 RB1 INT1	Bit uno de entrada/salida del puerto B. Interrupción externa 1
35	RB2/INT2 RB2 INT2	Bit dos de entrada/salida del puerto B. Interrupción externa 2
36	RB3/CCP2 RB3 CCP2	Bit tres de entrada/salida del puerto B. Módulo 2 de entrada para captura y comparación de datos, y salida de PWM2.
37	RB4	Bit cuatro de entrada/salida del puerto B.
38	RB5	Bit cinco de entrada/salida del puerto B.
39	RB6/PGC RB6 PGC	Bit seis de entrada/salida del puerto B. Programación ISP del reloj
40	RB7/PGD RB7 PGD	Bit siete de entrada/salida del puerto B. Programación ISP de datos

TABLA 3.5: DEFINICIÓN DE LOS PINES

3.5.2. PUERTOS DIGITALES DE ENTRADA Y SALIDA

El PIC cuenta con 5 puertos digitales, el puerto A cuenta con 7 bits disponibles, el puerto B cuenta con 8 bits, el puerto C cuenta con 8 bits, el puerto D cuenta con 8 bits, el puerto E cuenta con solo 3 bits, todos estos puertos son bidireccionales, con esto se quiere decir que pueden programarse como entradas o salidas. Por ejemplo una salida puede ser un led prendido y una entrada cualquier switch.

3.5.3. INTERRUPCIONES DEL SISTEMA

El PIC tiene fuentes de interrupción múltiples, así como prioridad de interrupción, esto quiere decir que a dichas fuentes de interrupción se les asigna un nivel de prioridad ya sea alto o bajo, los cuales se encuentran en el vector de interrupción 000008h para el nivel alto, y en el vector de interrupción 000018h para el nivel bajo. A grandes rasgos las tareas prioritarias de la interrupción a nivel alto eliminarán cualquier interrupción de prioridad baja que pueda estar en curso.

Cuenta con diversas fuentes de interrupción asociadas a la ocurrencia de algunos eventos como:

- Las interrupciones externas en los pines RB0/INT0, RB1/INT1, RB2/INT2
- El overflow en el temporizador del chip
- Cualquier cambio de nivel en los pines RB4-RB7
- Cuando se ha completado la escritura de un dato en la EEPROM

3.5.4. PUERTO DE COMUNICACIÓN USART

El módulo USART es conocido como el receptor transmisor sincrónico asincrónico universal, es uno de los dos módulos de entrada y salida, es la interfaz de las comunicaciones seriales.

Configurable en tres modos de trabajo:

1. Asíncrono (full dúplex): Recepción y transmisión independientes compartiendo generador de relación de baudios (BRG), TX: pin de transmisión (salida), RX: pin de recepción (entrada)
2. Síncrono modo Maestro (semi dúplex): CK: reloj generado por el PIC (salida), DT: datos entrantes (recepción) o salientes (transmisión)
3. Síncrono modo Esclavo (semi dúplex): CK: reloj entrante, DT: datos entrantes (recepción) o salientes (transmisión).

Su configuración de pines: RC6 se usa como transmisor y RC7 como receptor (RC6/TX/CK and RC7/RX/DT).

3.5.4.1 TRANSFERENCIA SÍNCRONA

El dispositivo Maestro: es el que genera la señal de reloj, es el que tiene capacidad de iniciar o finalizar una transferencia.

El dispositivo Esclavo: recibe la señal de reloj, no tiene capacidad para iniciar una transferencia de información.

Es posible una transmisión continua de bits, no hay límite en tamaño de datos.

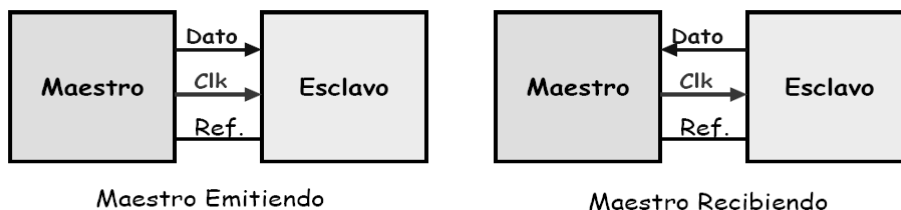


FIGURA 3.6: TRANSFERENCIA SÍNCRONA [17]

3.5.4.2 TRANSFERENCIA ASÍNCRONA

Se emplean relojes de igual frecuencia (se acuerda y configura la velocidad de transmisión) pero es necesario que estén en fase (sincronizados).

Cada paquete de bits de tamaño fijo se “enmarca” con bits de arranque y de parada que sirven para sincronizar los relojes del emisor y del receptor.

La línea de datos inactiva a “1”, si se desea enviar un dato se manda un bit de arranque que sitúa a “0” la línea durante el tiempo correspondiente a un bit.

Al finalizar el envío de un dato, la línea se sitúa a “1” al menos durante el tiempo de un bit: bit de parada.



FIGURA 3.7: TRANSFERENCIA ASÍNCRONA [17]

3.5.4.3 MÓDULO SCI EN TRANSMISIÓN ASÍNCRONA

Es la conexión más adecuada para la comunicación con un equipo alejado, Los niveles lógicos de las señales se corresponden con los niveles eléctricos de alimentación del microcontrolador, hay varias normas de transmisión serie asíncrona (RS232, RS485, RS422) que emplean niveles de tensión más inmunes al ruido (RS232) o que emplean tensiones diferenciales (RS485, RS422) y que son más apropiadas para distancias largas entre dispositivos, para implementar estas transmisiones, sería necesario la adaptación de niveles eléctricos mediante los correspondientes circuitos integrados de adaptación (drivers o transceivers). Esta transmisión puede ser unidireccional o bidireccional y simultánea

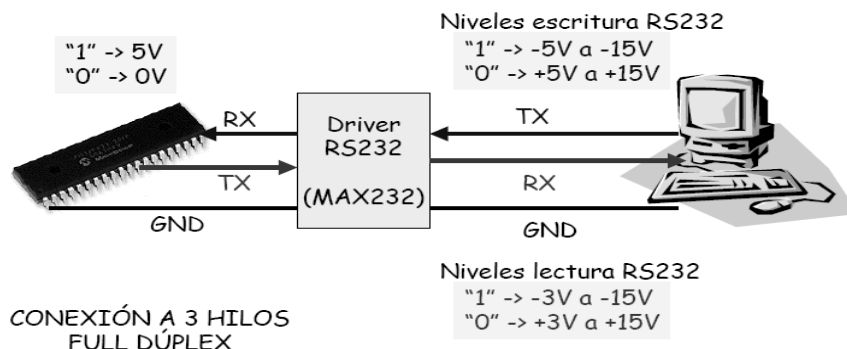


FIGURA 3.8: EJEMPLO DE COMUNICACIÓN PIC - PC MEDIANTE PUERTO SERIE (RS232) [18]

3.5.5. MÓDULO CCP (PWM/CAPTURA/COMPARACION)

Este tipo de PIC puede tener hasta 2 módulos CCP, cada módulo CCP tiene un registro de 16 bits que se puede utilizar de 3 formas distintas:

1. Como registro de 16 bits para captura de tiempo al producirse un evento.
2. Como registro de 16 bits para compararlo con el valor de cuenta del temporizador TMR1, pudiendo provocar un evento cuando se alcanza el valor contenido en este registro.
3. Como registro de 10 bits del ciclo de trabajo de una señal PWM generada por el microcontrolador.

Los 2 módulos CCP disponibles se comportan casi idénticamente, salvo el caso del funcionamiento por disparo de evento especial (special event trigger) que tiene una pequeña diferencia si se trata del módulo CCP1 ó del módulo CCP2. Tras un reset, el módulo CCP está apagado (al forzar los bits de configuración al valor 0).

La siguiente tabla muestra las posibles interacciones entre los módulos CCP, donde CCPx es uno de los módulos y CCPy es el otro:

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	Same TMR1 time-base.
Capture	Compare	The compare should be configured for the special event trigger, which clears TMR1.
Compare	Compare	The compare(s) should be configured for the special event trigger, which clears TMR1.
PWM	PWM	The PWMs will have the same frequency, and update rate (TMR2 interrupt).
PWM	Capture	None
PWM	Compare	None

TABLA 3.6: MÓDULO CCP [16]

3.5.6. TIMERS

Un temporizador, en general, es un dispositivo que marca o indica el transcurso de un tiempo determinado. Este PIC tiene 4 módulos temporizadores denominados TIMER0 (TMR0), TIMER1 (TMR1), TIMER2 (TMR2) y TIMER3 (TMR3). Los módulos temporizadores en los microcontroladores PIC se emplean para contabilizar intervalos de tiempo o para contar flancos que aparecen en pines externos del micro. Cuando trabajan como temporizadores, utilizan como patrón de cuenta un reloj que se genera a partir del oscilador del microcontrolador, Cada módulo puede generar una interrupción para indicar que algún evento ha ocurrido (que se ha sobrepasado el valor máximo de cuenta de un temporizador “overflow” o que se ha alcanzado un valor dado).

3.5.7. MODULO CONVERTIDOR ANALÓGICO/DIGITAL

El PIC cuenta con 8 canales de conversión, cinco pines en el puerto A y tres pines en el puerto E. Convierte la señal analógica en un número digital de 10 bits, puede usar tensiones de referencia VREF+ y VREF- seleccionables por software como Pueden ser VDD y VSS o las tensiones aplicadas en los pines RA3 / RA2.

También puede seguir funcionando cuando el PIC está en modo SLEEP ya que dispone de un oscilador RC interno propio.

A continuación se presenta el diagrama de bloques del PIC18F452:

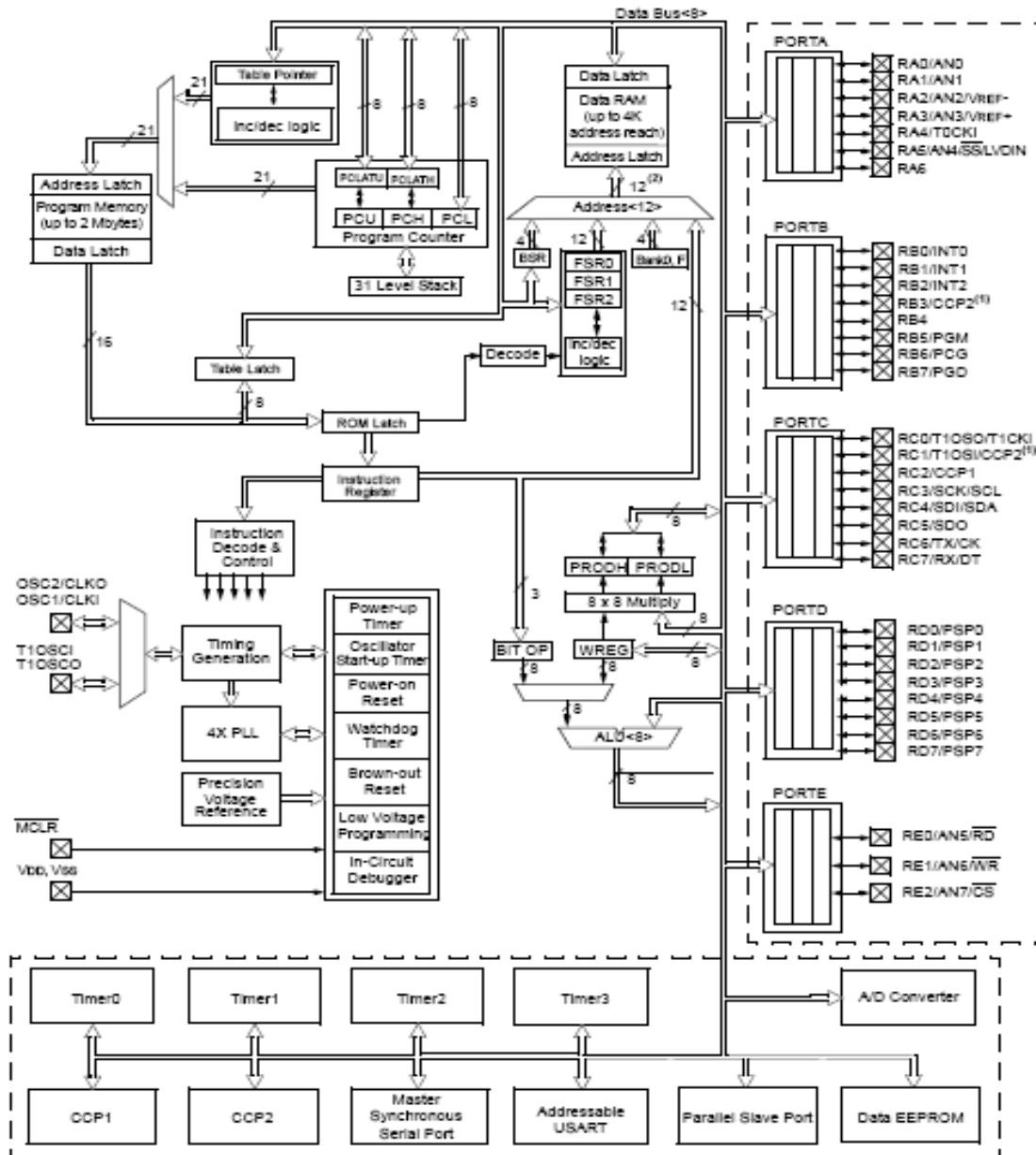


FIGURA 3.9: DIAGRAMA DE BLOQUES [16]

3.6. COMUNICACIÓN SERIE

La comunicación serial, como su nombre lo indica, realiza la transferencia de información enviando o recibiendo datos descompuestos en bits, los cuales viajan secuencialmente uno tras otro.

Las comunicaciones serie se utilizan para enviar datos a través de largas distancias, ya que las comunicaciones en paralelo exigen demasiado cableado para ser operativas. Los elementos básicos referidos a este tipo de comunicaciones son dos, el hardware que hace referencia a los niveles de tensión, a la configuración y tipo de conectores y el segundo, el software con el que se controla la información binaria que se quiere transferir.

Las características más importantes de la comunicación serial son la velocidad de transmisión, los bits de datos, los bits de parada y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

CARACTERÍSTICA	DESCRIPCIÓN
Velocidad de transmisión (baud rate)	Indica el número de bits por segundo que se transfieren y se mide en baudios, por ejemplo 9600 baudios representa 9600 bits por segundo.
Bits de inicio	Usado para indicar el inicio de la comunicación de un solo paquete.
Bits de datos	Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits.
Paridad	Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. La opción de no usar paridad alguna también está disponible.
Tipos de paridad	Paridad par (even) expresado con la letra "E". Paridad impar (odd) expresado con la letra "O". Paridad marca (mark) expresado con la letra "M". Paridad espacio (space) expresado con la letra "S". Sin paridad (no se añade ningún bit de paridad) expresado con la letra "N" Los sistemas de paridad par e impar; ambos métodos cuentan el número de unos contenidos en los bits de datos y añade un uno o un cero según el resultado. Por su parte la paridad Mark indica que se incluirá siempre una marca (bit de valor "1") como bit de paridad, mientras que la paridad Space añade siempre un espacio ("0"). Evidentemente estos dos últimos sistemas no aportan absolutamente ninguna información, por lo que son usados muy raramente.
Bits de parada	Usado para indicar el fin de la comunicación de un solo paquete, los valores típicos son 1, 1.5 o 2 bits.

TABLA 3.7: COMUNICACIÓN SERIE

Generalmente el protocolo utilizado es 8N1 el cual significa. 8 bits de datos, sin paridad y 1 bit de stop.

3.6.1 PROTOCOLO TRANSMISIÓN RS232

Sólo es un nombre para un estándar que se ha propagado de generación en generación de computadoras. Por medio de este protocolo se estandarizan las velocidades de transferencia de datos, la forma de control que utiliza dicha transferencia, los niveles de voltajes utilizados, el tipo de cable permitido, las distancias entre equipos, los conectores, etc.

Además de las líneas de transmisión (Tx) y recepción (Rx), las comunicaciones seriales poseen otras líneas de control de flujo, donde su uso es opcional dependiendo del dispositivo a conectar. A nivel de software, la configuración principal que se debe dar a una conexión a través de puertos seriales. RS-232 es básicamente la selección de la velocidad en baudios, la verificación de datos o paridad, los bits de parada luego de cada dato y la cantidad de bits por dato, que se utiliza para cada símbolo o carácter enviado.

La norma RS-232 fue definida para conectar una computadora a un modem. Además de transmitirse los datos de una forma serie asíncrona son necesarias una serie de señales adicionales, que se definen en la norma. Las tensiones empleadas están comprendidas entre +15/-15 volts.

Puerta serial full dúplex para comunicación punto a punto a una distancia no superior a 30 metros. Desde 3 hilos hasta 19 hilos.

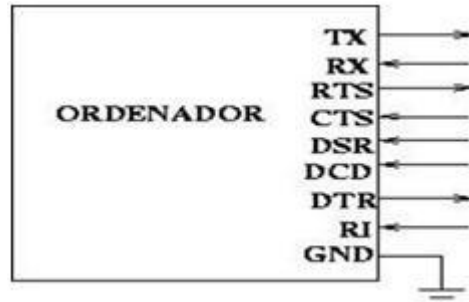


FIGURA 3.10: PROTOCOLO RS232

Este protocolo está disponible en los puertos seriales de la mayoría de las computadoras personales (PC). El primer puerto serial denominado comúnmente COM1 tiene asignada la interrupción IRQ4 y sus registros empiezan en la dirección de la memoria 3F8, y de ahí en adelante hasta la 3FE. Para las máquinas que tienen un segundo puerto serial este se denomina COM2, tiene asignada la interrupción IRQ3 y sus registros se alojan en las direcciones 2F8 hasta la 2FE. Los puertos denominados COM3 y COM4 a pesar de que se mapean en un espacio diferente de los puertos anteriores, comparten las interrupciones, COM1 con COM3 y COM2 con COM4, por esto es muy difícil utilizar los cuatro cuando se trata de hacerlos funcionar mediante interrupciones.

Cada pin puede ser de entrada o de salida, teniendo una función específica cada uno de ellos. Las más importantes son:

PIN	FUNCIÓN
TXD	Transmitir datos
RXD	Recibir datos
DTR	Terminal de datos listo
DSR	Equipo de datos listo
RTS	Solicitud de envío
CTS	Libre para envío
DCD	Detección de portadora

TABLA 3.8: PINES RS232

Las señales TXD, DTR y RTS son de salida, mientras que RXD, DSR, CTS y DCD son de entrada. La masa de referencia para todas las señales es SG (Tierra de Señal). Finalmente, existen otras señales como RI (Indicador de Llamada).

NÚMERO DE PIN DB-25	NÚMERO DE PIN DB-9	SEÑAL	DESCRIPCIÓN	E/S
1	1		Masa chasis	-
2	3	TxD	Transmit Data	S
3	2	RxD	Receive Data	E
4	7	RTS	Request To Send	S
5	8	CTS	Clear To Send	E
6	6	DSR	Data Set Ready	E
7	5	SG	Signal Ground	-
8	1	CD/DCD	(Data) Carrier Detect	E
15	-	TxC(*)	Transmit Clock	S
17	-	RxC(*)	Receive Clock	E
20	4	DTR	Data Terminal Ready	S
22	9	RI	Ring Indicator	E
24	-	RTxC(*)	Transmit/Receive Clock	S

(*) = Normalmente no conectados en el DB-25

TABLA 3.9: DB25 Y DB9 (PINES)

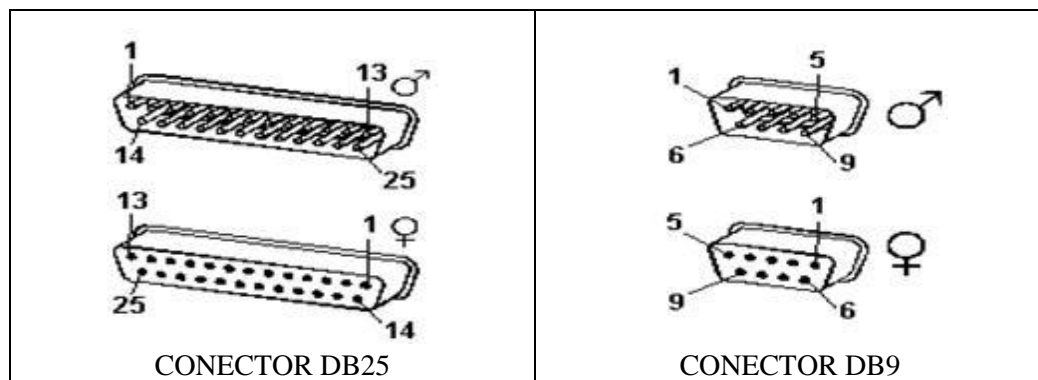


FIGURA 3.11: CONECTORES DB25 Y DB9

3.6.1.1 EL PUERTO SERIE EN EL PC

Mediante los puertos de E/S se pueden intercambiar datos, mientras que las IRQ producen una interrupción para indicar a la CPU que ha ocurrido un evento (por ejemplo, que ha llegado un dato, o que ha cambiado el estado de algunas señales de entrada). La CPU debe responder a estas interrupciones lo más rápido posible, para que dé tiempo a recoger el dato antes de que el siguiente lo sobrescriba. Sin embargo, las UART 16550A incluyen unos buffers de tipo FIFO, dos de 16 bytes (para recepción y transmisión), donde se pueden guardar varios datos antes de que la CPU los recoja. Esto también disminuye el número de interrupciones por segundo generadas por el puerto serie.

Una vez que ha comenzado la transmisión de un dato, los bits tienen que llegar uno detrás de otro a una velocidad constante y en determinados instantes de tiempo. Por eso se dice que el RS-232 es asíncrono por carácter y síncrono por bit. Los pines que portan los datos son RXD y TXD. Las demás se encargan de otros trabajos: DTR indica que el ordenador está encendido, DSR que el aparato conectado a dicho puerto está encendido, RTS que el ordenador puede recibir datos (porque no está ocupado), CTS que el aparato conectado puede recibir datos, y DCD detecta que existe una comunicación, presencia de datos.

Tanto el aparato a conectar como el ordenador (o el programa terminal) tienen que usar el mismo protocolo serie para comunicarse entre sí. Puesto que el estándar RS-232 no permite indicar

en qué modo se está trabajando, es el usuario quien tiene que decidirlo y configurar ambas partes. Los parámetros que hay que configurar son: protocolo serie (8N1), velocidad del puerto serie, y protocolo de control de flujo. Este ultimo puede ser por hardware o bien por software (XON/XOFF, el cual no es muy recomendable ya que no se pueden realizar transferencias binarias). La velocidad del puerto serie no tiene por qué ser la misma que la de transmisión de los datos, de hecho debe ser superior. Por ejemplo, para transmisiones de 1200 baudios es recomendable usar 9600, y para 9600 baudios se pueden usar 38400 (o 19200).

3.6.2 CIRCUITO MAX232

La electrónica ha cambiado un poco. Antes se utilizaban aparatos de relativamente alto poder y alto voltaje, -12V a 12V. Los aparatos modernos no operan a voltajes tan altos. De hecho, la mayoría trabajan entre 0V y 5V. ¿Entonces como hacemos trabajar el RS232 con nuestro micro de 5V con voltajes de +/-12V? Este problema fue solucionado por los fabricantes de integrados del mundo. Fabricaron un IC que es genéricamente conocido como el MAX232.

A grandes rasgos el MAX232 es el circuito integrado estándar para convertir señales TTL/CMOS a señales RS232. Las señales en RS232 tienen 1's y 0's estos son +12V y -12V respectivamente. Las señales de salida del PIC son entre 0 y 5V.

Lo que el MAX232 hace es poner 12V en el pin T1OUT cuando se alimenta con 5V el pin T1IN. De esta forma se pueden pasar datos hacia tu computador. Si presionas una tecla en el hyperterminal una señal será enviada a través del cable hasta el R1IN del MAX232 donde la señal de 12V proveniente del PC es convertida a 0-5V. Esta señal que sale por el R1OUT es perfectamente manejable por el PIC.

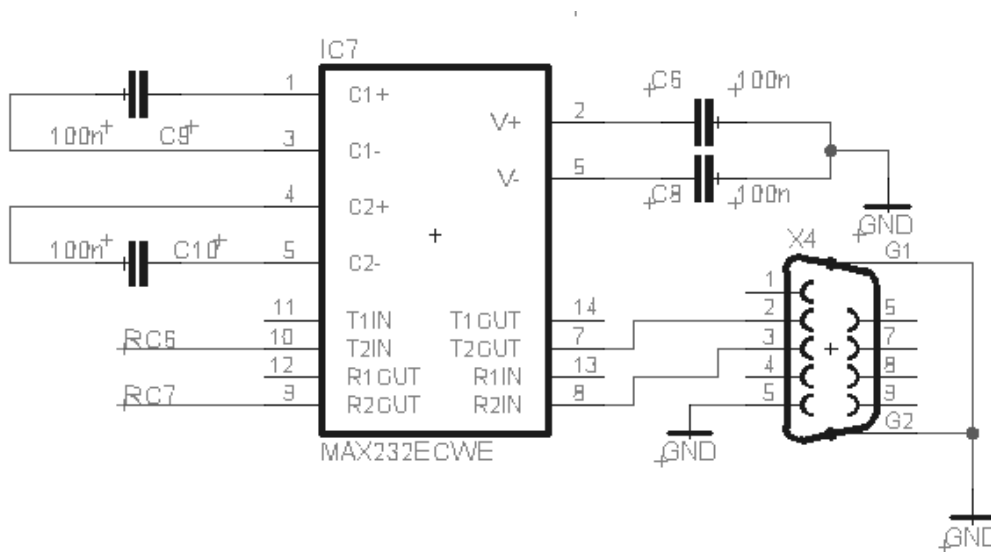


FIGURA 3.12: CIRCUITO MAX 232

3.7. TEORÍA DE MOTORES

Un motor es un dispositivo capaz de transformar cualquier tipo de energía como la eléctrica, combustibles de fósiles, etc., en energía mecánica capaz de realizar un trabajo.

Existen diversos tipos, siendo los más comunes:

CLASIFICACIÓN	DESCRIPCIÓN
Motores térmicos	Se ocupan cuando el trabajo se obtiene a partir de energía térmica.
Motores de combustión interna	Son motores térmicos en los cuales se produce una combustión del fluido motor, transformando su energía química en energía térmica, a partir de la cual se obtiene energía mecánica. El fluido motor antes de iniciar la combustión es una mezcla de un comburente (como el aire) y un combustible, como los derivados del petróleo, los del gas natural o los biocombustibles.
Motores de combustión externa	Son motores térmicos en los cuales se produce una combustión en un fluido distinto al fluido motor. El fluido motor alcanza un estado térmico de mayor energía mediante la transmisión de energía a través de una pared.
Motores eléctricos	<p>El motor eléctrico permite la transformación de energía eléctrica en energía mecánica, esto se logra mediante la rotación de un campo magnético alrededor de una espira o bobinado que toma diferentes formas.</p> <p>Al pasar la corriente eléctrica por la bobina ésta se comporta como un imán cuyos polos se rechazan o atraen con el imán que se encuentra en la parte inferior; al dar media vuelta el paso de corriente se interrumpe y la bobina deja de comportarse como imán pero por inercia se sigue moviendo hasta que da otra media vuelta y la corriente pasa nuevamente repitiéndose el ciclo haciendo que el motor rote constantemente.</p>

TABLA 3.10: CLASIFICACIÓN DE MOTORES

3.7.1 MOTOR DE CORRIENTE DIRECTA O CONTINUA

Los Motores de Corriente Directa (CD) o Corriente Continua (CC) se utilizan en casos en los que es importante el poder regular continuamente la velocidad del motor, además, se utilizan en aquellos casos en los que es imprescindible utilizar corriente directa, como es el caso de motores accionados por pilas o baterías. Este tipo de motores debe de tener en el rotor y el estator el mismo número de polos y el mismo número de carbones.

Como su nombre lo indica, un motor eléctrico de corriente continua, funciona con corriente continua. En estos motores, el inductor es el estator y el inducido es el rotor. Fueron los primeros en utilizarse en vehículos eléctricos por sus buenas características en tracción y por la simplicidad de los sistemas de control de la electricidad desde las baterías. Presentan desventajas en cuanto al mantenimiento de algunas de sus piezas (escobillas y colectores) y a que deben ser motores grandes si se buscan potencias elevadas, pues su estructura (y en concreto el rozamiento entre piezas) condiciona el límite de velocidad de rotación máxima.

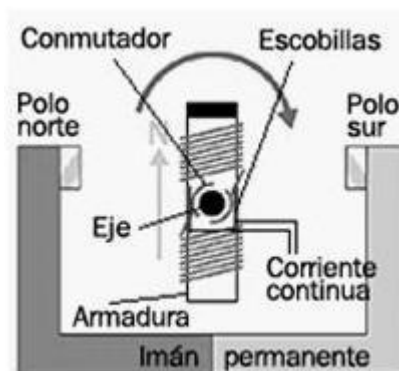


FIGURA 3.13: MOTOR DE CORRIENTE DIRECTA [19]

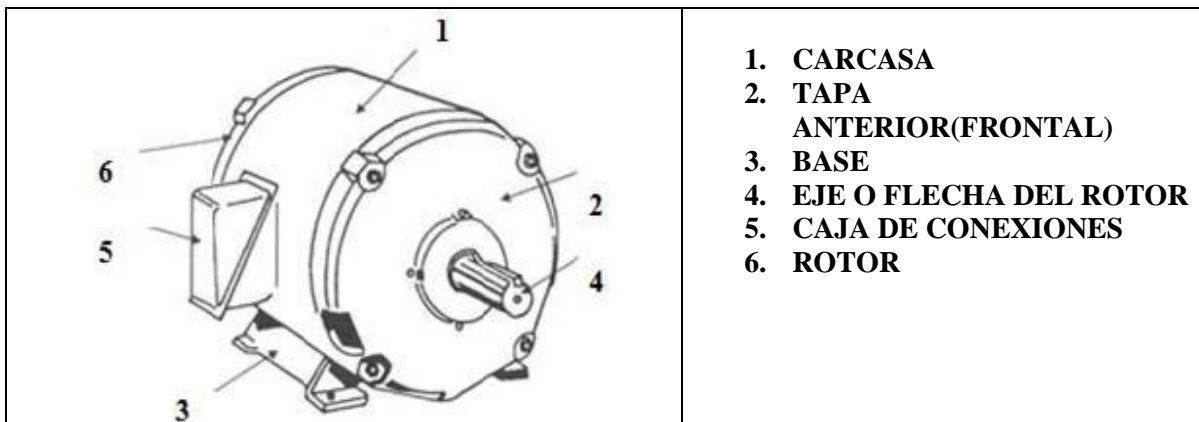


FIGURA 3.14: PARTES DE UN MOTOR DE CORRIENTE DIRECTA [19]

Internamente está conformado por:

- Inductor.
- Polo inductor.
- Inducido, al que va arrollado un conductor de cobre formando el arrollamiento.
- Núcleos polares, va arrollando, en forma de hélice al arrollamiento de excitación.
- Cada núcleo de los polos de conmutación lleva un arrollamiento de conmutación.
- Conmutador o colector, que está constituido por varias láminas aisladas entre sí.

El arrollamiento del inducido está unido por conductores con las laminas del colector. Sobre la superficie del colector rozan unos contactos a presión mediante unos muelles. Dichas piezas de contacto se llaman escobillas. El espacio libre entre las piezas polares y el inducido se llama entrehierro.

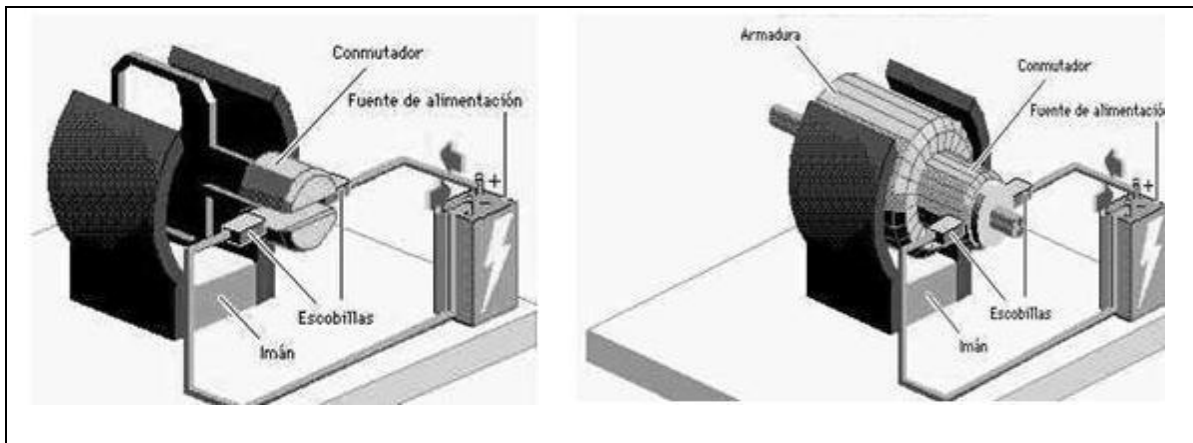


FIGURA 3.15: ESQUEMA INTERNO DE UN MOTOR DE CORRIENTE CONTINUA [19]

La principal característica del motor de corriente continua es la posibilidad de regular la velocidad desde vacío a plena carga.

Este tipo de motor se compone principalmente de dos partes, un estator que da soporte mecánico al aparato y tiene un hueco en el centro generalmente de forma cilíndrica. En el estator además se encuentran los polos, que pueden ser de imanes permanentes o devanados con hilo de cobre sobre núcleo de hierro. El rotor es generalmente de forma cilíndrica, también devanado y con núcleo, al que llega la corriente mediante dos escobillas.

3.7.1.1 PRINCIPIO DE FUNCIONAMIENTO

Según la Ley de Lorentz, cuando un conductor por el que pasa una corriente eléctrica se sumerge en un campo magnético, el conductor sufre una fuerza perpendicular al plano formado por el campo magnético y la corriente, siguiendo la regla de la mano derecha, con módulo

$$F = B \cdot l \cdot I$$

- **F**: Fuerza en newtons
- **I**: Intensidad que recorre el conductor en amperios
- **l**: Longitud del conductor en metros lineales
- **B**: Inducción en teslas

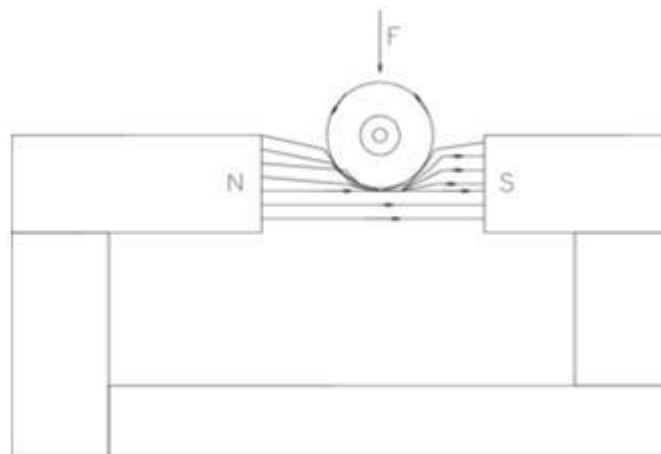


FIGURA 3.16: SEGUNDA LEY DE LORENTZ 1 [20]

Si el conductor está colocado fuera del eje de giro del rotor, la fuerza producirá un momento que hará que el rotor gire.

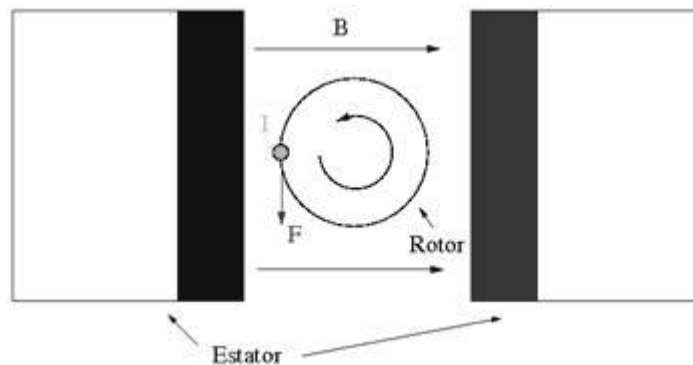


FIGURA 3.17: SEGUNDA LEY DE LORENTZ-2 [20]

El rotor no solo tiene un conductor, sino varios repartidos por la periferia. A medida que gira, la corriente se activa en el conductor apropiado.

Normalmente se aplica una corriente con sentido contrario en el extremo opuesto del rotor, para compensar la fuerza neta y aumentar el momento.

PRINCIPIALES ELEMENTOS DE UN MOTOR CC	DESCRIPCIÓN
Fuerza contra electromotriz inducida en un motor	Es la tensión que se crea en los conductores de un motor como consecuencia del corte de las líneas de fuerza, es el efecto generador. La polaridad de la tensión en los generadores es inversa a la aplicada en bornes del motor. Las fuertes puntas de corriente de un motor en el arranque son debidas a que con máquina parada no hay fuerza contra electromotriz y el bobinado se comporta como una resistencia pura del circuito.
Número de escobillas	Las escobillas deben poner en cortocircuito todas las bobinas situadas en la zona neutra. Si la máquina tiene dos polos, tenemos también dos zonas neutras. En consecuencia, el número total de escobillas ha de ser igual al número de polos de la máquina. En cuanto a su posición, será coincidente con las líneas neutras de los polos.
Sentido de giro	El sentido de giro de un motor de corriente continua depende del sentido relativo de las corrientes circulantes por los devanados inductor e inducido. La inversión del sentido de giro del motor de corriente continua se consigue invirtiendo el sentido del campo magnético o de la corriente del inducido. Si se permuta la polaridad en ambos bobinados, el eje del motor gira en el mismo sentido. Los cambios de polaridad de los bobinados, tanto en el inductor como en el inducido se realizarán en la caja de bornes de la máquina, y además el ciclo combinado producido por el rotor produce la fmm (fuerza magnetomotriz).

TABLA 3.11: ELEMENTOS DE UN MOTOR CORRIENTE CONTINUA

3.7.2 PUENTE H

Un Puente H o Puente en H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avance y retroceso. A grandes rasgos es una interfaz que se considera básicamente como un sistema de conmutación controlado por dos señales digitales de baja potencia, generalmente es utilizado para el control de giro de los motores, cuando el sistema detecta un 1 lógico en una de sus dos entradas de control y un cero en la otra, este conecta el motor a la fuente de alimentación con determinada polaridad si la señal de control que estaba en 1 pasa a cero y la de cero a uno el PUENTE H conecta la fuente al motor con la polaridad invertida facilitando así el giro en sentido contrario.

El término "puente H" proviene de la típica representación gráfica del circuito. Un puente H se construye con 4 interruptores (mecánicos o mediante transistores), como el que se muestra en la figura 3.18.

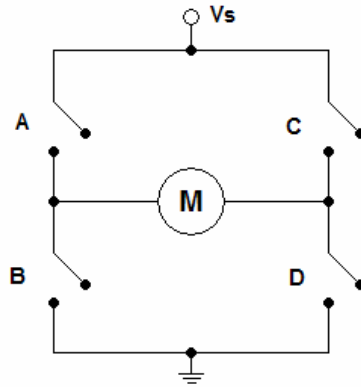


FIGURA 3.18: ESQUEMA PUENTE H

Por ejemplo si vemos la figura 3.19-A, cuando los interruptores A y D están cerrados (a su vez B y C abiertos) se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores A y D (y cerrando B y C), el voltaje se invierte, permitiendo el giro en sentido inverso del motor como en la figura 3.19-B.

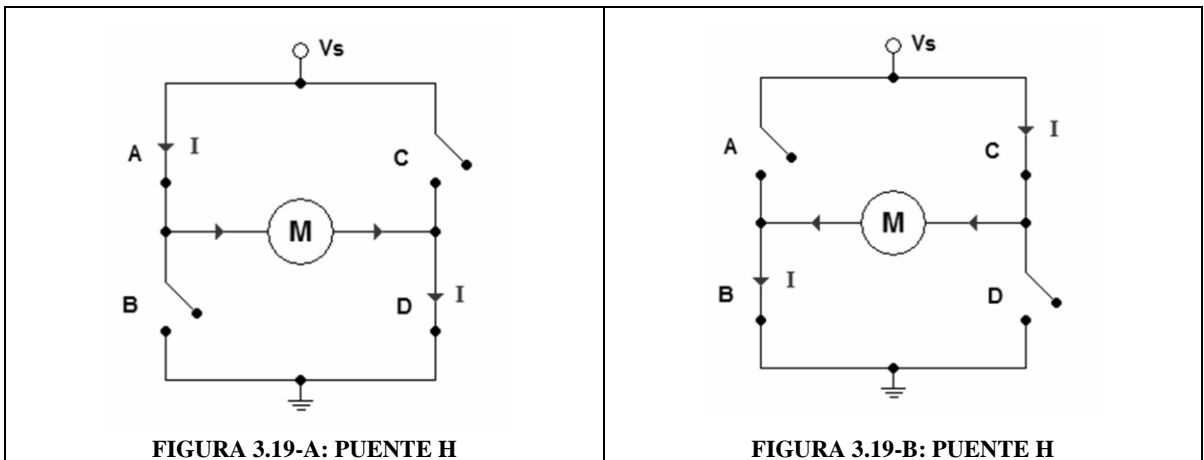


FIGURA 3.19-A: PUENTE H

FIGURA 3.19-B: PUENTE H

Cabe mencionar, que si los interruptores A y C nunca podrán estar cerrados al mismo tiempo, porque esto cortocircuitaría la fuente de tensión. Lo mismo sucede con B y D.

Como hemos dicho el puente H se usa para invertir el giro de un motor, pero también puede usarse para frenarlo de manera brusca o en seco, esto se logra al hacer un corto entre los bornes del motor, o incluso puede usarse para permitir que el motor frene bajo su propia inercia, cuando desconectamos el motor de la fuente que lo alimenta.

A continuación por medio de una tabla se muestra un ejemplo del funcionamiento del puente H de acuerdo al esquema mostrado en la tabla 3.12

A	B	C	D	Acción
1	0	0	1	Avanza
0	1	1	0	Retrocede
0	0	0	0	Se detiene bajo su propia inercia
0	1	0	1	Frena

TABLA 3.12: TABLA DE VERDAD DEL PUENTE H.

3.7.2.1 CIRCUITO INTEGRADO L298.

Este integrado contiene dos puentes en H y por tanto con él seremos capaces de manejar hasta dos motores. El motor, gracias a este integrado, podrá ser alimentado a una tensión distinta de los 5V que emite el PIC. El L298 soporta hasta 2 Amperios de corriente, con lo que debemos tener mucho cuidado al elegir los motores, porque puede que consuman más de estos 2 Amperios.

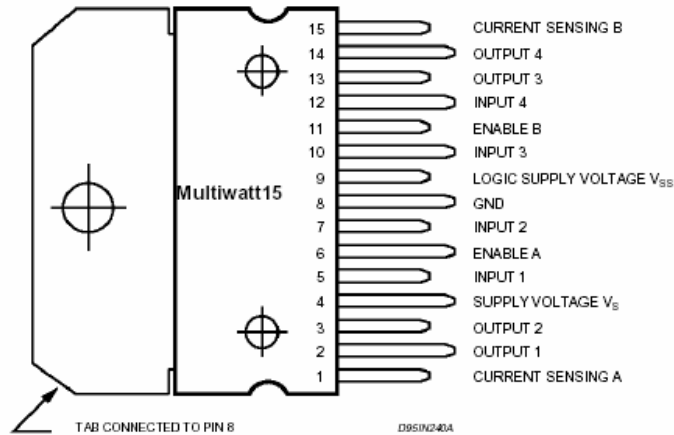


FIGURA 3.20: CIRCUITO INTEGRADO. L298 [21]

Desde el micro tendremos que enviar tres señales a cada motor. Dos se enviarán al puente en H para regir el sentido de giro, y la última será el tren de pulsos (señal PWM).

El esquemático de cada L298 se muestra en la figura 3.21. Cómo se ve en dicho esquemático el L298 posee dos puente H, cada uno con tres señales de control: In1, In2 y EnA (señal de habilitación). De esta forma según la tabla 3.12 se tiene el funcionamiento básico para los motores.

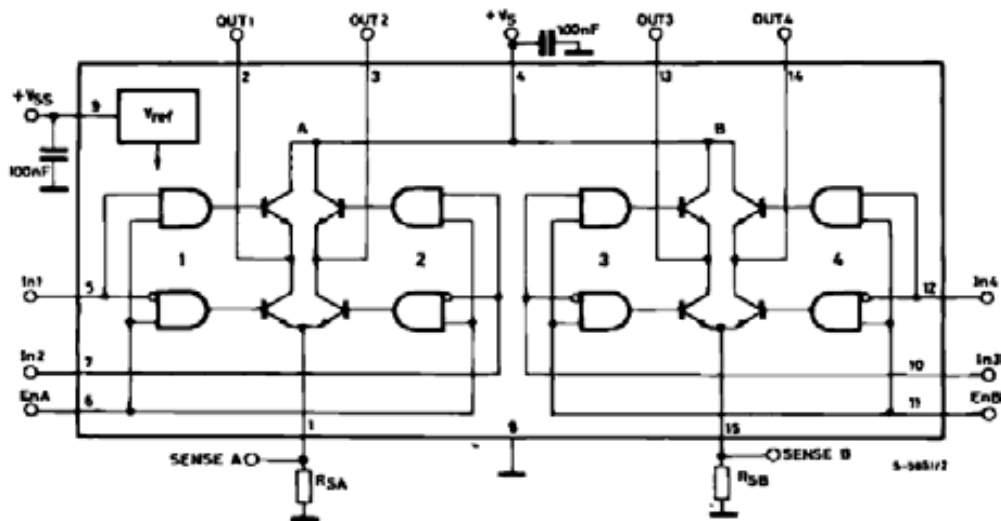


FIGURA 3.21: ESQUEMÁTICO L298 [21]

3.7.2.2 SISTEMAS DE PROTECCIÓN

En la figura 3.22 se aprecia cómo se debe conectar al motor el L298. Nótese que debe haber 4 diodos de protección para el motor. Los diodos son la protección de los transistores contra los picos de tensión que puedan generar el motor. Tienen un funcionamiento "Activo" solo ante los pulsos de tensión inversa que genera el motor al momento de apagado del mismo, colocarlos es una práctica habitual para el manejo de cargas inductivas (Motores, solenoides, transformadores, Etc).

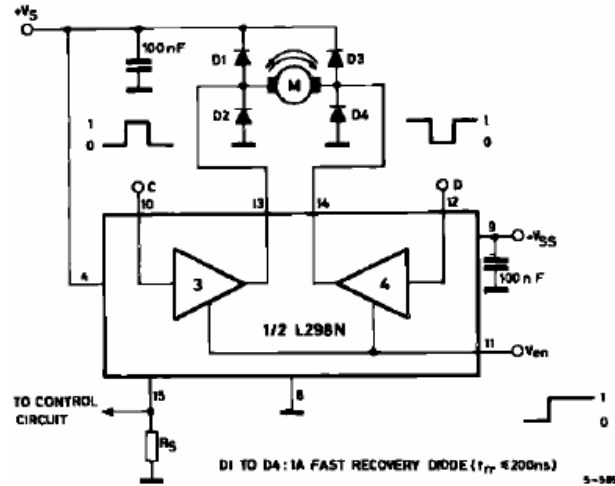


FIGURA 3.22: SISTEMAS DE PROTECCIÓN EN EL PUENTE H [21]

3.7.2.2.1 . DIODO RECTIFICADOR COMO ELEMENTO DE PROTECCIÓN

La desactivación de un motor provoca una corriente de descarga de la bobina en sentido inverso que pone en peligro el elemento electrónico utilizado para su activación. Un diodo polarizado inversamente cortocircuita dicha corriente y elimina el problema. En la siguiente tabla se muestra los dos tipos de polarización de un diodo.

Polarización	Circuito	Características
Directa: el ánodo se conecta al positivo de la batería y el cátodo al negativo		El diodo conduce con una caída de tensión de 0,6 a 0,7V. El valor de la resistencia interna sería muy bajo. Se comporta como un interruptor cerrado
Inversa: el ánodo se conecta al negativo y el cátodo al positivo de la batería		El diodo no conduce y toda la tensión de la pila cae sobre él. Puede existir una corriente de fuga del orden de μA . El valor de la resistencia interna sería muy alto. Se comporta como un interruptor abierto.

TABLA 3.13: POLARIZACIÓN DE UN DIODO

3.7.3 SERVOMOTORES

Un servo es un motor de continua dotado de su propia electrónica. Tiene su propia etapa de potencia, por lo que no es necesario conectarlo a una etapa de potencia como un driver L298. Pero, como un motor que es, necesita recibir un tren de pulsos.

En un servomotor el tren de pulsos no rige la velocidad del servo, sino su posición. Según el ancho del pulso, girará un ángulo entre 0 y 180 grados, aunque depende del servo en cuestión:

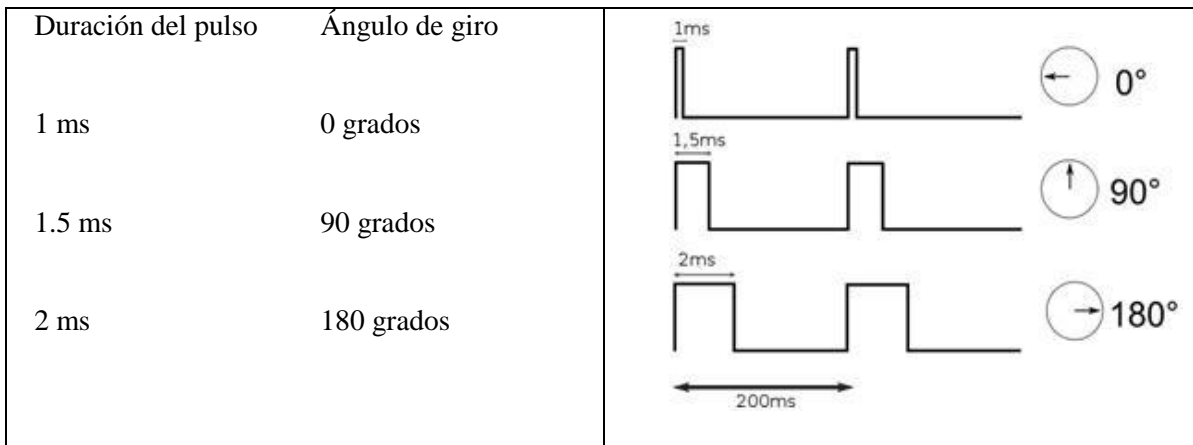


FIGURA 3.23: PULSO Y ÁNGULO DE UN SERVOMOTOR

Así pues, el servo posee tres cables: 1. Referencia de tensión (masa o tierra) que suele ser de color negro, 2. Alimentación (generalmente 6V) de color rojo o naranja y 3. Cable de color blanco o amarillo por el que se envía el tren de pulsos. El período total del pulso (tiempo entre el inicio de un pulso y el siguiente) es válido si se encuentra entre los 10 y los 30 ms, por lo que dependiendo del reloj del PIC quizá se pueda (o no) enviar el PWM directamente del microcontrolador.



FIGURA 3.24: SERVOMOTOR

Conviene resaltar que el servo es un sistema realimentado: Es decir, conoce su posición y la que debe ocupar, para tratar de reducir el error entre ellas. Si el servo recibe un tren de pulsos, permanece en el ángulo que ese tren de pulsos le fuerza a ocupar. Si una fuerza externa afecta al servo, éste la contrarresta para permanecer en la posición que debe ocupar. Para ello, necesita consumir más potencia, y aumenta la intensidad y el consumo.

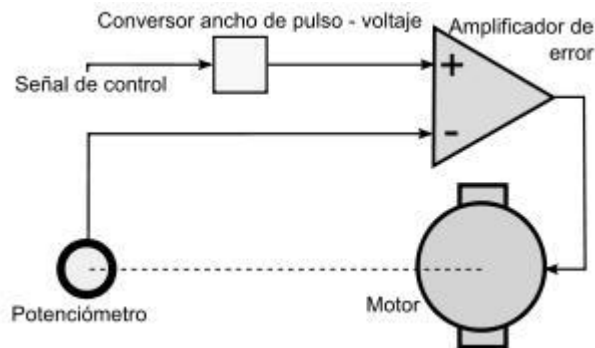


FIGURA 3.25: CONVERTOR DE ANCHO PULSO [22]

3.7.4 TEORIA PWM

Es una técnica de modulación en la que se modifica el ciclo de trabajo de una señal periódica para, entre otras cosas, variar la velocidad de un motor. El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación al período. Cuando más tiempo pase la señal en estado alto, mayor será la velocidad del motor. Este tren de pulsos, en realidad, hace que el motor marche alimentado por la tensión máxima de la señal durante el tiempo en que esta se encuentra en estado alto, y que pare en los tiempos en que la señal está en estado bajo.

3.7.4.1 CICLO DE TRABAJO

Recibe este nombre la relación de tiempos entre el estado alto y bajo de la señal utilizada. Se expresa como un porcentaje entre el periodo y el ancho del pulso. Cuando el ciclo de trabajo es cercano al 100%, el motor girará a una velocidad cercana a la máxima, ya que la tensión promedio aplicada en sus bornes será casi igual a V. Si el ciclo de trabajo se aproxima a 0%, el motor girará muy despacio, ya que la tensión promedio será casi cero.

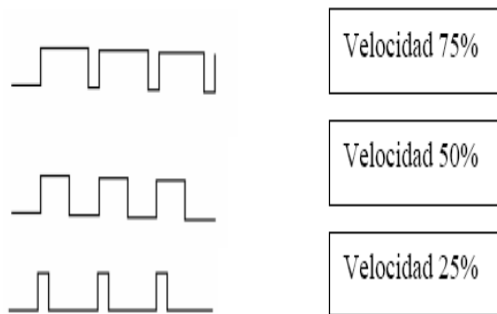


FIGURA 3.26: CICLO DE TRABAJO

3.8 SISTEMAS MECÁNICOS

Un mecanismo es un conjunto de elementos generalmente rígidos, cuyo propósito es transmitir ó convertir el movimiento, algunos ejemplos pueden ir desde un simple sacapuntas de manivela, lámpara ajustable de escritorio y sombrilla.

Por otro lado una máquina la podemos definir como un sistema de elementos dispuestos para transmitir movimiento y energía en un modo predeterminado, algunos ejemplos pueden ser una batidora o mezcladora de alimentos, puerta de la bóveda de un banco, engranaje de transmisión de un automóvil o de un robot.

3.8.1 TIPOS DE PLATAFORMAS MÓVILES

El primer paso que se da en la construcción del robot es la elección de su configuración, esto es, definir como estarán distribuidos los principales elementos que lo componen: ruedas, plataforma, motores. La precisión de las mediciones que haga nuestro robot, dependerá en gran medida de la configuración que le demos.

La elección de la plataforma móvil es sin duda la parte más fundamental a la hora de diseñar una plataforma, dependiendo de las necesidades que necesitemos, buscaremos unas u otras características, dichas características son principalmente respuesta ante cualquier superficie, velocidad, maniobrabilidad, equilibrio, etc. En relación a las ruedas, existen distintas

configuraciones, típicamente utilizadas en robótica móvil: diferencial, triciclo, Ackerman, sincronizada, omnidireccional, con múltiples grados de libertad y movimiento mediante orugas.

3.8.1.1 CONFIGURACIÓN DIFERENCIAL

La configuración diferencial en plataformas móviles se presenta como la más sencilla de todas. Consta de dos ruedas situadas diametralmente opuestas en un eje perpendicular a la dirección del robot. Cada una de ellas irá dotada de un motor, de forma que los giros se realizan dándoles diferentes velocidades. Así, si queremos girar a la derecha, daremos mayor velocidad al motor izquierdo.

Para girar a la izquierda, será el motor derecho el que posea mayor velocidad. Con dos ruedas es imposible mantener la horizontalidad del robot. Se producen cabeceos al cambiar la dirección. Para solventar este problema, se colocan ruedas “locas”. Estas ruedas no llevan asociadas ningún motor, giran libremente según la velocidad del robot. Además, pueden orientarse según la dirección del movimiento, de forma análoga a como lo hacen las ruedas traseras de los carritos del supermercado, dependiendo de las necesidades, se pueden colocar una, dos o más ruedas “locas”.

Sin embargo, hay que tener en cuenta que a la hora de diseñar, la presencia de más de tres apoyos en el robot (incluidas las dos ruedas de tracción), puede llevar a graves cálculos de odometría en terrenos irregulares, e incluso a pérdida de tracción total. En la figura 3.28 se aprecia cómo la rueda de tracción pierde agarre, haciendo imposible el avance del robot:

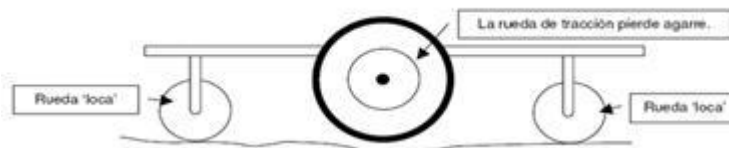


FIGURA 3.27: TRACCIÓN ERRÓNEA EN UN PAR DIFERENCIAL [23]

Podemos resumir la configuración como: dos ruedas con tracción independiente, y una o más ruedas locas. Para llevar a cabo una navegación por odometría, es necesario acoplar a los motores de las ruedas laterales encoders, de forma que contando los pulsos que avanza cada rueda y teniendo en cuenta el radio de la misma y la reducción del motor, no hay más que aplicar las ecuaciones cinemáticas del robot para hallar la posición exacta en la que se encuentra y el ángulo de desviación respecto a una dirección de referencia.

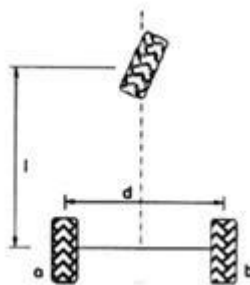


FIGURA 3.28: TRACCIÓN EN UN PAR DIFERENCIAL [23]

3.8.1.2 CONFIGURACIÓN EN TRICICLO

En este caso, se dispone de tres ruedas en el robot, situadas de forma similar a los triciclos de los niños, de ahí su nombre. Tendremos por tanto, dos ruedas traseras, que no llevan acopladas ningún motor. La tracción estará en la rueda delantera, que además, será la que usaremos para dirigir al robot. En este caso, el cálculo de la odometría es mucho más sencillo. La posición del robot vendrá dada por el número de pulsos que avanza el encoder de la rueda motora, y la dirección es simplemente la que lleve dicha rueda.

Un problema asociado a esta configuración es que el centro de gravedad tiende a alejarse de la rueda de tracción terrenos inclinados cuando el robot lleva la dirección de subida. Esto se traduce en una pérdida de la tracción del robot. Al perderse el contacto con el suelo la rueda motora sigue girando, pero el robot no avanza. Esto supone un error grande al hacer el cálculo de la odometría, ya que el robot indica que está en un punto más avanzado, cuando en realidad se encuentra más atrás.

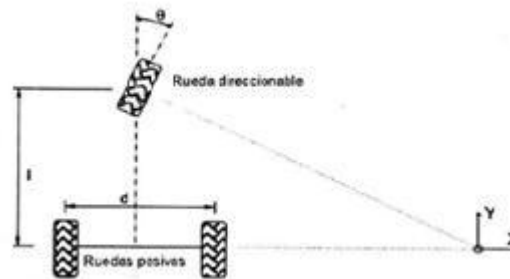


FIGURA 3.29: CONFIGURACIÓN EN TRICICLO [23]

3.8.1.3 CONFIGURACIÓN ACKERMAN

Se usa casi exclusivamente en la industria del automóvil. Es la configuración que llevan los coches: dos ruedas con tracción traseras, y dos ruedas de dirección delanteras. Esta configuración está diseñada para que la rueda delantera interior en un giro tenga un ángulo ligeramente más agudo que la exterior, y evitar así el derrape de las ruedas.

Como se puede apreciar en la figura 3.31, las normales a ambas ruedas se cortan en un punto, que se encuentra sobre la prolongación del eje de las ruedas traseras. Así, se puede comprobar que las trayectorias de ambas ruedas para ángulos de giro constantes son circunferencias concéntricas.

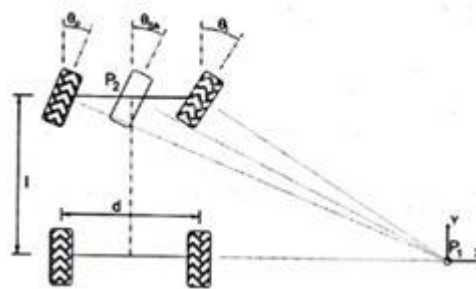


FIGURA 3.30: CONFIGURACIÓN ACKERMAN [24]

La relación entre los ángulos de las ruedas de dirección (figura 3.31) viene dada por la ecuación de Ackerman:

$$\cot(\theta_1) - \cot(\theta_2) = d / l$$

donde:

θ_1 = ángulo relativo de la rueda interior

θ_2 = ángulo relativo de la rueda exterior

l = separación longitudinal entre ruedas

d = separación lateral entre ruedas

TABLA 3.14: ECUACIÓN DE ACKERMAN

La configuración de Ackerman da una solución bastante precisa para la odometría a la vez que constituye un buen sistema de tracción incluso con terrenos inclinados. No obstante, la construcción mecánica de un robot con configuración Ackerman se complica de forma exponencial respecto a las anteriores. Por otro lado, el robot propuesto va a operar en interiores de edificios, con lo que el terreno no va a presentar dificultades como para necesitar una configuración tan compleja.

3.8.1.4 CONFIGURACIÓN DE DIRECCIÓN SINCRONIZADA

Supone una configuración innovadora. Consiste en tres o más ruedas, todas ellas dotadas de tracción y acopladas mecánicamente, de forma que todas rotan en la misma dirección y a la misma velocidad. Se necesita que todas ellas pivoten de la misma manera al cambiar la dirección. Este sistema necesita de una gran sincronización, que redundará en una odometría mejorada reduciendo el deslizamiento de las ruedas respecto al suelo, ya que todas las ruedas generan fuerzas con vectores de igual módulo y paralelos en todo momento.

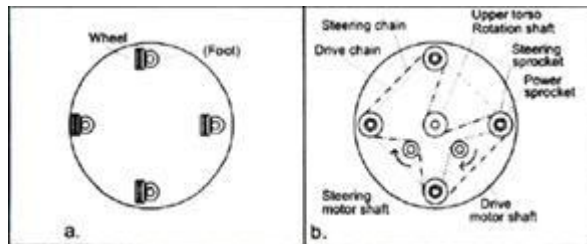


FIGURA 3.31: CONFIGURACIÓN SINCRONIZADA [24]

3.8.1.5 CONFIGURACIÓN OMINDIRECCIONAL

Se trata de dotar al robot con ruedas omnidireccionales.

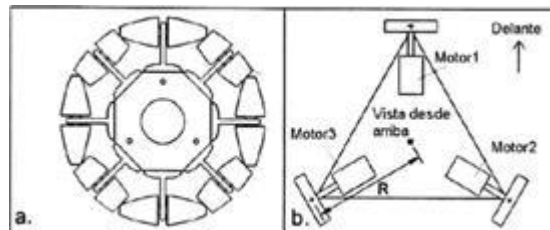


FIGURA 3.32: CONFIGURACIÓN OMNIDIRECCIONAL [24]

El movimiento omnidireccional es de gran ayuda ya que le permite al robot desplazarse en diferentes direcciones sin tener que rotar, ahorrando tiempo y sobre todo le permite moverse en espacios reducidos ya que al rotar como se acostumbraría normalmente ocuparía un poco más de espacio.

3.8.1.6 CONFIGURACIÓN MEDIANTE ORUGAS

Se trata de sustituir las ruedas por orugas. Es un caso particular de la tracción diferencial. Sin embargo, en esta configuración el deslizamiento en los giros es muy grande, perdiéndose bastante precisión en el cálculo odométrico. Se emplea en casos en los que el terreno presente irregularidades. La ventaja que presenta es que es simple de controlar.

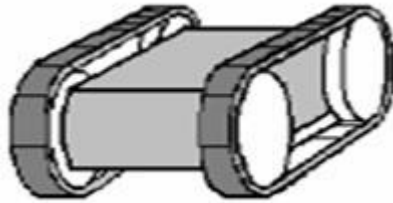


FIGURA 3.33: CONFIGURACIÓN MEDIANTE ORUGAS

3.8.2 BRAZOS MECÁNICOS

Por siglos el ser humano ha construido máquinas que imiten las partes del cuerpo humano. Los antiguos egipcios unieron brazos mecánicos a las estatuas de sus dioses. Estos brazos fueron operados por sacerdotes, quienes clamaban que el movimiento de estos era inspiración de sus dioses. Los griegos construyeron estatuas que operaban con sistemas hidráulicos, los cuales se utilizaban para fascinar a los adoradores de los templos.

En la actualidad el uso de los robots industriales está concentrado en operaciones muy complejas, como tareas que requieren de una gran precisión. Un robot industrial es una máquina programable de uso general que tiene algunas características antropomórficas o humanoides. Las características humanoides más típicas de los robots actuales es la de sus brazos móviles, los que se desplazarán por medio de secuencias de movimientos que son programados para la ejecución de tareas de utilidad.

3.8.2.1 GRADOS DE LIBERTAD

Los grados de libertad (GDL) de un sistema es el número de parámetros independientes que se necesitan para definir unívocamente su posición en el espacio en cualquier instante es decir, el número de grados de libertad en ingeniería se refiere al número mínimo de números reales que necesitamos especificar para determinar completamente la velocidad de un mecanismo o el número de reacciones de una estructura.

En el plano se requiere de tres parámetros (GDL): dos coordenadas lineales(x, y) y una coordenada angular (q). En el espacio se requiere de seis GDL: tres distancias (x,y,z) y tres ángulos(q ,f ,r).

3.8.2.1.1 DETERMINACIÓN DEL GRADO DE LIBERTAD

Grados de libertad en mecanismos planos:

Para un mecanismo plano cuyo movimiento tiene lugar sólo en dos dimensiones, el número de grados de libertad del mismo se pueden calcular mediante el criterio de **Grübler-Kutzbach**:

$$GL = 3(n - 1) - 2j_1 - j_2$$

Donde:

GL, grados de libertad del mecanismo.

n: número de elementos de un mecanismo.

j_1 : número de uniones de 1 grado de libertad.

j_2 : número de uniones de 2 grados de libertad.

TABLA 3.15: G. LIBERTAD EN MECANISMOS PLANOS

Es importante saber que esta fórmula es válida sólo en el caso de que no existan enlaces redundantes, es decir enlaces que aparecen físicamente en el mecanismo pero no son necesarios para el movimiento de éste. Para poder emplear el criterio, debemos eliminar los enlaces redundantes y calcular entonces los grados de libertad del mecanismo.

Todas las partes fijas (uniones al suelo) se engloban como el primer elemento. Aunque el grado de libertad de algunas uniones es fácil de visualizar, en otras ocasiones se pueden cambiar por sistemas equivalentes.

Grados de libertad en estructuras:

Podemos extender la definición de grados de libertad a sistemas mecánicos que no tienen capacidad de moverse, llamados estructuras fijas. En el caso particular de estructuras de barras en d dimensiones, si n es el número de barras y existen m restricciones (uniones entre barras o apoyos) que eliminan cada una r_i grados de libertad de movimiento; definimos el número de grados de libertad aparentes como:

$GL = \left[d + \binom{d}{2} \right] (n - 1) - \sum_{i=1}^m r_i$	<p>Donde:</p> <p>GL: Grados de libertad del mecanismo.</p> <p>n: Número de elementos de barras de la estructura.</p> <p>r_i: Número de grados de libertad eliminados por la restricción $i \in 1, \dots, m$</p>
---	---

TABLA 3.16: G. LIBERTAD EN ESTRUCTURAS

3.8.3 GRIPPER

Diseñar una mano robótica no es tarea fácil. Además de la cantidad de articulaciones que se necesitan para darle versatilidad, se necesita también sensibilidad táctil. En este caso solo basto con hacer una pinza llamada gripper que consta de 2 terminales de sujeción y un servomotor para quitar los cables.

Este tipo de pinzas robóticas han sido de gran utilidad en la medicina especialmente en cirugías, ya que son capaces de realizar movimientos de 180° en oposición a los movimientos del cirujano, lo que permite acceder a la zona afectada aunque no se disponga de ángulo.



FIGURA 3.34: GRIPPER [25]

3.9 SISTEMAS DE VISIÓN ARTIFICIAL

Un sistema de visión es un conjunto de elementos que permiten obtener imágenes del entorno, procesarlas y tomar ciertas decisiones basadas en la evaluación de las imágenes adquiridas.

La Visión artificial, también es conocida como Visión por Computadora o Visión técnica, el cual es considerado como un subcampo de la inteligencia artificial. El propósito de la visión artificial es programar un computadora para que "entienda" una escena o las características de una imagen.

Alguna de sus características de la visión artificial se puede ejemplificar con el siguiente diagrama:

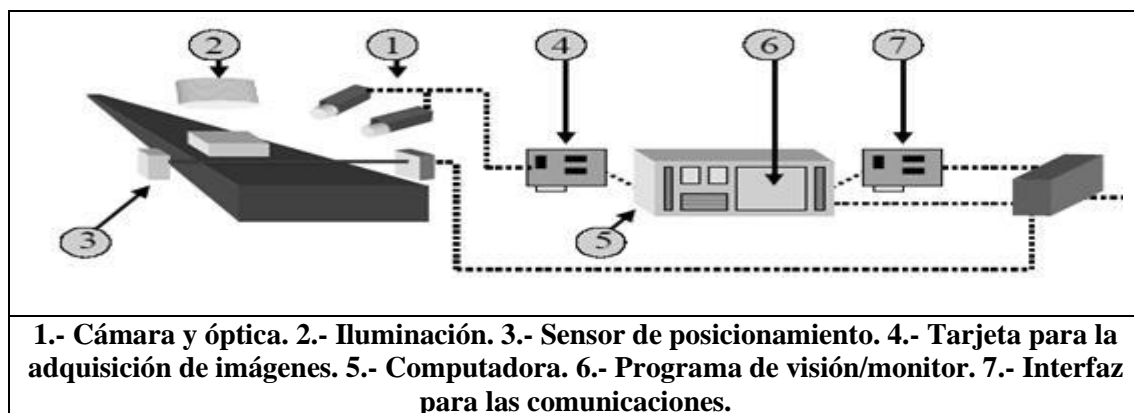


FIGURA 3.35: DIAGRAMA DE VISIÓN ARTIFICIAL [26]

De los cuales sus principales componentes fundamentales los podemos explicar de la siguiente forma:

1. **Iluminación:** Es un componente fundamental de cualquier sistema de visión. Las fuentes de iluminación fundamentales son las luces fluorescentes, leds y luces halógenas. A continuación se muestran algunos ejemplos:

Cámara con iluminación de leds incorporados [27].	Luz fluorescente en forma de anillo [28].	Sistemas para iluminar áreas [29].	Luz halógena [30].

FIGURA 3.36: TIPOS DE ILUMINACIÓN

2. **Cámara y óptica:** Es el elemento primario de adquisición de la imagen. Puede incorporar sensores CCD (el CCD es el sensor con diminutas células fotoeléctricas que registran la imagen. Desde allí la imagen es procesada por la cámara y registrada en la tarjeta de memoria) cuyo tamaño de píxel y características de la lente, determinan la resolución del objeto presente en el campo de visión. En algunos casos puede integrar la electrónica necesaria para adquirir la información y enviarla directamente a un monitor, sin necesidad de ordenador o transferir la información a una red de área local. Adicionalmente, puede

poseer software para evaluar las características de la imagen. Algunas de las más utilizadas en robótica son:



FIGURA 3.37: TIPOS DE CÁMARAS

Es importante mencionar que una adecuada calidad de imagen (que determina la extracción de la información para su análisis) depende de estos tres elementos: La iluminación, la lente u óptica y la cámara.

3. **Tarjetas para la adquisición de imágenes:** Son tarjetas encargadas de transferir las imágenes capturadas por el sensor CCD de la cámara a la memoria del centro de procesamiento de las mismas (computadora). A las mismas se les exige fundamentalmente alta velocidad de adquisición de los grandes volúmenes de datos que genera la cámara.
4. **Programa para el procesamiento de imágenes:** Es el encargado de extraer la información necesaria de la imagen. El programa de procesamiento aplica filtros, detecta bordes, segmenta la imagen, ecualiza el histograma y ejecuta los algoritmos necesarios para ejecutar las tareas que se le exige al sistema de visión: reconocimiento de caracteres, lectura de matrículas de vehículos, seleccionar piezas defectuosas con un brazo robótico en una línea de producción, leer códigos de barras, seleccionar la calidad de las frutas por análisis del color, inspeccionar la calidad de las piezas mecánicas, etc.
5. **Comunicaciones:** Las comunicaciones en el sistema de visión permiten conectar el mismo a otros dispositivos, como pueden ser microcontroladores, centro de decisiones de vehículos autónomos, interfaces con un operador, etc. Son un elemento fundamental que, junto a la adecuada selección de interfaces y conectores, determina la velocidad con que puede actuar el sistema en su conjunto.

3.9.1 TEORÍA DE COLORES

En el arte de la pintura, el diseño gráfico, la fotografía, la imprenta y en la televisión, la teoría del color es un grupo de reglas básicas en la mezcla de colores para conseguir el efecto deseado combinando colores de luz o pigmento. La luz blanca se puede producir combinando el rojo, el verde y el azul, mientras que combinando pigmentos cian, magenta y amarillo se produce el color neutro.

3.9.1.1 SISTEMA DE VISIÓN RGB

Este es uno de los modelos anteriores solo que se profundiza un poco más ya que este modelo fue el que se ocupó en este proyecto. Se le conoce como modelo de color en síntesis aditiva lo que quiere decir que es posible representar un color mediante la mezcla por adición de los tres colores primarios: rojo, azul y verde.

Todos los colores posibles que pueden ser creados por la mezcla de estas tres luces de color son aludidos como el espectro de color de estas luces. Cuando ningún color luz está presente, uno percibe el negro. Los colores luz tienen aplicación en los monitores, televisores, proyectores de vídeo y todos aquellos sistemas que utilizan combinaciones de materiales que fosforecen en el rojo, verde y azul.

Para indicar con qué proporción mezclamos cada color, se asigna un valor a cada uno de los colores primarios, de manera, por ejemplo, que el valor 0 significa que no interviene en la mezcla y, a medida que ese valor aumenta, se entiende que aporta más intensidad a la mezcla. Aunque el intervalo de valores podría ser cualquiera (valores reales entre 0 y 1, valores enteros entre 0 y 37, etc.), es frecuente que cada color primario se codifique con un byte (8 bits). Así, de manera usual, la intensidad de cada una de las componentes se mide según una escala que va del 0 al 255. Por lo tanto, el rojo se obtiene con (255, 0, 0), el verde con (0, 255, 0) y el azul con (0, 0, 255), obteniendo, en cada caso un color resultante monocromático. La ausencia de color lo que nosotros conocemos como color negro se obtiene cuando las tres componentes son 0, (0, 0, 0).

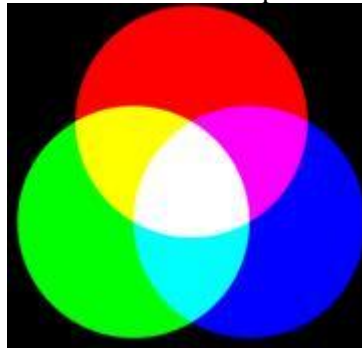


FIGURA 3.38: MEZCLA ADITIVA DE COLORES

En este tipo de modelos se encuentran los espacios de colores, donde un espacio de color define un modelo de composición del color. Por lo general un espacio de color lo define una base de N vectores en este caso el espacio RGB lo forman 3 vectores: Rojo, Verde y Azul, cuya combinación lineal genera todo el espacio de color. Los espacios de color más generales intentan englobar la mayor cantidad posible de los colores visibles por el ojo humano, aunque existen espacios de color que intentan aislar tan solo un subconjunto de ellos.

Existen espacios de color de:

- Una dimensión: escala de grises, escala Jet.
- Dos dimensiones: sub-espacio rg, sub-espacio xy.
- Tres dimensiones: espacio RGB, HSV, YCbCr, YUV.
- Cuatro dimensiones: espacio CMYK.

De los cuales, los espacios de color de tres dimensiones son los más extendidos y los más utilizados. Entonces, un color se especifica usando tres coordenadas, o atributos, que representan su posición dentro de un espacio de color específico. Estas coordenadas no nos dicen cuál es el color, sino que muestran dónde se encuentra un color dentro de un espacio de color en particular.

Como se definió en los párrafos anteriores RGB es conocido como un espacio de color aditivo (colores primarios) porque cuando la luz de dos diferentes frecuencias viaja junta, desde el

punto de vista del observador, estos colores son sumados para crear nuevos tipos de colores. Los colores rojo, verde y azul fueron escogidos porque cada uno corresponde aproximadamente con uno de los tres tipos de conos sensitivos al color en el ojo humano (65% sensibles al rojo, 33% sensibles al verde y 2% sensibles al azul). Con la combinación apropiada de rojo, verde y azul se pueden reproducir muchos de los colores que pueden percibir los humanos. Por ejemplo, rojo puro y verde claro producen amarillo, rojo y azul producen magenta, verde y azul combinados crean cian y los tres juntos mezclados a máxima intensidad, crean el blanco.

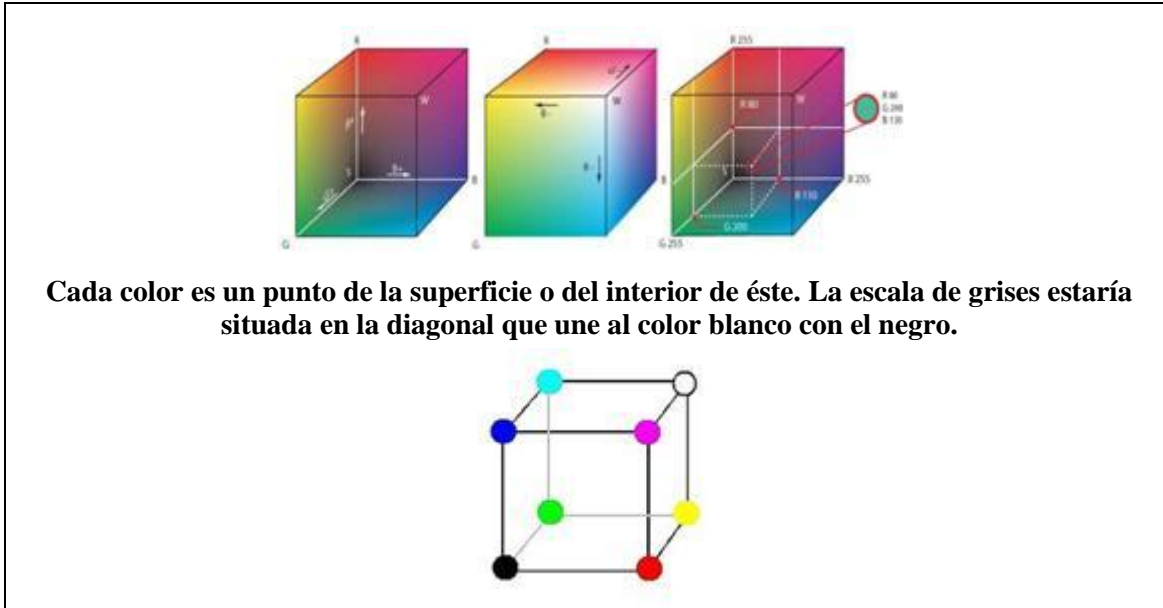


FIGURA 3.39: CUBO DE COLOR RGB [33]

3.9.2 CÁMARA CMUCAM V3

La CMUCAM es un sistema embebido formado por un módulo sensor de video, un procesador y puerto de entrada y salida tiene una resolución (352 x 288) pixeles, por tanto son imágenes muy pequeñas. Entre una de sus características en que cuenta con una ranura para insertarle una tarjeta MMC flash, carga imágenes en la memoria a una velocidad de 26 cuadros por segundo, posee una salida analógica de video.

Tiene un software para comprimir en formato JPEG, así como incluye un software de programación compatible con Linux y Windows en un ambiente de desarrollo libre (open source), donde podemos encontrar el CYGWIN el cual nos va a servir para poder compilar los programas hechos para la Cmucam, el LPC210x FLASH UTILITY con el cual vamos a poder grabar los programas al microcontrolador, y el CC3 que es una carpeta la cual tiene varias aplicaciones para la CMUCAM.



FIGURA 3.40: CMUCAM V3 [31]

Posee propiedades para trabajar en tres espacios de color como son: RGB, YCrCb y HSV. Se le pueden conectar hasta cuatro servomotores, ya que a generalmente va montada arriba de brazos mecánicos o sistemas que giren. Para poder establecer comunicación con la CMUCAM hacemos uso del puerto serial, a través de la comunicación UART. El siguiente diagrama nos muestra como está físicamente el sistema embebido de la cámara:

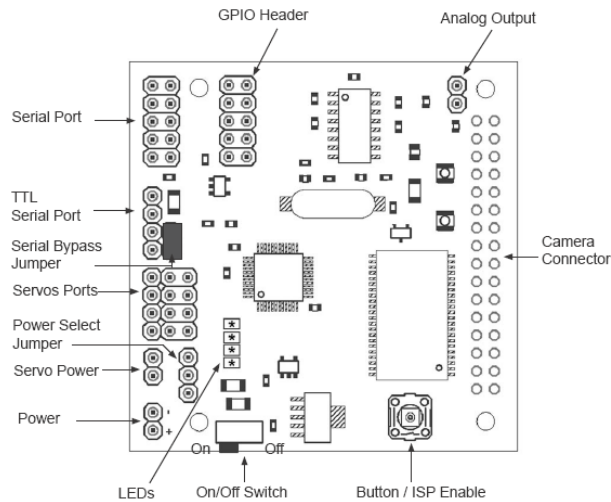


FIGURA 3.41: SISTEMA EMBEBIDO DE VISIÓN [31]

En el cual podemos observar el puerto serial, el puerto serial TTL, puerto para la conexión de los servomotores, pines para conectar la fuente de alimentación la cual puede estar en un rango de 6 a 15 volts de corriente directa, hasta 150 miliamperios de corriente, tiene un botón de encendido y apagado, un botón para cuando lo tengamos que programar (ISP), el conector de la cámara, también podemos visualizar la salida analógica de video.

La placa que contiene el chip de la cámara es montada en paralelo con la tarjeta del microprocesador, y se conecta empezando por el pin 1 como el siguiente diagrama:

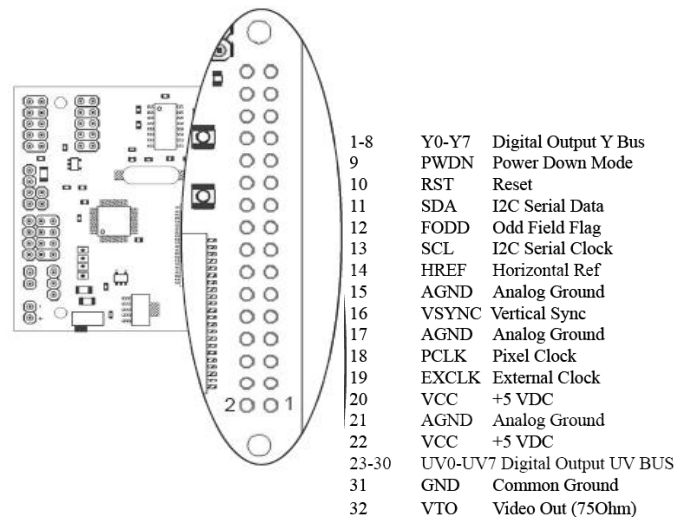


FIGURA 3.42: DIAGRAMA DE CONEXIÓN DEL CHIP DE LA CÁMARA Y EL MICROPROCESADOR [31]

Como se menciona anteriormente la CMUcam3 tiene la capacidad de controlar hasta cuatro servomotores, siendo esto opcional porque se pueden manejar con una tarjeta controladora de servos.

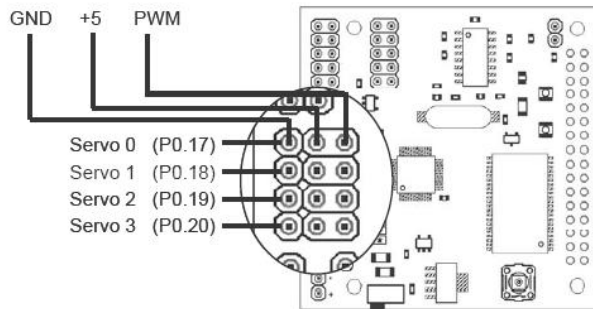


FIGURA 3.43: DIAGRAMA DEL PUERTO PARA SERVOMOTORES [31]

3.10 SENSORES COMO ELEMENTOS DE SEÑALIZACIÓN

Los sensores desempeñan un papel fundamental en la robótica, ya que a partir de las informaciones captadas por ellos, el robot actúa en consecuencia. Para ello, convierte una magnitud física en una señal eléctrica codificada, que puede ser analógica o digital.

Los sensores de tacto básicamente, consisten en un interruptor con dos posiciones, que según donde se sitúe, proporciona mayor o menor información. Por ejemplo, situado en el brazo, sólo da seguridad en relación con los obstáculos, mientras que, en la pinza puede dar información más estratégica.

El pulsador es un sensor digital muy sencillo que cierra o abre un circuito al pulsarlo. Tiene muchas aplicaciones en minirobots, por ejemplo en la detección de obstáculos.

El montaje general, que suele funcionar, es: si el interruptor no está pulsado (circuito abierto) la patilla del micro estará a 5V (1 digital) y cuando pulsamos, cerramos el circuito y en la señal que llega al micro es 0V (0 digital).

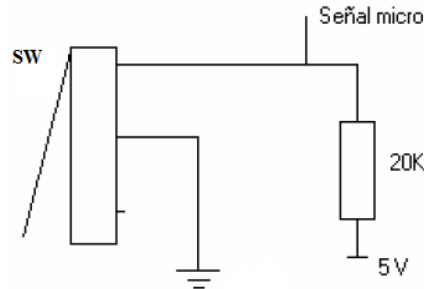


FIGURA 3.44: SWITCH DE CONTACTO

3.11 SISTEMAS EMBEBIDOS

Un sistema embebido o integrado es un sistema de uso específico construido dentro de un dispositivo mayor. Los sistemas integrados se utilizan para usos muy diferentes de los usos generales para los que se emplea una computadora personal. En un sistema embebido la mayoría de los componentes se encuentran incluidos en la placa base y algunos de sus componentes pueden ser: la tarjeta de vídeo, audio, módem, etc.

Dos de las diferencias principales son el precio y el consumo. Puesto que los sistemas integrados se pueden fabricar por decenas de millares o por millones de unidades, una de las principales preocupaciones es reducir los costes. Los sistemas integrados suelen usar un procesador relativamente pequeño y una memoria pequeña para reducir los costes. Se enfrentan, sobre todo, al problema de que un fallo en un elemento implica la necesidad de reparar la placa íntegra.

Los sistemas integrados emplean a menudo periféricos controlados por interfaces síncronos en serie, que son de diez a cientos de veces más lentos que los periféricos de una computadora personal normal.

Algunas de sus características son:

En la parte central se encuentra el microprocesador, microcontrolador, DSP, etc. Es decir, el CPU o unidad que aporta capacidad de cómputo al sistema, pudiendo incluir memoria interna o externa, un micro con arquitectura específica.

La comunicación adquiere gran importancia en los sistemas integrados. Lo normal es que el sistema pueda comunicarse mediante interfaces estándar de cable o inalámbricas. Así un SE normalmente incorporará puertos de comunicaciones del tipo RS232, RS485, SPI, PC, CAN, USB, IP, WiFi, GSM, GPRS, DSRC, etc.

El subsistema de presentación tipo suele ser una pantalla gráfica, táctil, LCD, alfanumérico, etc.

Denominamos actuadores a los posibles elementos electrónicos que el sistema se encarga de controlar. Puede ser un motor eléctrico, un conmutador tipo relé etc. El más habitual puede ser una salida de señal PWM para control de la velocidad en motores de corriente continua. El módulo de E/S analógicas y digitales suele emplearse para digitalizar señales analógicas procedentes de sensores, activar diodos LED, reconocer el estado abierto cerrado de un conmutador o pulsador, etc.

El módulo de reloj es el encargado de generar las diferentes señales de reloj a partir de un único oscilador principal. El tipo de oscilador es importante por varios aspectos: por la frecuencia necesaria, por la estabilidad necesaria y por el consumo de corriente requerido.

El módulo de energía (power) se encarga de generar las diferentes tensiones y corrientes necesarias para alimentar los diferentes circuitos del SE.

Además de los convertidores ac/dc y dc/dc, otros módulos típicos, filtros, circuitos integrados supervisores de alimentación, etc. El consumo de energía puede ser determinante en el desarrollo de algunos sistemas integrados que necesariamente se alimentan con baterías, con lo que el tiempo de uso del SE suele ser la duración de la carga de las baterías.

3.12 MÁQUINA DE ESTADOS O MÁQUINAS SECUENCIALES

Una máquina secuencial es aquella que su salida depende, además de sus entradas, de los estados anteriores del sistema (el sistema tiene memoria), es decir realiza una serie de operaciones una tras otra en el tiempo.

Las operaciones pueden ser diferentes dependiendo del estado de la máquina por mencionar un ejemplo tenemos un candado secuencial, el cual para abrirlo, se requiere hacerlo en el orden correcto, por el contrario tenemos candados que trabajan como máquinas combinatorias el cual va a abrirse no importando el orden en que se pongan los números siempre y cuando al final sea el correcto.

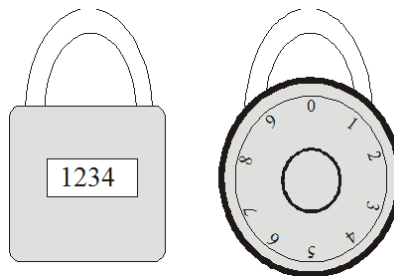


FIGURA 3.45: EJEMPLO MÁQUINAS SECUENCIALES Y COMBINATORIAS [34]

3.13 FUENTES DE ENERGÍA ELÉCTRICAS (PILAS)

La energía compactada en una pila permite escuchar música, operar a distancia equipos electrónicos y mantener en funcionamiento otros aparatos como cámaras fotográficas y teléfonos celulares.

¿Qué es una pila?

Una pila es una pequeña unidad electroquímica, contenida en una caja cuadrada o cilíndrica con dos terminales que representan los polos positivo y negativo. Sus componentes químicos se transforman en energía que hace funcionar a los aparatos. Por otro lado una batería contiene más de una pila o celda conectadas entre sí mediante un dispositivo permanente, junto con su caja y terminales.

¿Cuántos tipos de pilas existen?

Existen dos tipos: las primarias y las secundarias. Las primarias son las pilas desechables, cuyos componentes químicos, al convertirse en energía eléctrica, ya no pueden recuperarse. Las pilas secundarias son las que se pueden recargar y pueden llegar a sustituir hasta 300 pilas desechables.

De las que podemos encontrar:

Tipo de pila	Características tóxicas	Usos	Toxicidad
Primarias			
Secas o de carbón de zinc	Contienen muy poco mercurio (0.01%)	Linternas, radios, juguetes, etc.	Muy baja
Alcalinas	Tienen un contenido en mercurio del 0.5%	Juguetes, cámaras fotográficas, etc.	Tóxicas
Botón de óxido de mercurio (HgO)	Algunas contienen hasta un 30% de mercurio y litio	Aparatos de sordera, calculadoras, relojes e instrumentos de precisión.	Muy alta
Litio	Litio de 10% a 30%	En comunicaciones, radios portátiles, transmisores, instrumentos médicos, computadoras, calculadoras, celulares, agendas.	Muy alta
Verdes	Carecen de cadmio y mercurio, aunque se desconocen parte de sus componentes, tienen alcohol		Desconocida
Secundarias Recargables			
	Contienen cadmio, plomo y níquel, no contienen mercurio	Una pila recargable puede sustituir 300 desechables	Tóxicas
	Níquel-cadmio(Ni-cd) cadmio 18%	Juguetes, lámparas, artículos electrónicos, teléfonos inalámbricos, pueden durar 500 veces más que una pila de carbón zinc.	Tóxicas
	Níquel-metal Hidruro(Ni-MH) Níquel 25%	Productos electrónicos portátiles	Tóxicas
	Ion-Litio(Ion-Li)	Telefonía celular, computadoras, cámaras fotográficas y de video	Tóxicas
	Plomo	Uso automotriz, industrial y doméstico	Tóxicas

TABLA 3.17: TIPOS DE PILAS

CAPÍTULO 4

DESARROLLO

En este cuarto capítulo se explica cómo se construyó el robot y la implementación del software para hacer posible su funcionamiento.

4.1 CONSTRUCCIÓN DEL HARDWARE

El sistema electrónico, está constituido por todos aquellos elementos que tienen como función convertir las señales que genera el microcontrolador en señales que controlen el sistema mecánico y a su vez, interpretarlas para generar una respuesta previamente programada. El sistema electrónico cuenta con 3 módulos:

- Sistema embebido de control.
- Sistema electrónico de visión.
- Sistema electrónico de control de motores.

4.1.1 DISEÑO DEL SISTEMA EMBEBIDO.

La tarjeta fue diseñada para poder ocupar casi al 100% las funciones que trae el PIC como los puertos de entrada y salida (analógica y digital), comunicación UART, comunicación I²C, puertos Master-Slave, entre otras.

Para hacer el diseño se requirió del software libre llamado Eagle, con el cual se diseña la tarjeta electrónica, para esto partimos primero desarrollando el diagrama esquemático que son las conexiones internas entre los componentes del sistema.

El sistema embebido cuenta con un microcontrolador PIC18F452, un circuito max232 que permite la comunicación entre el PIC y la computadora o con otro dispositivo como la cámara, un Display de cristal líquido entre otros componentes como son resistencias, capacitores, y switches de contacto para realizar pruebas.

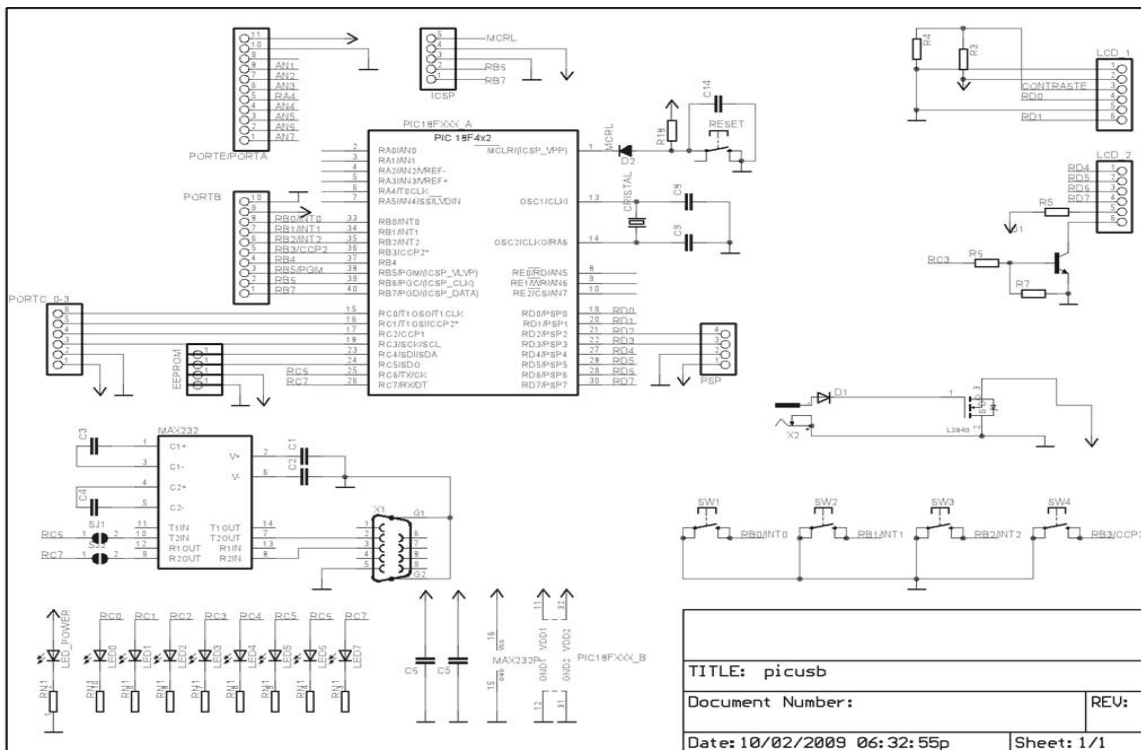


FIGURA 4.1: DIAGRAMA ELECTRICO KITPIC

Una vez que se terminó de crear el esquemático se hizo el board que es donde se empiezan a acomodar todas las pistas que unirán a todos los componentes puestos en el esquemático, y según se acomoden es como va a quedar la vista de la tarjeta en el diseño final.

La tarjeta fue diseñada por ambas caras, con esto, ahorramos un espacio significativo en la fabricación, así como también se ocuparon elementos de montaje superficial.

El diseño final de la tarjeta vista por la parte de arriba es la que se muestra en la figura 4.2:

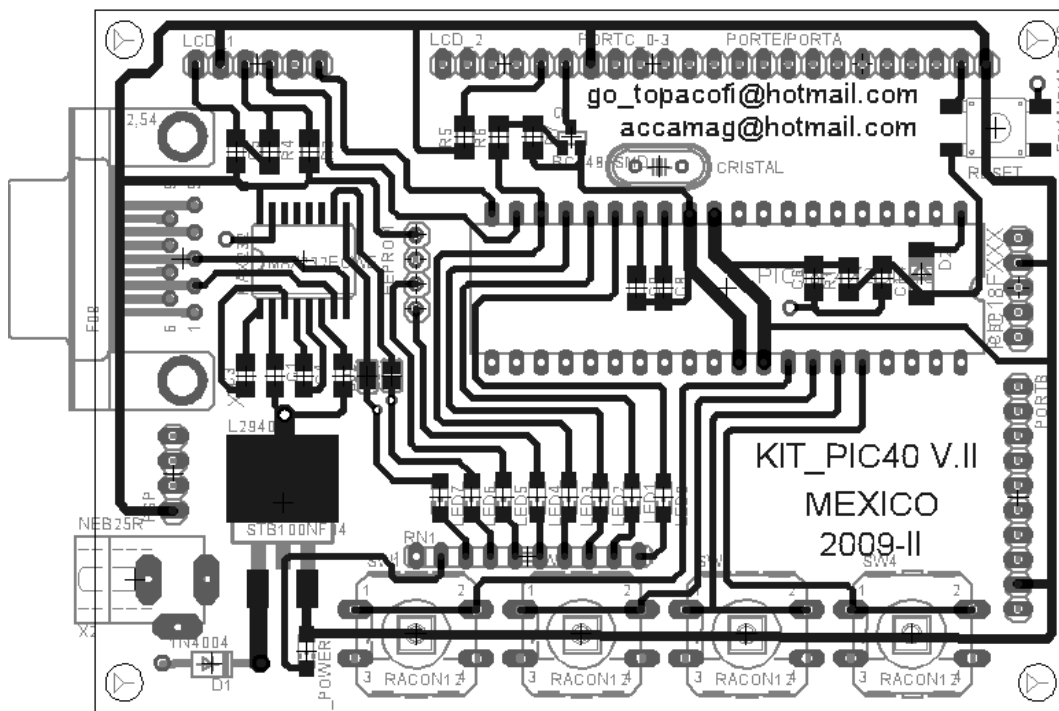


FIGURA 4.2: VISTA POR ARRIBA KITPIC

Una vez terminado el diseño, se prosiguió a imprimir en papel couche para posteriormente plancharlo en la placa de cobre y poder sumergirlo en cloruro férrico, de esta forma, el cloruro ataca las partes que no están cubiertas por la tinta y nos queda la tarjeta lista para barrenar y soldar. Terminada la tarjeta se ve de la siguiente forma:



FIGURA 4.3: TARJETA KIT PIC TERMINADA

Descripción del la tarjeta de desarrollo

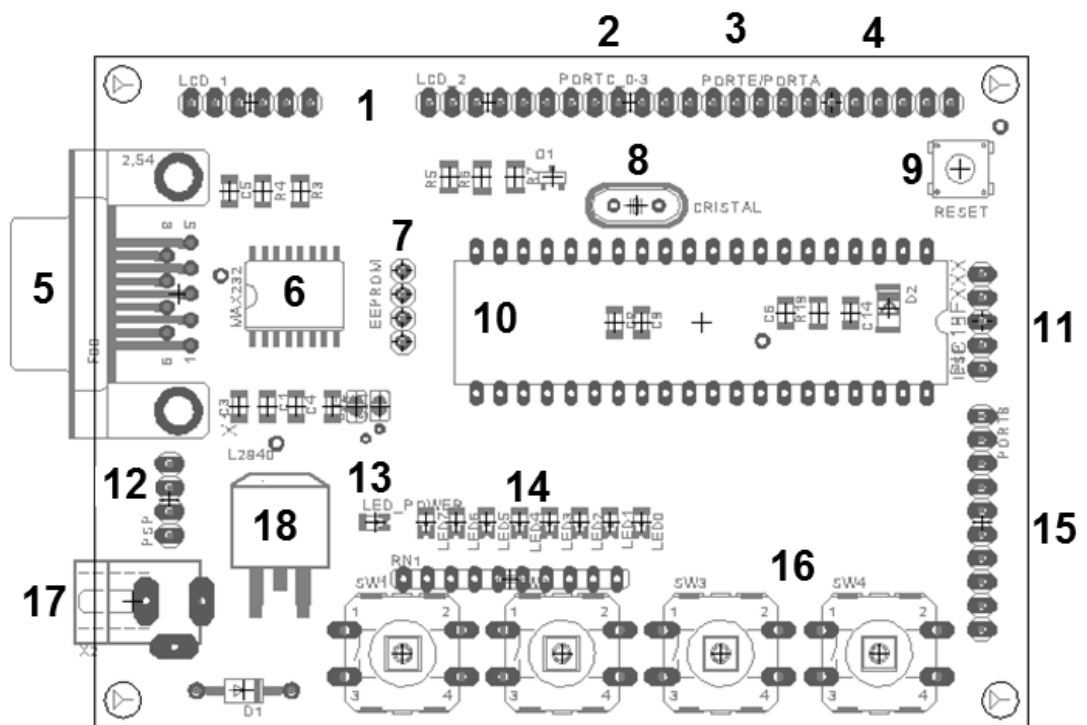


FIGURA 4.4: DESCRIPCIÓN DE LA TARJETA DE DESARROLLO

De acuerdo al esquema anterior a continuación se describe lo que significa cada círculo con un número:

1. Conector para el LCD(display).
2. Puerto C que consta de 8 bits
3. Puerto E que consta de 3 bits
4. Puerto A que consta de 7 bits
5. Conector db9 (dispositivo para la comunicación con la PC)
6. Circuito max 232 (hace posible la comunicación de la tarjeta con el puerto com de la PC)
7. Conector para trabajar con una EEPROM externa
8. Cristal de 10 mhz.
9. Botón de reset
10. Microcontrolador PIC18F452
11. Conector para el programador
12. Conector para configuración master-slave
13. Led indicador de que el sistema esta prendido
14. 8 Leds de prueba de salida de datos (conectados al puerto C)
15. Puerto B que consta de 8 bits
16. 4 Switches de prueba de entrada de datos (conectados al puerto B)
17. Conector para la fuente.
18. Regulador

4.1.2 DISEÑO DEL DRIVER PARA LOS MOTORES (PUENTE H)

El driver puente H se usos con los motores de las llantas y el motor que hace que se levante el brazo, elegimos usar este porque es muy fácil implementar las señales de control y de velocidad, ya que de ocupar otro aditamento (reles), no podríamos controlar estos parámetros que son esenciales en el desempeño del robot, la tarjeta se diseñó bajo el mismo ambiente Eagle, empezando por el esquemático.

Esquemático:

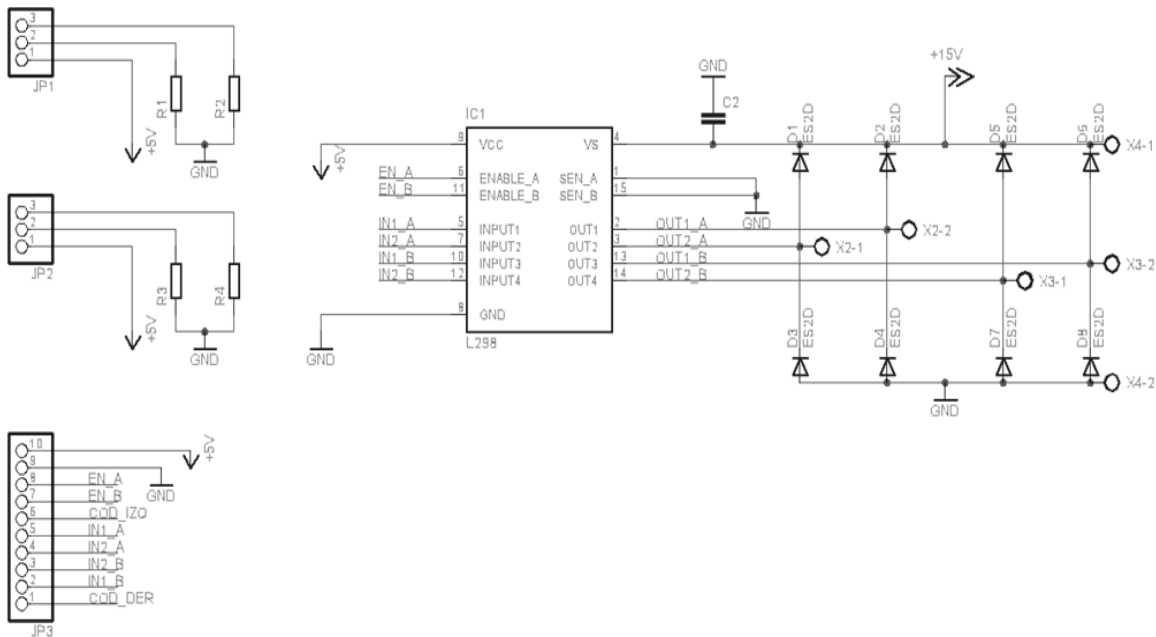


FIGURA 4.5: ESQUEMÁTICO DEL PUENTE H

En esta tarjeta se decidió usar el circuito L298, porque a diferencia del L293 este circuito puede soportar hasta 2 amperios por canal, que es lo mínimo que requerimos para trabajar los motores.

El diseño final de la tarjeta vista por la parte de arriba es la que se muestra en la figura 4.6 por el top.

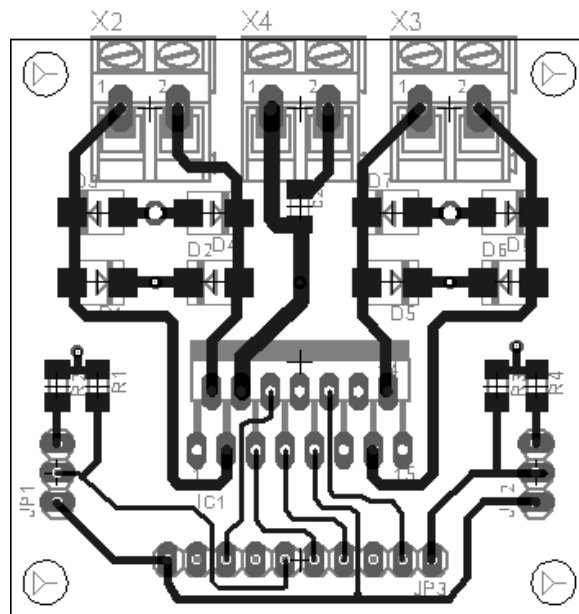


FIGURA 4.6: VISTA TOP PUENTE H

4.2 CONSTRUCCIÓN DEL ROBOT

Antes de diseñar la estructura del robot, hay que considerar el tipo de material que se va a ocupar, para nuestro caso, requerimos de un cuerpo lo suficientemente fuerte y a su vez, que sea muy ligero, ya que las condiciones físicas de la plataforma de prueba, ameritan que así sea, para poder subir una pendiente de 45°.

Considerando estas limitantes, en un principio se pensó en hacer la estructura con acrílico, ya que es muy resistente y en cierto modo, fácil de manejar, sin embargo, uno de nuestros problemas era que teníamos que hacer dobleces, esto es, teníamos que ser capaces de poder doblar el acrílico sin romperlo y para eso, necesitaríamos de un equipo especial para dicha tarea, además, no sabíamos cuanto iban a aguantar las uniones que debían ser pegadas. Otro problema al ocupar acrílico, era el costo, así como el hecho que teníamos en mente el poder desarrollar todo el robot con material reciclado, esto nos hizo pensar en ocupar lamina de acero delgada, así que nos dimos a la tarea de buscar en los depósitos de fierro viejo, algo que nos pudiera servir, sin embargo, no logramos encontrar nada concreto y lo que había, estaba muy oxidado, sin embargo, pudimos hallar estructura de aluminio que aunque estaba maltratada, no se oxida, es bastante fuerte y no pesa mucho, así que con la estructura de una ventana vieja de aluminio, obtuvimos el material necesario para poder construir nuestro robot, así que el material que se ocupó fue el siguiente:

En la siguiente figura 4.7, se muestra la estructura de aluminio que se empleó en la elaboración de este proyecto de tesis.

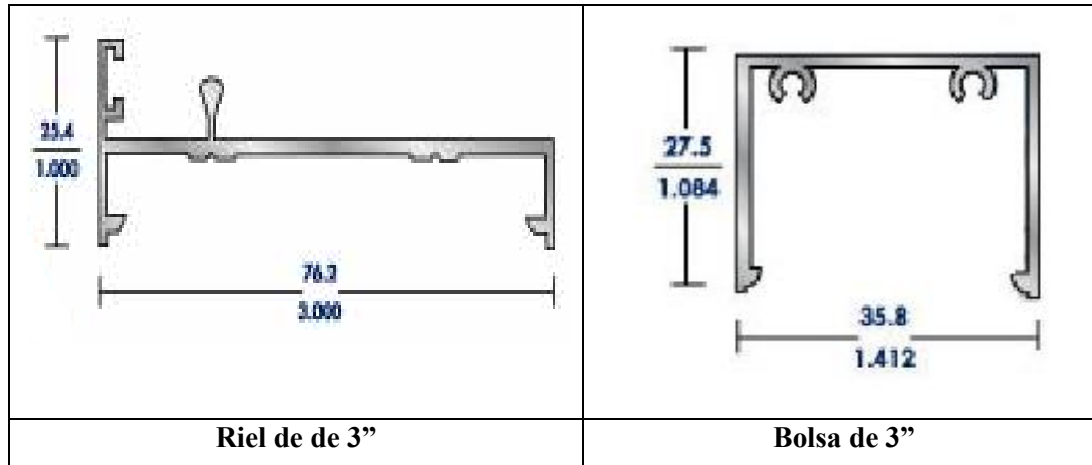


FIGURA 4.7: PERFILES DE ALUMINIO

Ya con la estructura de aluminio, nos dedicamos ahora a la locomoción del robot, para ello se tuvo la necesidad de buscar unos motores que tuviesen bastante torque y que no consumieran mucha corriente, así que se encontraron estos motores que se adaptaron a nuestras necesidades.



FIGURA 4.8: MOTOR DE LA BASE MÓVIL

A continuación se muestran las características de los motores que se ocuparon:

Material: Metal	Torque: 8 kgf*cm	Voltaje: 24 DC
Eje: Acero Plata	Velocidad sin Carga: 101± 10% rpm	Relación de Reducción: 120:1
Dirección de Rotación: bidireccional	Temperatura de Operación: -10°C ~ +60°C	Corriente con Carga: .90 A
Caja de Engranajes: Metálica	Humedad Soportada: 20% ~ 70%	Corriente sin Carga: 0.40 A Max.

TABLA 4.1: CARACTERÍSTICAS DEL MOTOR

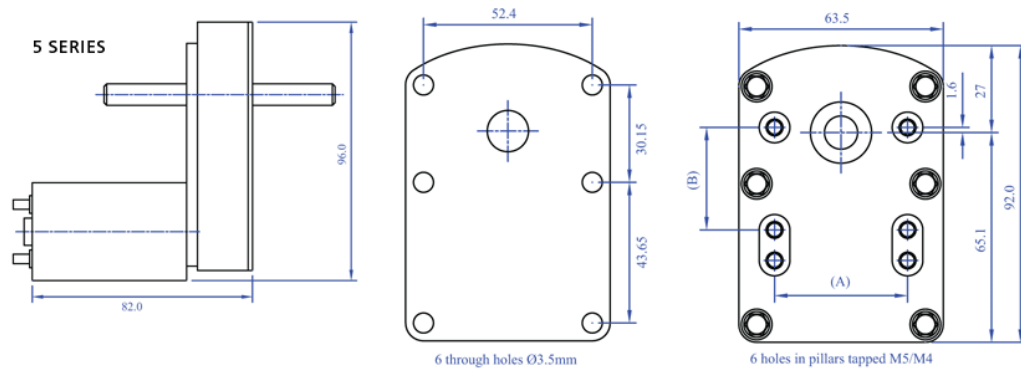


FIGURA 4.9: MEDIDAS FISICAS DEL MOTOR

Una de nuestras ideas fue desarrollar un sistema que pudiera moverse bien en el escenario, y que tuviera bastante agarre en la superficie, así que decidimos acondicionar a nuestra plataforma, unas orugas con una superficie plástica, para esto, tomamos idea de una plataforma comercial como la que se muestra continuación:



FIGURA 4.10: ROBOTSHOP-ROVER-ARDUINO-TANK-KIT [35]

Considerando lo anterior, nuestro sistema tendría que tener un sistema de locomoción similar, así que se tuvo la necesidad de requerir 3 pares de llantas como las que a continuación se muestran.



FIGURA 4.11: LLANTAS

4.2.1 CONSTRUCCIÓN DE LA BASE MÓVIL

Contando con las llantas, los motores, la estructura de aluminio y la banda plástica, proseguimos a armar la base móvil de nuestro robot, así que tomando en cuenta las dimensiones máximas permitidas en la construcción, hicimos los cortes necesarios a la estructura.

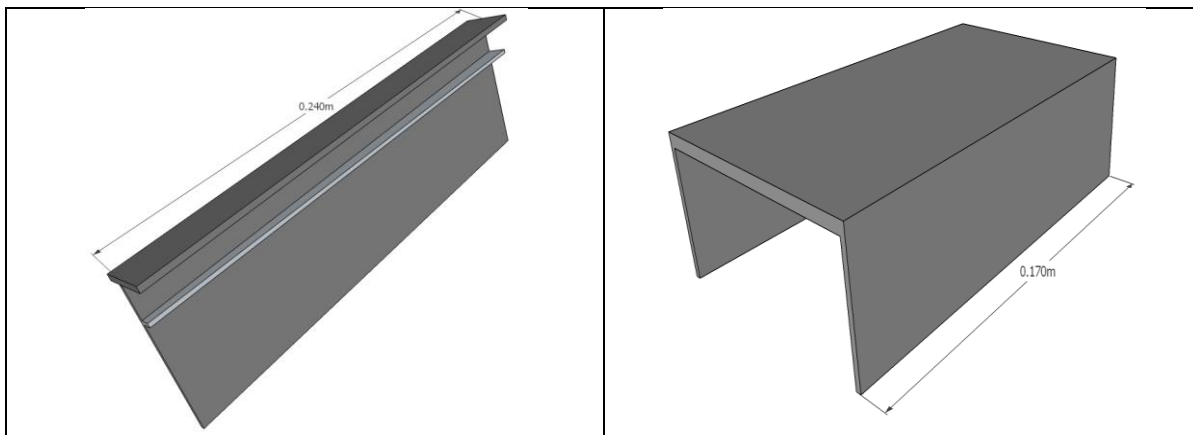


FIGURA 4.12: ESTRUCTURA DE LA BASE MÓVIL

Uno de los aspectos más importantes a considerar, son todos los aditamentos que se tienen que incorporar en la base móvil, así que, tomando en cuenta las dimensiones de los motores y de las llantas, proseguimos a armar su sistema de tracción, el cual, comenzamos por fijar las llantas a los motores.

En la figura (4.13) se muestra como el motor se acopla directamente a la llanta y permite el control de dirección del robot,

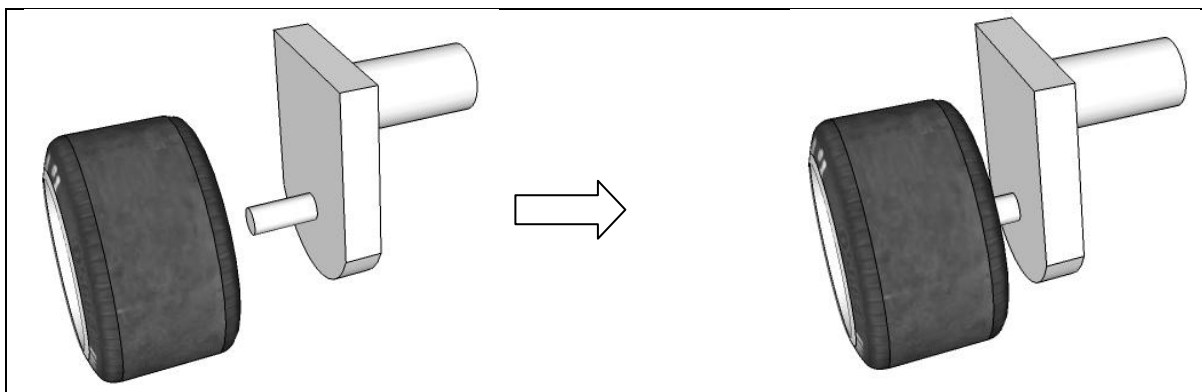


FIGURA 4.13: ACOPLAMIENTO DEL MOTOR CON LA LLANTA

El sistema acoplado a las llantas, delanteras, permiten un libre rodamiento con respecto a la banda de tracción, para asegurar que las llantas delanteras estuvieran alineadas, requerimos de un eje que las fijara a la base, y a su vez, nos diera la fuerza necesaria para mantenerlo derecho ante las tensiones y el movimiento, en la figura (4.14) se muestra cómo fue posible esto.

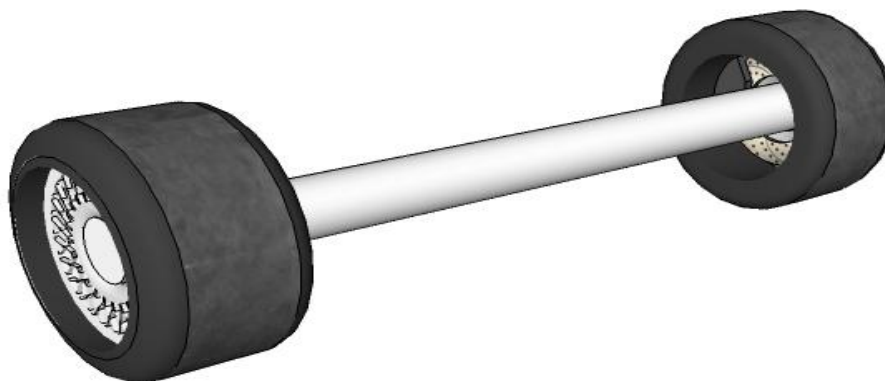


FIGURA 4.14: ACOPLAMIENTO DE LAS LLANTAS CON EL EJE

Esto permite que la tracción que viene del motor en la parte trasera, sea conducida por medio de la banda a la parte delantera, con esto, tenemos un sistema de movimiento tipo oruga. En la figura (4.15) se muestra el diseño mecánico de este sistema.

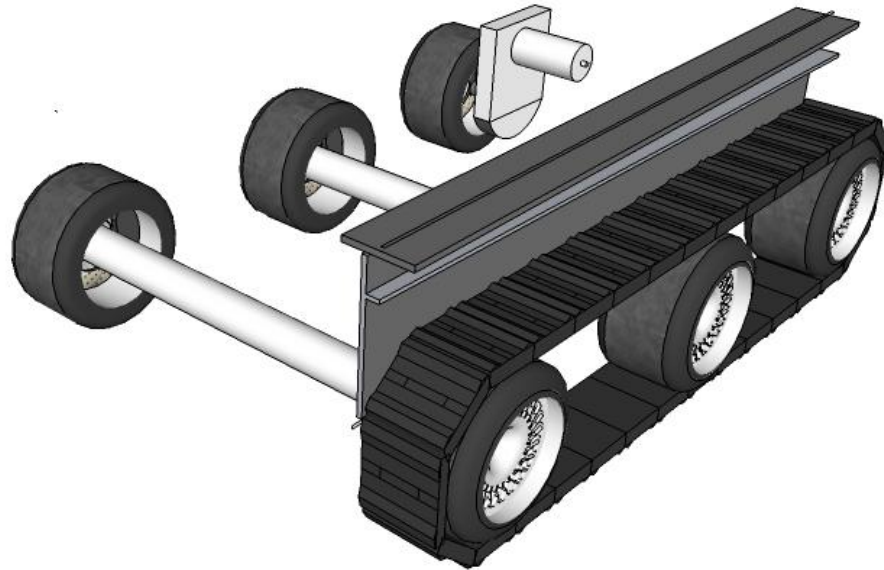


FIGURA 4.15: ACOPLAMIENTO DE LA ORUGA CON LAS LLANTAS

Una vez que se definió el tipo de tracción en el que se puede observar un par diferencial conectado a las llantas de la parte trasera y con la banda, se transmite este movimiento a las demás llantas del mismo lado, ahora solo queda fijar los laterales con la parte trasera, de esta forma, los motores quedan cubiertos por la estructura metálica.

La siguiente tarea a desarrollar, es colocar los sensores de tacto en el frente del robot, para esto, se consiguieron los siguientes switch de contacto, los cuales nos ayudan a saber si ha chocado con algo, y para poder alinearlos de frente a la bomba.

Considerando las dimensiones del objeto que tiene que alcanzar, este debe de estar por arriba de los 5 centímetros, midiendo a partir del suelo, así de esta forma, nos aseguramos que al chocar con el cubo, este no se salga de su base, de esta forma, el diseño final queda como se ve en la figura siguiente.

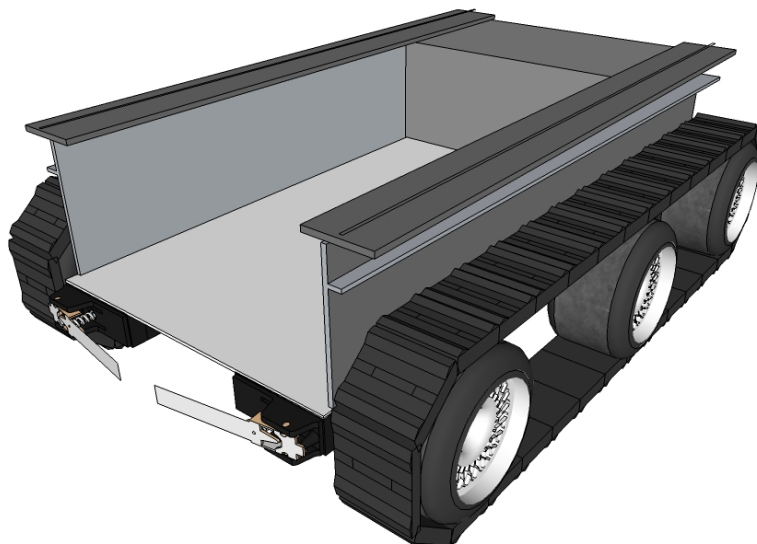


FIGURA 4.16: IMPLEMENTACIÓN SW DE CONTACTO

La siguiente parte fue la implementación de la fuente de energía (pilas) en el cuerpo, para la cual se utilizaron 8 pilas de laptop de Niquel-Metal, de 3.6 [V], a 1800[mA], conectadas por pares, en serie este arreglo nos proporciona un total de 14.4 [V], a 3600[mA], con una ramificación de 7.2 [V], a 3600[mA], como se muestra en el diagrama siguiente:

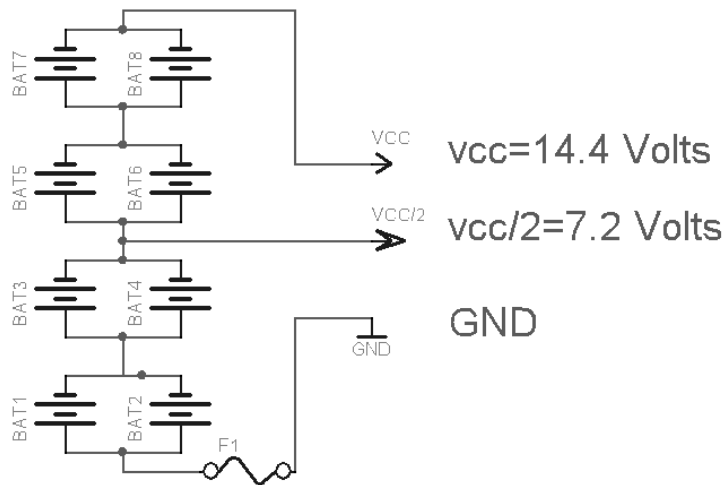


FIGURA 4.17: DIAGRAMA ELÉCTRICO DE LA CONEXIÓN DE LAS PILAS

Para alimentar los motores, se utilizó la fuente de energía de a 14.4 [V], y para la parte de las tarjetas electrónicas, como son la CMUCAM y el KITPIC así como los servomotores se ocupó la de 7.2 [V] en el siguiente dibujo se muestra como se colocaron las pilas dentro de la base.



FIGURA 4.18: IMPLEMENTACIÓN DE LAS PILAS EN EL ROBOT

4.2.2 IMPLEMENTACIÓN DE LAS TARJETAS EN EL CUERPO

Ya con la base terminada, proseguimos a colocar las tarjetas tanto de control como de potencia de los motores, para esto, requerimos de realizar agujeros en la parte trasera donde van los motores tratando que no queden muy separadas las tarjetas una de otra, de esta forma, nos aseguramos que la distancia de los cables no sea muy larga, ya que de lo contrario, podría causarnos

ruido y perturbaciones en la electrónica empleada, la tarjeta del Puente H, se colocó en la parte de abajo donde van los motores, los cuales están cubiertos por el mismo cuerpo, y la tarjeta Kit PIC se encuentra colocada en la parte trasera, una de las razones por las cuales la tarjeta de control se colocó así, es para facilitar el acceso a esta para programar y correr el programa con solo presionar un botón de la tarjeta, de este modo, nuestro sistema queda como se muestra a continuación.

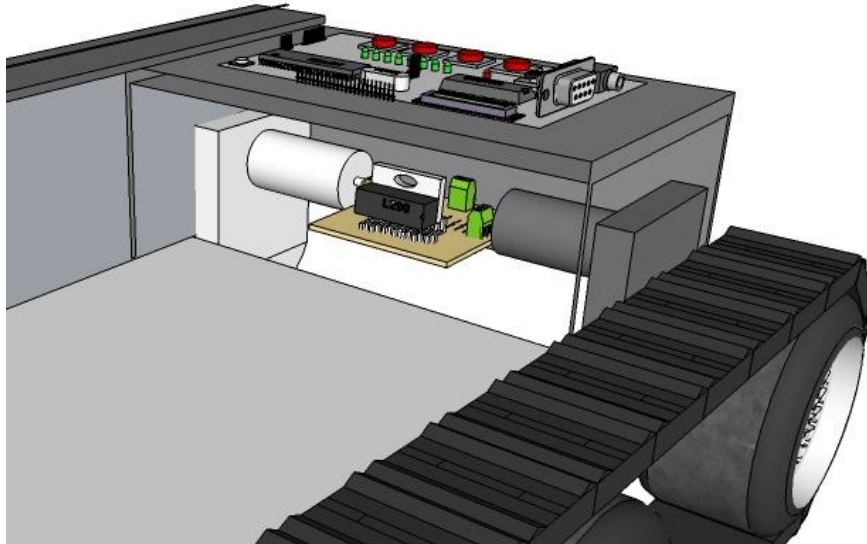


FIGURA 4.19: IMPLEMENTACIÓN DE LAS TARJETAS DE CONTROL EN EL ROBOT

4.2.2.1 CONEXIONES ELÉCTRICAS DE LA BASE MÓVIL

Ahora vamos a realizar las conexiones eléctricas de la base móvil, la cual se limita a la tarjeta de control, tarjeta de potencia para los motores (puente h) y los switches de contacto en el frente de la base.

En el siguiente cuadro, se puede ver el diagrama simplificado de conexiones de estos aditamentos:

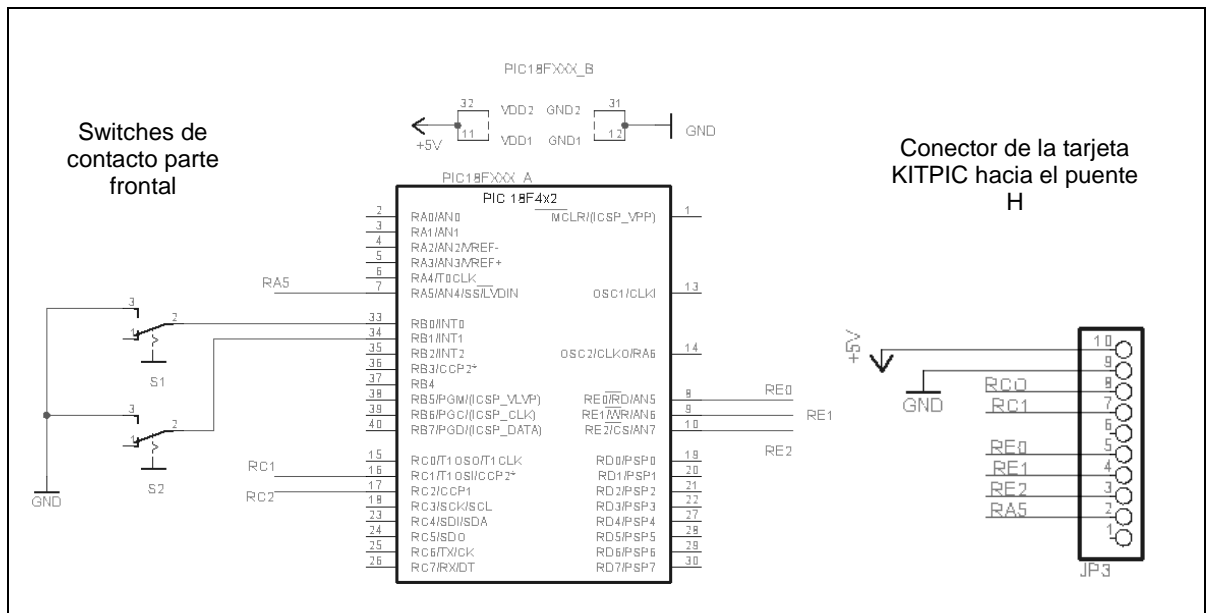


FIGURA 4.20: DIAGRAMA ELÉCTRICO DE LA BASE MÓVIL

Como se muestra en el diagrama anterior, la conexión con el puente h, se realiza mediante el conector de la derecha, de esta forma, queda perfectamente alineado con el conector del puente h, así, de esta forma, requerimos de una señal de PWM y dos de control por cada motor, de lo anterior, se deduce que RC0, RE0 y RE1 controlan el motor izquierdo mientras que RC1, RE2 y RA5 controlan el motor derecho de la base móvil.

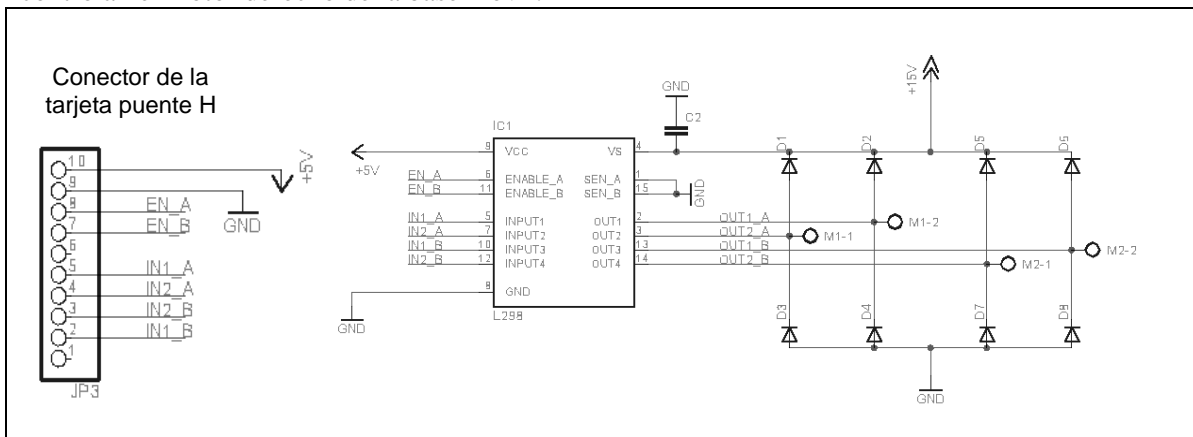


FIGURA 4.21: CONEXIONES ELÉCTRICAS DE LA ETAPA DE POTENCIA

De esta forma, los motores son conectados a las terminales M1 y M2 de la tarjeta de potencia

4.2.3 CONSTRUCCIÓN Y MONTAJE DEL MECANISMO DE LA CÁMARA Y EL GRIPPER

En esta etapa, integraremos la parte del movimiento de la cámara con el gripper, que es lo que hace posible encontrar los cables que se tienen que quitar del cubo amarillo, de esta manera, el robot desplaza este mecanismo de un lado a otro y hace un barrido completo de la cara del cubo, seleccionando un rango más exacto de la localización del cable.

Para esto, requerimos de un mecanismo que fuera capaz de moverse de un lado a otro por medio de un motor, este mecanismo, no es difícil de diseñar, ni mucho menos de construir, ya que su estructura es simple, sin embargo, uno de nuestros propósitos de este proyecto, es reciclar estructuras o mecanismos que se puedan encontrar en cualquier parte y así, bajar los costos en el diseño y construcción de este robot, por lo tanto, nos dimos a la tarea de buscar un sistema que satisficiera nuestras necesidades.

Encontramos que en los deshechos de equipo electrónico, existen muchos mecanismos de reproductores de cintas, de discos compactos, videograbadoras y un sin número de piezas de desecho, ahí, encontramos el mecanismo de un equipo reproductor de cd, el cual seleccionamos para nuestro propósito. En la foto siguiente se muestra el mecanismo empleado en el robot:



FIGURA 4.22: MECANISMO INTERNO DE UN REPRODUCTOR DE DISCOS COMPACTOS

A este sistema, tuvimos que hacerle unas adaptaciones para poder emplearlo de manera efectiva a nuestro robot, teniendo toda la precaución de no dañar los motores o la estructura principal, ya que podríamos causar un daño irreversible al sistema y tendríamos que conseguir otro, las adaptaciones que se hicieron son:

Antes que nada, veamos cómo funciona este mecanismo, como se puede ver en la figura (), este sistema consta de una base que va fija en una estructura ya sea metálica o plástica y una charola que es la que se desplaza a lo largo de los rieles por medio de un motor y un juego de engranes para aumentar el torque de este, de esta forma es capaz de recoger un disco en la charola e introducirlo al equipo de sonido.

Ahora, consideremos que para nuestro caso, lo que fijamos es la charola y dejamos libre la base a la que recortamos y solo dejamos la parte del motor y los engranes, de ésta forma, tenemos un sistema que puede correr de un lado a otro sobre un riel de plástico.

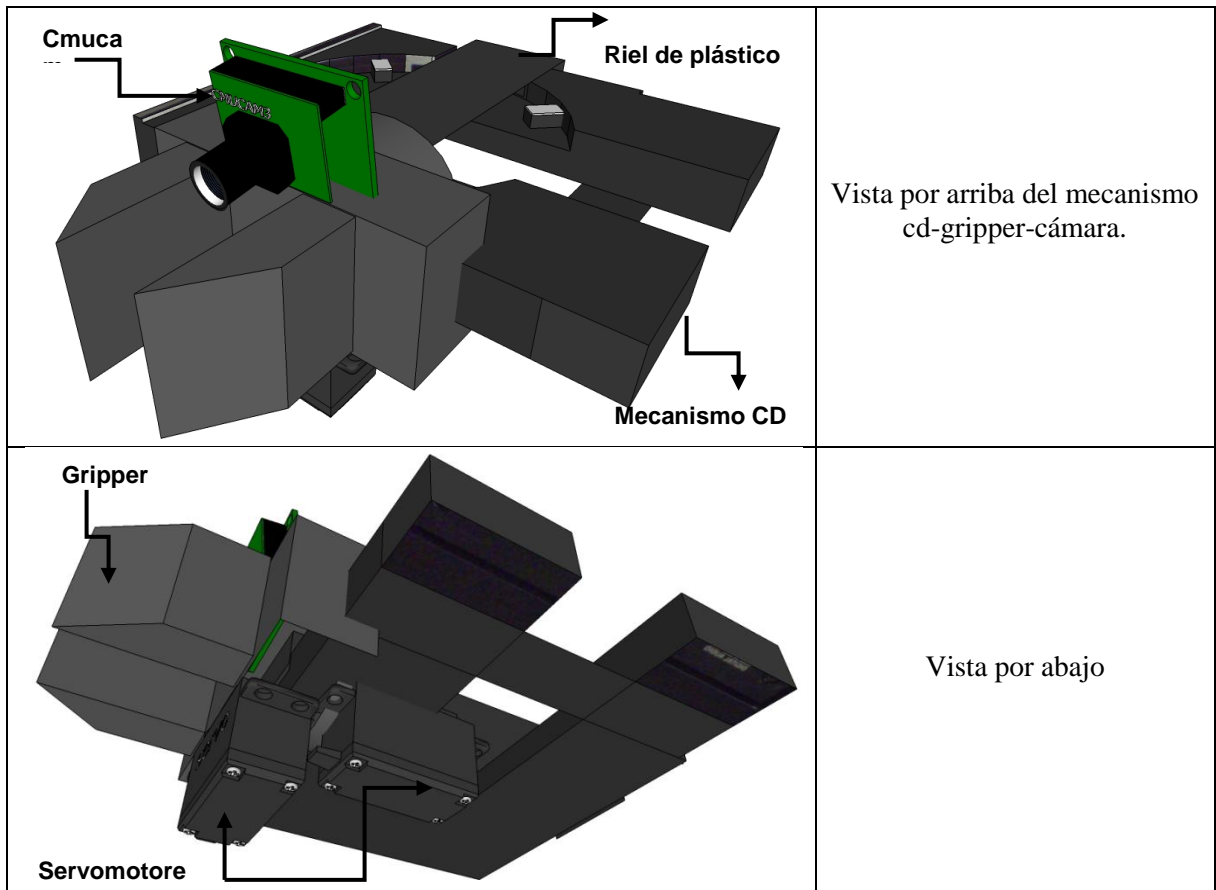


FIGURA 4.23: ADAPATACIÓN DEL MECANISMO CD Y GRPPER

Tomando en cuenta que ya tenemos el mecanismo de barrido del cubo, lo que sigue es montar la cámara y el gripper al sistema, para esto, requerimos de construir una base de plástico donde se montaran estos aditamentos, además de unos sensores de tacto que nos indicarán en qué posición se encuentra este sistema.

4.2.3.1 CONEXIONES ELÉCTRICAS DE LA CÁMARA Y EL GRIPPER

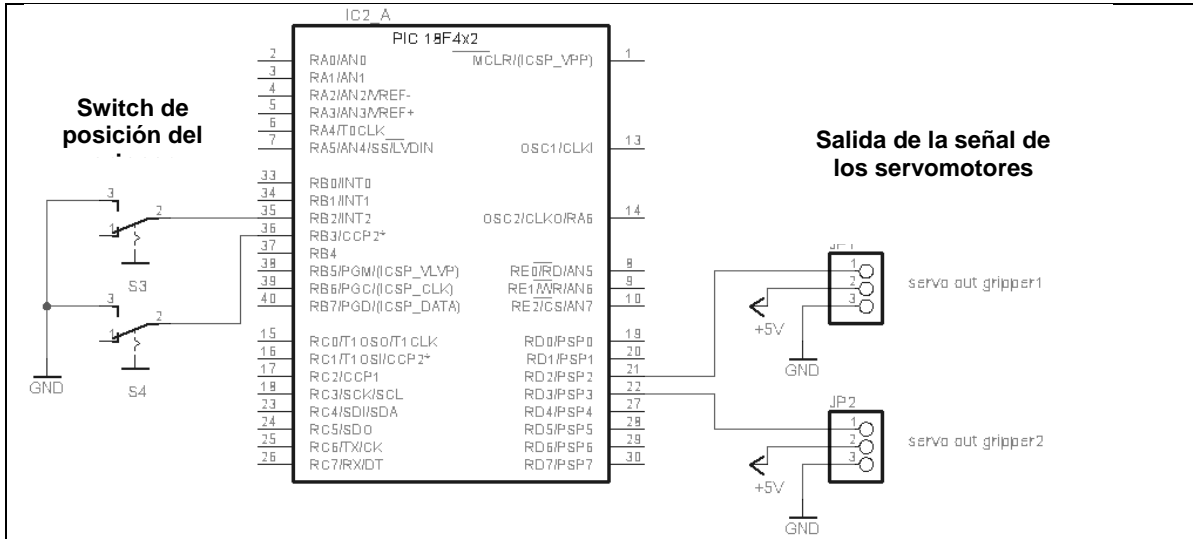


FIGURA 4.24: DIAGRAMA ELÉCTRICO DEL MOVIMIENTO DE LA CÁMARA

Como se puede ver, sus conexiones electricas son muy sencillas, ya que se ocupan servomotores que se encargan e posicionar el mecanismo dependiendo del pulso que se le envíe, de esta forma, solo requerimos de un pin de salida digital para poder controlarlo, a su vez, requerimos de dos switches que nos indican si el sistema se encuentra en uno de los extremos.

Ademas a este mecanismo, le montamos la camara de vision para detectar los colores que se tienen indicados en las reglas del concurso

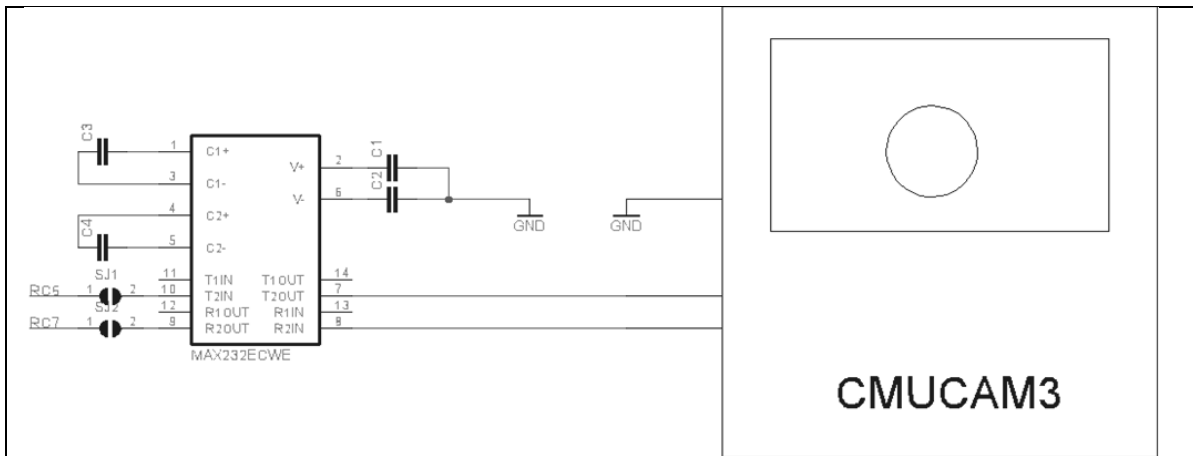


FIGURA 4.25: DIAGRAMA ELÉCTRICO DEL SISTEMA DE VISIÓN CMUCAM 3

De esta forma, nuestro mecanismo terminado, puede montarse directamente a la base móvil fijándola por ambos lados de la estructura metálica y considerando las dimensiones de altura que son requeridas para la tarea específica.

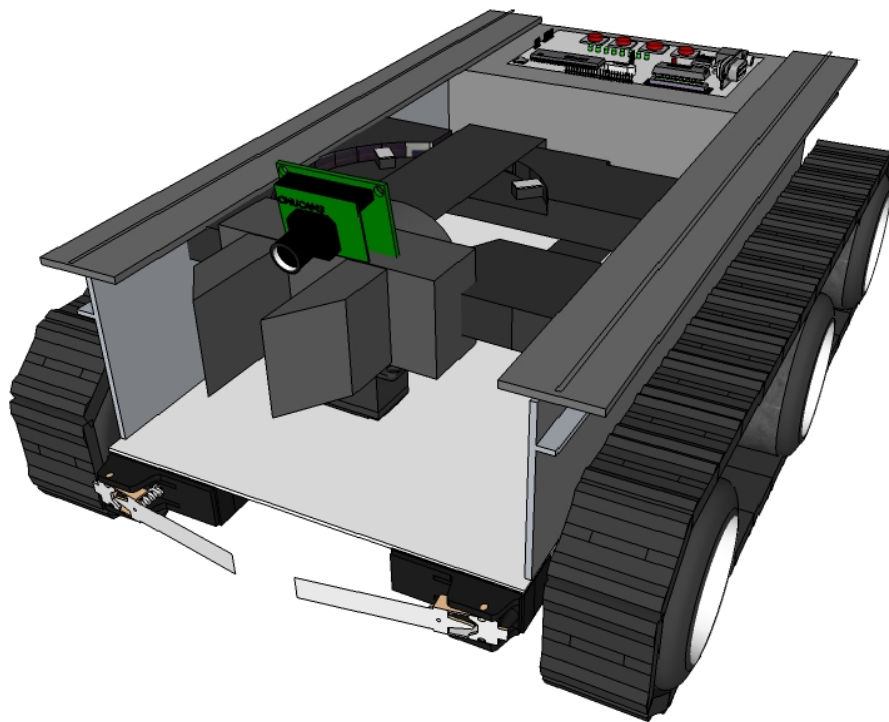


FIGURA 4.26: MECANISMO TERMINADO

4.2.4 CONSTRUCCIÓN Y MONTAJE DEL BRAZO MECÁNICO

Como ya hemos visto en otro capítulo, existen una gran variedad de brazos mecánicos que se clasifican por el tipo de movimiento que emplea, es decir, los grados de libertad con los que cuentan, para este proyecto de tesis, no se requiere un brazo que sea muy complejo, así que se pensó en un brazo solo con 2 grados de libertad que fuera capaz de tomar el cubo y poder girarlo sobre su eje de rotación.

El brazo mecánico que se diseño, consta de un mecanismo en forma de pinza con un servomotor que se encarga de girar el cubo, y de un motor de corriente directa que se encarga de posicionar dicho mecanismo.

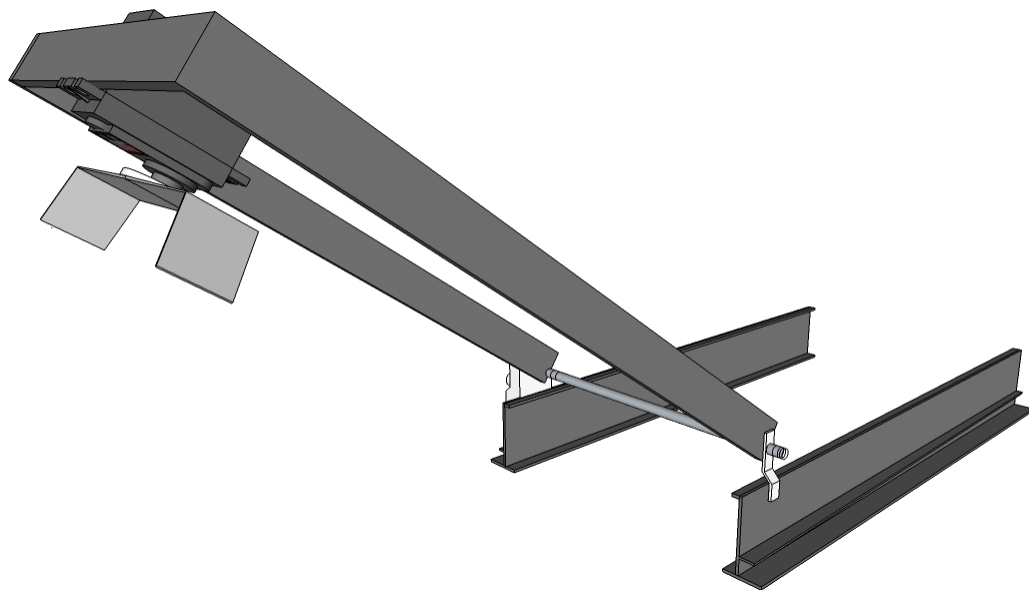


FIGURA 4.27: MONTAJE DEL BRAZO

Las características de los materiales empleados, son los siguientes:

Moto reductor de corriente directa a 24 volts, a 1.85 rpm.
 Servomotor hitec 311
 Estructura de aluminio.

Como podemos ver, este brazo emplea un moto reductor muy lento (1.85 RPM) lo que nos indica que su torque es muy alto, esto es de gran ayuda, ya que debe levantar desde su base, todo el brazo completo y el peso del servomotor.

4.2.4.1 CONEXIONES ELÉCTRICAS DEL BRAZO MECÁNICO

Su diagrama eléctrico es el siguiente:

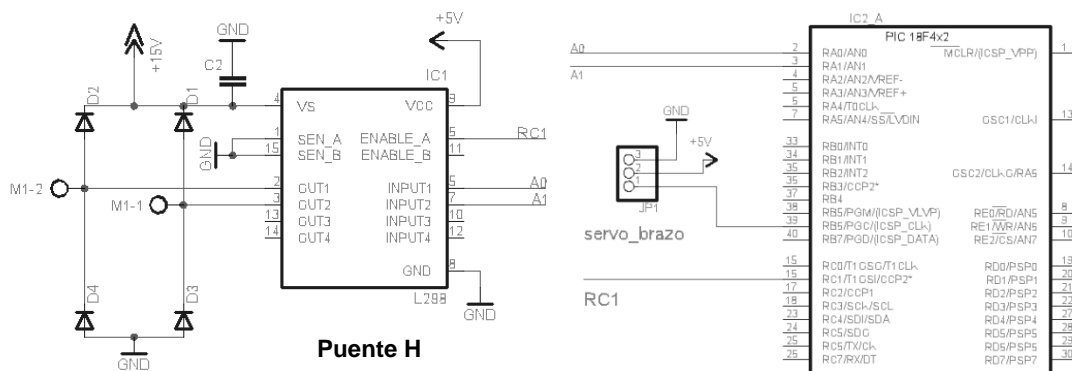


FIGURA 4.28: DIAGRAMA ELÉCTRICO DEL BRAZO MECÁNICO

4.3 ESTRUCTURA TERMINADA

Una vez que se implementaron todos los elementos anteriores el robot queda finalmente como se muestra a continuación:

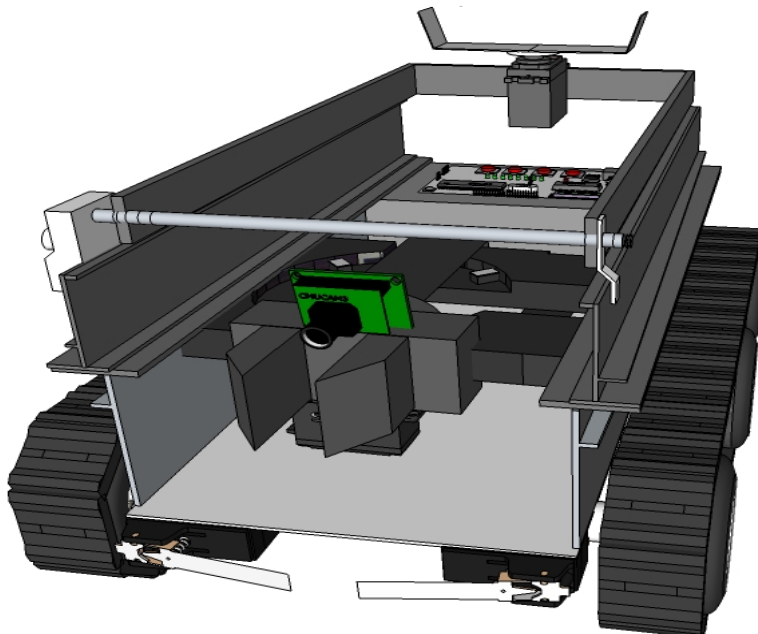


FIGURA 4.29: ESTRUCTURA TERMINADA VISTA 1

En la siguiente figura se muestra el robot con el brazo extendido:

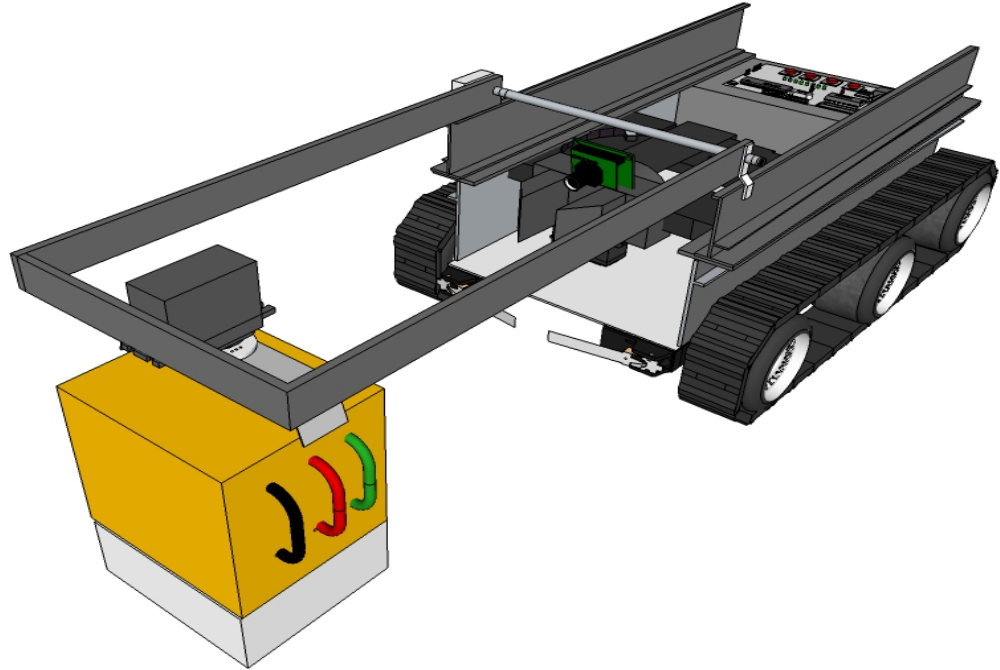


FIGURA 4.30-A: ESTRUCTURA TERMINADA VISTA 2



FIGURA 4.30-B: ESTRUCTURA TERMINADA ROBOT REAL

4.4 IMPLEMENTACIÓN DEL SOFTWARE

El software que se implemento para este proyecto, consta de dos módulos ya que casi toda la inteligencia del movimiento del robot, está contenida en el PIC18F452, y la otra parte se encuentra en el sistema embebido de la tarjeta de visión.

La comunicación que se tiene de la cámara y el PIC, es bajo el protocolo rs232, aunque también es posible la comunicación con niveles lógicos de TTL, esto se desarrollo así, ya que la tarjeta de control, ya tiene integrada esta función ya que de lo contrario tendríamos que estar conectando y desconectando este puerto del PIC para reprogramar la tarjeta de control.

4.4.1 MOVIMIENTO DEL ROBOT

El robot fue programado como una máquina secuencial es decir que nuestras respuestas o salidas van a depender de sus sensores, además de los estados anteriores, lo que va a permitir realizar una serie de operaciones una tras otra en el tiempo.

Como se menciona el PIC es el cerebro principal, por tanto es quien decide cuando mandar las señales para que el robot se mueva, y siempre para tomar estas decisiones está en constante comunicación con el sistema de visión preguntándole que color está viendo, según le interese al PIC.

4.4.1.1 DIAGRAMA DE FLUJO

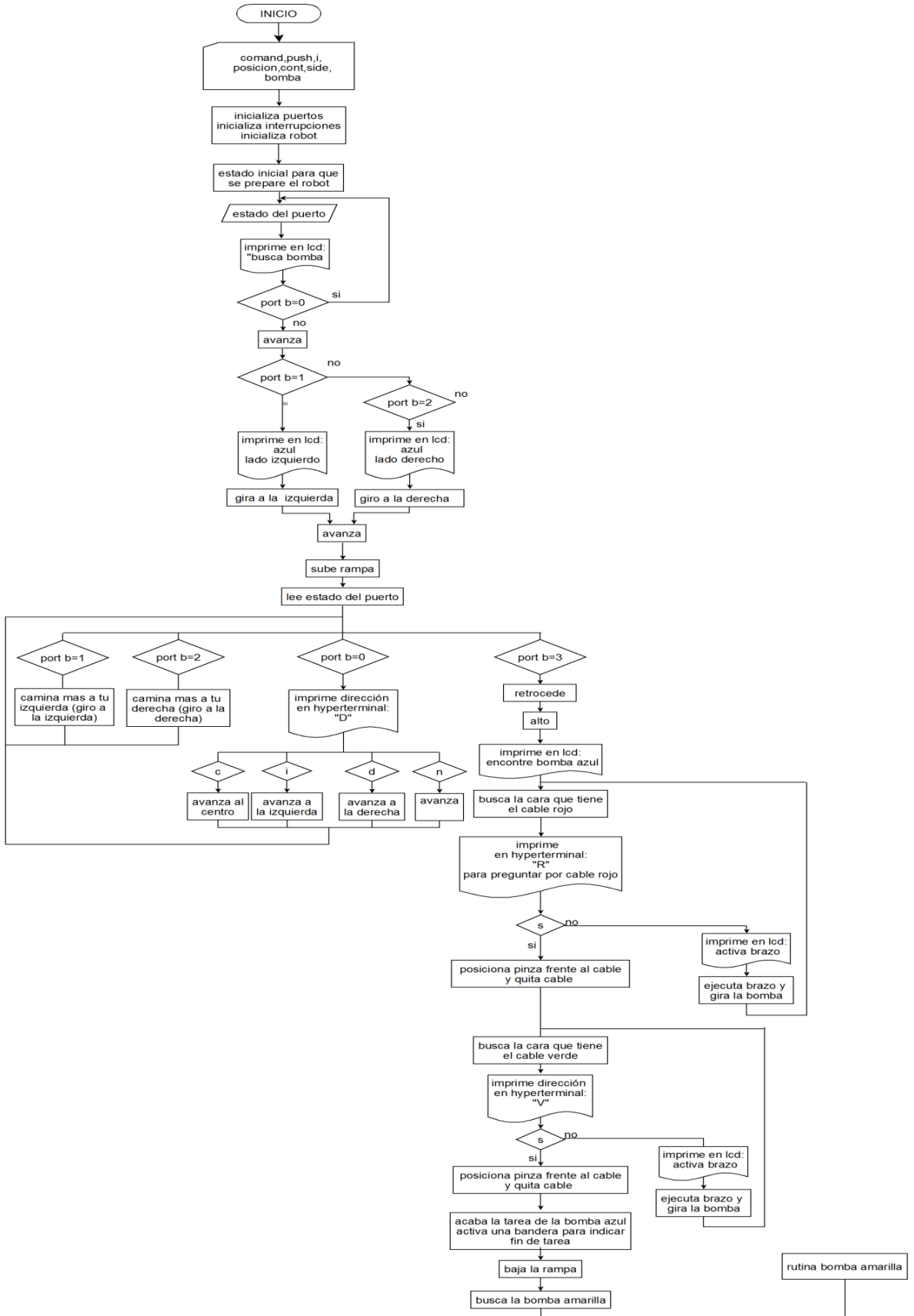


FIGURA 4.31: DIAGRAMA DE FLUJO BOMBA AZUL

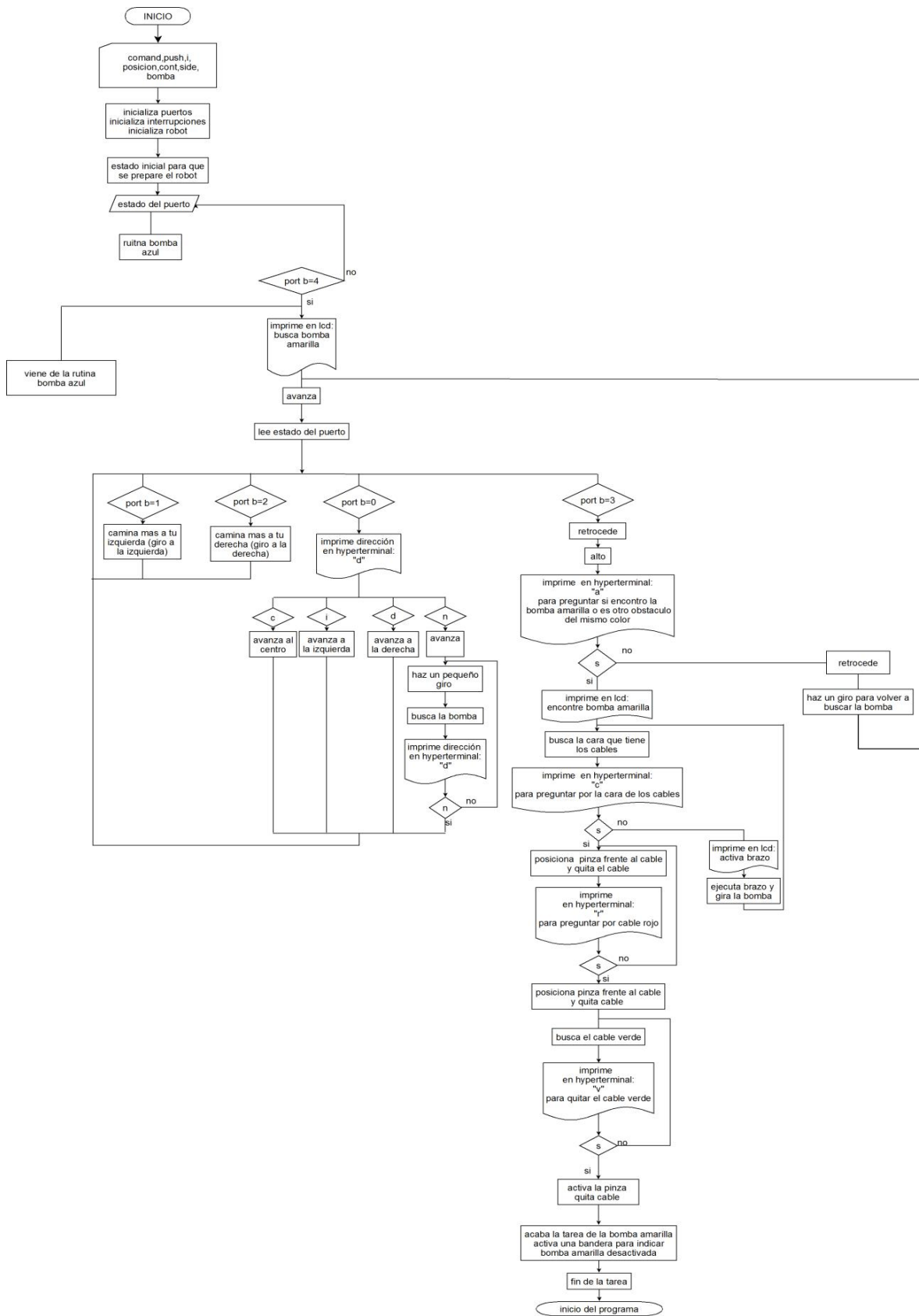


FIGURA 4.32: DIAGRAMA DE FLUJO BOMBA AMARILLA

4.4.1.2 EXPLICACIÓN DEL SOFTWARE MOVIMIENTO DEL ROBOT

El programa está hecho para que por medio de los botones que tiene la tarjeta el robot se pueda ir sólo por la bomba amarilla, o por la azul o ambas esto se hizo así por estrategia en la competencia, y acumular el mayor número de puntos en el menor tiempo, donde el código lo podemos consultar en la sección de apéndices llamado movimiento del robot.

La idea general para ambas bombas es primero buscar la bomba a través de su color ya sea azul o amarilla, y caminar a hacia ella una vez que la haya localizado, tratando de alinearse al centro, es decir que cuando llegue a ella estará colocado en frente y al centro de alguna de las caras del cubo.

Una vez hecho esto trata de localizar la cara que contiene los cables, sino llego enfrente de la cara de los cables, utiliza su brazo para girarla hasta encontrarla, posteriormente empieza a hacer un barrido para colocar la pinza en exactamente en frente de los cables primero localiza el rojo y luego el verde, una vez que los localizo, los va a quitar por medio de la pinza haciendo tantos intentos sean necesarios para asegurarse que los quitó, esto se hace porque puede que el robot no esté tan cerca y no logre quitar bien el cable y eso no cuenta, ya que tiene que quitar por completo el cable, y así se asegura que lo haya quitado bien.

El programa empieza con definiciones del PIC aquí se va a poner la librería del Pic con que vamos a trabajar, los fusibles en este caso ocupamos HS (High Speed Crystal. Se suele usar con cristales de más de 4MHz), NOWDT (No WatchDog. Desactiva el perro guardián), NOPROTECT (No proteger el código. Permite que pueda leerse el programa), NOVLP (No Low Voltage Program. Impide que el micro pueda grabarse en modo de bajo voltaje). Así como el valor del reloj con el que vamos a trabajar, la comunicación que vamos a usar incluyendo la velocidad de baudios y los pines de transmisión y recepción.

En seguida definimos el área de reserva para el bootloader, el que ocupamos para programar el circuito, sin necesidad de tener físicamente un programador. Posteriormente tenemos un área llamada include porque aquí ponemos las librerías que vamos a usar entre ellas la del lcd y una que se creó con puras funciones llamada <chacharas_brazo.c>. Así como la zona de define.

En seguida habilitamos las resistencias de pullup, esto es porque el PIC, ya trae integradas resistencias aterrizadas a V+, de lo contrario, tendríamos que modificar el hardware y sería más costoso. Además mandamos a deshabilitar o se pone en off el puerto analógico que en este caso no ocupamos, y en la siguiente línea se inicializa el puerto b.

Ya en el programa principal, damos de alta las variables que vamos a ocupar durante el programa, posteriormente antes de entrar a las rutinas principales mandamos llamar a las funciones del lcd para poder utilizarlo, así como las que inicializan al robot(init_robot) la cual prepara al robot para que se alinee el brazo, la pinza y la cámara en el centro, esto se hace para asegurar que antes de empezar todo este en su posición y no lleguen mal porque al girar el cubo y quitar los cable se hacen con movimientos muy precisos.

Después entra a un while que es el que contiene las dos rutinas de las dos bombas, en donde se hace una variable llamada push igual al valor del puerto que hace referencia a los 4 botones de la tarjeta, los cuales pueden ser 1,2,4 y por default cero cuando no se ha presionado nada, con el valor de uno en el puerto va a subir por la bomba azul por el lado izquierdo de la pista, con el valor de dos va a subir por el lado derecho, con el valor de cuatro va a ir por la bomba amarilla en l parte de abajo, y si no es cualquiera de estos valores va a estar esperando que se presione cualquiera de estos para seguir. Una vez que se presiono manda a imprimir al lcd el color de la bomba, cabe mencionar que

se ocupa el lcd y los leds para mandar mensajes y así poder ver en que parte del programa esta y así saber si falla algo.

Hemos dividido en dos rutinas el diagrama de flujo en dos partes la rutina de la bomba azul y la bomba amarilla.

Rutina bomba azul:

Una vez que ya sabe que va a ir por la azul, gira a la izquierda o la derecha dependiendo el botón que se ha presionado, con esos giros se logra que quede de frente a la rampa, y sube, alineándose para llegar lo más al centro de la bomba, como se explicó en la sección de arquitectura, el robot cuenta con switches de tacto en la parte de enfrente, que van a permitir saber cuándo choque con la bomba u otro obstáculo; cuando estos switches no están presionados el robot camina y a la vez trata de caminar lo más derecho posible para llegar justo en frente de ella y en medio, esta alienación se lo indica el sistema de visión mandándole los centroides del color de la bomba, imprimiendo un carácter, “I” de izquierda, “C”, centro y “D” de derecha, en caso de que por algún motivo ya no la vea, manda una “n”, cuando uno de los switches es presionado el robot hace un giro para tratar de que el otro haga contacto, esto se hace porque seguramente el robot ya llevo a la bomba pero si no tocaron los dos switches al mismo tiempo es porque llevo un poco chueco y así lo hacemos que se alinee, hasta que se indique que los dos han hecho contacto.

Cuando hace esa rutina, ya estamos seguros de que quedo enfrente y exactamente en medio, y hacemos que retroceda un poco para ampliar el rango de visión, entonces el PIC le pregunta a la CMUCAM, si ve el cable rojo en esa cara del cubo, mandando el caracter “R”, si le contesta con una “y” indica el cable esta dentro de esa cara, si contesta con “n”, se manda a llamar una función la cual se llama “ejecuta_brazo”, donde se activa el brazo y hace un giro a la bomba, hasta encontrar la cara que contenga el cable rojo siguiendo el mismo protocolo de comunicación, mandando R para preguntar y para contestar “y” o “n”. Cuando ya sabemos en qué cara esta el cable empezamos a hacer una búsqueda más detallada con la cámara a lo ancho de la bomba, por medio de la función “quita_cables”, lo que hace esta función es hacer un barrido, recorriendo poco a poco la bomba, hasta situarse exactamente en medio del cable rojo y manda a activar el gripper(pinza) y hace que jale el cable para poder quitarlo, está rutina de quitar el cable se repite tantas veces sea necesario hasta que lo haya quitado preguntando si sigue viendo el cable, una vez que ya no lo vea es porque lo ha quitado con éxito. La siguiente tarea, es quitar el cable verde, la rutina es la misma que se empleo para el cable rojo, una vez que quitó el cable verde se activa una bandera para pasar a la rutina de la bomba amarilla, posteriormente baja la rampa de espaldas, y dependiendo de qué lado subió hace un giro para colocarse de frente a la pista.

Rutina bomba amarilla:

Esta rutina es un poco similar a la de la búsqueda de la bomba azul, solo que aquí tiene un poco más de espacio para la búsqueda y así realizar más movimientos, así como encontrarse obstáculos como las paredes de la pista, y tiene más dificultad en su sistema de visión porque como las paredes no son tan altas, puede que lo que está afuera contengan colores similares a los que se están buscando y pueda crear un poco de confusión.

Lo primero que hace es caminar ya sea en cualquiera de la entradas de la pista (recordando que esto se logra apretando el botón numero tres de la tarjeta kitpic para saltarse la rutina de la bomba azul si es que así se eligió empezar), o en la posición que se quedo al bajar por la rampa, camina aproximadamente unos 40 [cm]. Y se vuelve a ocupar la rutina de los switches de contacto, mientras no se active ninguno el robot camina y hace la búsqueda de la bomba, y trata de dirigirse a ella, tratando de llegar lo más centrado posible, hasta que hagan contacto los dos switches, una vez que se haya avisado que hicieron contacto los dos, se le pregunta a la cámara si con lo que ha chocado es en verdad la bomba, esto se pregunta porque por las variaciones de luz o si llega a haber

algo afuera de la pista del mismo color que la bomba cabe la posibilidad de que se haya confundido, y con esto aseguramos de que en verdad haya llegado a la bomba, así que su protocolo de comunicación con la CMUCAM, es que el pic le mande un carácter “a”, y si es la bomba la cámara responderá devolviéndole una “y”, si es la bomba y si no manda una “n”, (este paso se omitió en la bomba azul porque cuando sube y choca con algo es seguro que sea la bomba, por el espacio tan limitado que se tiene además de que no hay ninguna pared), si recibe una “n”, quiere decir que se encontró con otra cosa, cabe mencionar que en esta pregunta se hace un filtrado o se acota el rango de color amarillo para poder diferenciar un obstáculo de la bomba, por lo que retrocede y hace un giro para seguir buscándola y al caminar va preguntando si la ve y en qué dirección se encuentra, como al principio. Ya que obtuvo una “y”, al encontrar la bomba, retrocede unos centímetro para ampliar su rango de visión y así poder preguntar si la cara que tiene enfrente contiene los cables verde y rojo, donde el pic manda una “c”, obteniendo de respuesta de la cámara una “y” en caso afirmativo y una “n” en caso negativo, si obtiene una “n”, mandara a llamar a la función que activa el brazo para poder girarla hasta encontrar la cara de los cables, es decir, hasta que se obtenga una “y”, posteriormente hace el mismo barrido que se explico en la bomba azul, con la cámara para localizar tanto el cable rojo como el verde, y poder quitarlos y dar por desactivada la bomba amarilla, y finalmente se termina con la tarea.

4.4.1.4 EXPLICACIÓN DE LA LIBRERÍA CREADA PARA EL ROBOT

Esta librería la creamos para tener en un solo archivo de todas las funciones que necesitamos llamar, a lo largo del programa y cómo podemos observar son las mismas para las dos rutinas, de las bombas azul y amarilla, el código lo podemos encontrar en la zona de apéndices llamado librería del robot.

Podemos observar las siguientes funciones dentro de la librería:

init_robot (): En esta función iniciamos el robot y habilitamos la parte del pwm.

mv_robot (): En esta función se determinan las velocidades y las direcciones de las llantas del robot, así como el freno rápido que se hace para mayor precisión de movimiento.

mv_brazo (): esta función prácticamente hace lo mismo que la anterior pero está se refiere a los motores que ocupa el brazo y podemos ver que los pines de dirección no son los mismos.

atras (): esta función manda a llamar a la función mv_robot y se creó para hacer caminar el robot hacia atrás.

adelante (): esta función manda a llamar a la función mv_robot y se creó para mover el robot hacia adelante.

centro (): esta función se creó para poder alinear en el centro la pinza que hace girar la bomba, cuando se prenda el robot o inicie una rutina, ya que el movimiento lo logramos con un servomotor.

levanta_brazo (): esta función manda a llamar la función mv_brazo(), le da los pines de dirección y manda a mover el motor del brazo para que lo levante.

baja_brazo (): esta función manda a llamar la función mv_brazo(), le da los pines de dirección y manda a mover el motor del brazo para que lo baje.

ejecuta_brazo (): esta función manda a llamar a las dos anteriores en una sola rutina para posteriormente girarla, mandando a llamar la función gira().

gira (): la función se encarga de hacer girar la pinza, es similar a la de centro, pero esta tiene un rango más amplio para poder girar el servomotor, la función necesita el valor de la variable side para decidir a qué lado girar si izquierda o derecha, esto se vuelve importante porque cuando se va a la bomba azul, dependiendo del valor de side hará solo 2 giros en vez de cuatro y esto ahorra tiempo, porque los cables solo estarán colocados en 2 posiciones fijas que es por el lado de las dos rampas.

center_posicion_servo (): en esta función lo que hace es centrar la pinza o el gripper que quita los cables en el centro del robot.

center_pos_izq (): coloca la pinza o gripper a la izquierda del robot, la pinza sabe que ya llego a la izquierda porque hace contacto con un switch conectado a b2.

PWM_GENERATION_SERVO2 (): es una rutina que nos permite utilizar los servos de la pinza para abrirla o cerarla

ejecuta_gripper (): acerca al robot al cable, manda a llamar a PWM_GENERATION_SERVO2, primero para cerrar la pinza y quitar el cable, haciendo retroceder el robot, y después se manda a abrir el gripper para que lo suelte.

quita_cables (): esta rutina se encarga de centrar la pinza y empezar con el barrido para localizar el cable rojo o verde, hasta que reciba una “y” manda a llamar la función de ejecuta_gripper para quitarlo, en esta función se asegura de haberlo quitado por completo preguntando si lo sigue viendo se sigue ejecutando la función de ejecuta_gripper hasta que la cámara le mande una “n”.

4.4.2 SOFTWARE DEL SISTEMA DE VISIÓN

En el sistema de visión se ocupó la cámara CMUCAM, con la cual lo que se hace es hacer un reconocimiento de los colores que se van a usar, generalmente se le llama trackeo de color.

El siguiente esquema muestra el procedimiento para calibrar o trackear un color:

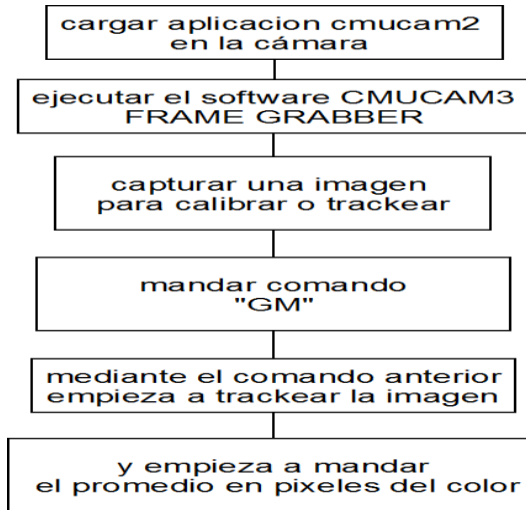


FIGURA 4.33: ESQUEMA DE CALIBRACIÓN CMUCAM

El procedimiento que se sigue para calibrar o trackear un color es por medio de la aplicación llamada cmucam2, que vienen en la carpeta projects del cc3, esta aplicación viene en forma de programa el cual se graba a la cmucam, una vez grabado, se abre el programa “cmucam3 frame grabber” el cual es un ejecutable por lo que no hay necesidad de instalar, y tenemos una pantalla como la siguiente:

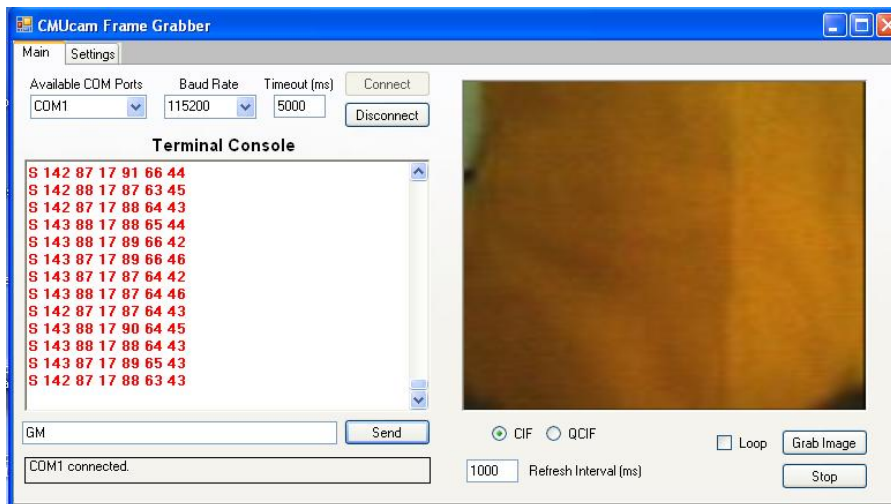


FIGURA 4.34: SOFTWARE DE CALIBRACIÓN CMUCAM

En la cual el programa cmucam 2 está programado para ocupar una velocidad de baudios de 115200, así que esa es la que vamos a seleccionar y también seleccionamos el puerto de comunicaciones con el que estamos trabajando, y le damos conectar, al conectar podemos ver en la pantalla la imagen que está viendo en ese momento la cámara, esta pantalla es muy útil porque así podemos centrar o mover el objeto que vamos a calibrar, a la distancia que nosotros queremos. Una vez hecho este ajuste tomamos una especie de foto del objeto dándole un click al botón “Grab image”, en este caso pusimos frente a la cámara una tela amarilla.

Aquí tratamos que el objeto abarque casi todo la pantalla, y ahora para que nos empiece a trackear mandamos en la línea de comandos “GM”, y nos empieza a desplegar el promedio de los valores RGB de la imagen, estos valores que nos despliegue no siempre van a ser los mismos pero se toma el valor que más se repite, de estos valores para determinar el rango mínimo y máximo se le tiene que quitar dos unidades para arriba y para abajo, esto recomienda el manual de la cmucam, pero se pueden bajar y subir tantas unidades sean necesarias para tener un valor más grande para que caiga ese color, solo que hay que tener cuidado si se ocupan diferentes colores no vayan a caer dentro ese mismo rango.

Por ejemplo si se nos presenta en pantalla para el color amarillo un promedio de 142, 87, 17

Quiere decir que el amarillo tiene un valor en la componente “R” de 142, en la componente “G” de 87 y en la componente “B” de 17.

Por lo que quitándole 2 unidades y agregándole dos unidades las componentes mínimas y máximas quedarían así:

$$R_{\text{mínima}} = 140$$

$$R_{\text{máxima}} = 144$$

$$G_{\text{mínima}} = 85$$

$$G_{\text{máxima}} = 87$$

$$B_{\text{mínima}} = 15$$

$$B_{\text{máxima}} = 19$$

Una vez obtenidos estos valores se tienen que poner en el programa que se ocupa para la comunicación con el Pic. Este mismo procedimiento se hace para todos los colores que se ocuparon durante la competencia como son: rojo, azul verde y amarillo.

Cabe mencionar que esta calibración se tiene que hacer constantemente, por las variaciones de luz que se presentan durante el día y es recomendable sacar un promedio del color a calibrar con mucha luz, poca luz, con sombra, etc. También es importante resaltar que el cable verde y rojo toman diferentes valores sobre las bombas azul y amarilla, por lo que se recomienda no poner los mismos valores.

4.4.2.1 DIAGRAMA DE FLUJO SISTEMA DE VISIÓN

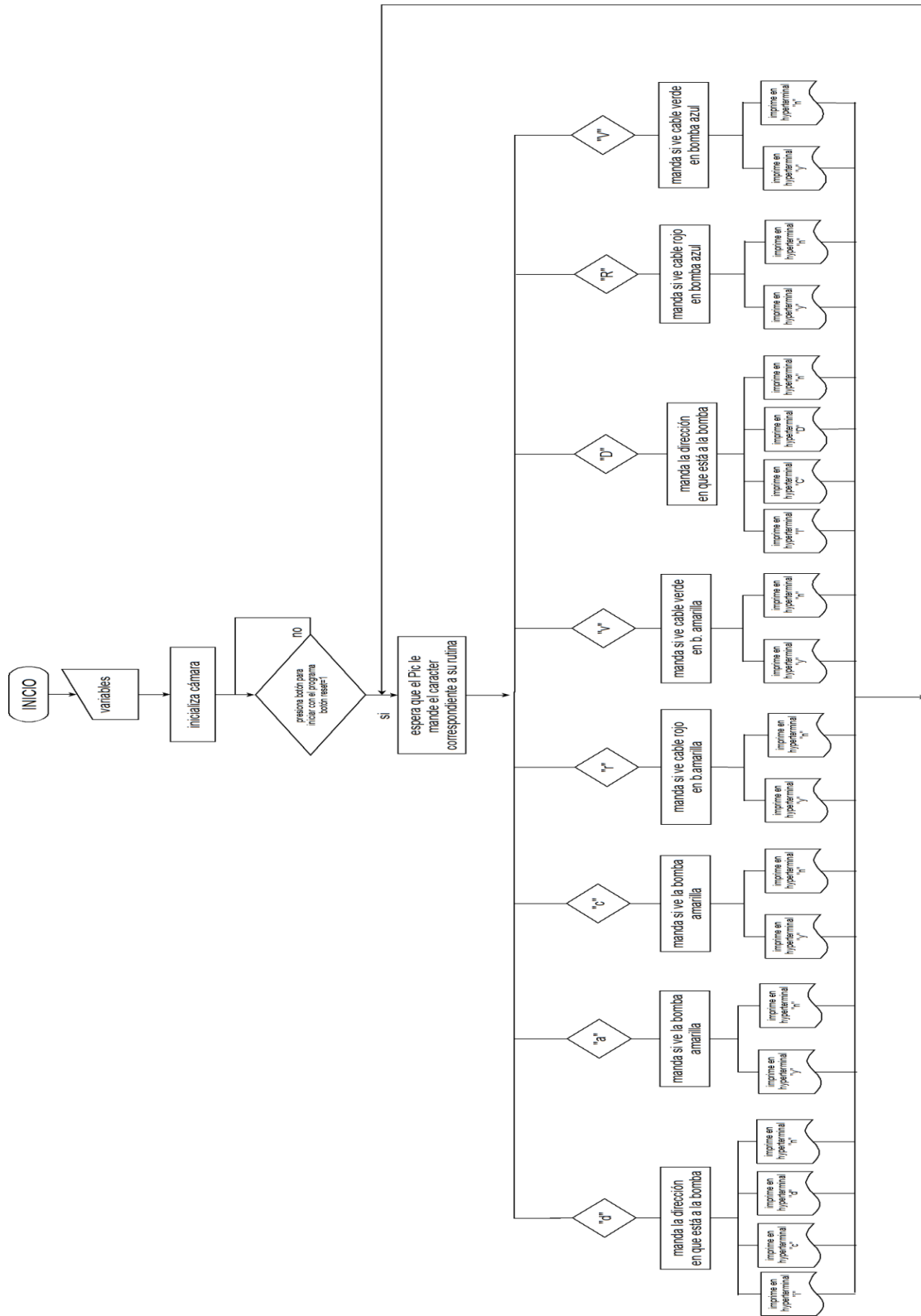


FIGURA 4.35: DIAGRAMA DE FLUJO SISTEMA DE VISIÓN

4.4.2.2 EXPLICACIÓN DEL SOFTWARE DEL SISTEMA DE VISIÓN

Este programa se basa en una aplicación que trae la cmucam, el cual lo que hace es estar en constante comunicación con el PIC, contestándole si ve un color y en caso de ser necesario indica la dirección, el código lo podemos encontrar en el apéndice llamado código del sistema de visión.

La forma que hemos adoptado para que pueda responder lo que pregunta el Pic, es primero asignándole los valores en RGB, máximo y mínimo a cada color que hemos obtenido al trackearlos con el programa cmucam2, de la siguiente forma:

Amarillo:

```
t_pkt.lower_bound.channel[CC3_CHANNEL_RED] = 80;  
t_pkt.upper_bound.channel[CC3_CHANNEL_RED] = 167;  
t_pkt.lower_bound.channel[CC3_CHANNEL_GREEN] =85;  
t_pkt.upper_bound.channel[CC3_CHANNEL_GREEN]=165;  
t_pkt.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;  
t_pkt.upper_bound.channel[CC3_CHANNEL_BLUE] = 16;
```

Donde indicamos primero los valores de rojo en mínimo y máximo, y así también para el verde y el azul, que forman el color amarillo (t_pkt) de la bomba, así indicamos todos los valores en RGB para el cable rojo con fondo amarillo (t_pkt2), el cable verde con fondo amarillo (t_pkt3), la bomba azul (t_pkt6), el cable rojo con fondo azul (t_pkt4), y verde con fondo azul (t_pkt5).

En el programa principal se hizo un switch que es donde entran todos los caracteres, que nos manda el pic para preguntar, en el caso de que entra al caso “d” es porque el pic quiere que le mande la dirección a la que está viendo la bomba ya sea a la izquierda, derecha o centro. Estas direcciones las determinamos dividiendo en tres cuadrantes la imagen que trackeamos por ejemplo:

Supongamos que el robot esta frente a la bomba amarilla, por lo que solo va a tener en su rango de visión la cara del cubo, y le queremos preguntar en qué dirección se encuentra exactamente, y la cámara nos tendrá que responder si a la derecha, izquierda o centro.

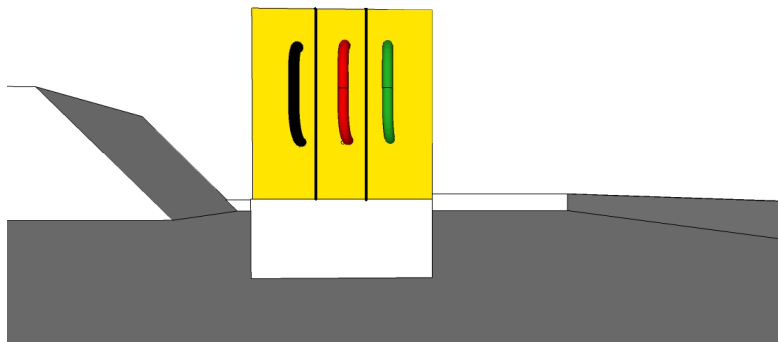


FIGURA 4.36: IMAGEN DIVIDIDA EN TRES

Lo que hacemos es dividir la imagen(figura 4.37 que está viendo la cmucam, en tres cuadrantes, esta imagen le asignamos un tamaño que viene sugerido en las aplicaciones, que va de 0- 174 (píxeles), estos 174 píxeles los dividimos entre tres de tal forma que a su izquierda el rango será de 0-58, para el centro será de 58-116, y a la derecha de 116- 174; pero en el programa tratamos de cerrar más los rangos para que sea aun más precisa la dirección, dichos valores se asociaran a la función centroide.

Por ejemplo si queremos que la cámara pueda responder si ve el color amarilla a su izquierda, al centro o a la derecha la función se representa de la siguiente forma:

(t_pkt6.centroid_x>0 && t_pkt.centroid_x<=58) para el lado izquierdo (i)

(t_pkt6.centroid_x>58 && t_pkt.centroid_x<=116) para el centro (c)

(t_pkt6.centroid_x>116 && t_pkt.centroid_x<=174) para el lado derecho (d)

Donde (t_pkt6) es ocupado para el color azul, en nuestro programa. Los valores en que se divide la imagen pueden ampliarse o reducirse, dependiendo el rango de visión que necesitemos.

CAPÍTULO 5

PRUEBAS Y RESULTADOS

En este capítulo se muestran las pruebas que se hicieron de cada uno de los sistemas así como los resultados obtenidos de cada una de ellas.

5.1 PRUEBA DEL MOVIMIENTO DE LA BASE MOVIL

La primera prueba que se realizó fue el movimiento de la base móvil, para esto realizamos un programa sencillo el cual podemos ver en el apéndice D, que sólo involucra las velocidades y las direcciones que se emplean en el robot.

Para esto requerimos montar una computadora a la base, y por medio de una comunicación remota, se le daban comandos para indicarle el movimiento y la velocidad a la que queríamos mover la base, con esto determinamos la velocidad máxima y la velocidad óptima para el desempeño del robot.

Teóricamente los motores trabajan a 101 rpm, a 24 [V], si este fuera el voltaje que se le suministra al robot y con una llanta de 8 [cm] de diámetro, la velocidad máxima estaría dada por la función:

$$V_{\max} = N \text{ rpm}(2\pi*r)$$

DONDE: $N = 101 \text{ rpm}$
 $r = 4 \text{ cm}$

por lo tanto $V_{\max} = 2.542 \text{ [m/h]}$

$$V_{\max} = 2.542 \text{ [m/h]}(1\text{h}/60\text{seg}) = .42 \text{ [m/s]}$$

Sin embargo, como nuestro robot no trabaja a 24 [V], sino a 14.4[V], ya además de que las ruedas están cubiertas por la oruga la velocidad máxima que alcanzó fue de 30 [cm/s] aproximadamente.

Para poder obtener los datos anteriores, primero se realizó un programa donde los motores trabajan a máxima velocidad, y se mueven por comando sencillos, vía hyperterminal, primero el PIC nos presenta en pantalla que esta lista y el nombre del programa que está corriendo, después el PIC, nos pregunta la dirección a la que queremos que se mueva la base, esto lo hace imprimiendo una 'd', y la respuesta que espera es 'i', para indicar un movimiento a la izquierda, 'd' para un movimiento a la derecha, 'c' para el centro y una 'n' para girar sobre su eje.

En la figura 5.1 se muestra la hyperterminal con estos comandos.

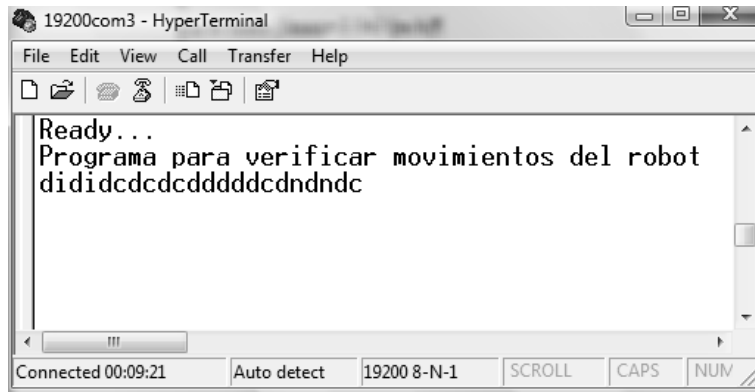


FIGURA 5.1: HYPERTERMINAL MOVIMIENTO DEL ROBOT

5.2 PRUEBA DE RECONOCIMIENTO DE COLORES

Para esta prueba del reconocimiento de los colores se ocupó un programa donde solo contestaba si veía el color o no lo veía, para esta prueba se calibró, poniéndole los objetos a diferentes distancias y diferentes tonos de luz, con lo que obteníamos un rango más amplio de los valores RGB, estos objetos con los que se calibró fueron cartulinas de color amarillo, rojo, verde y azul.

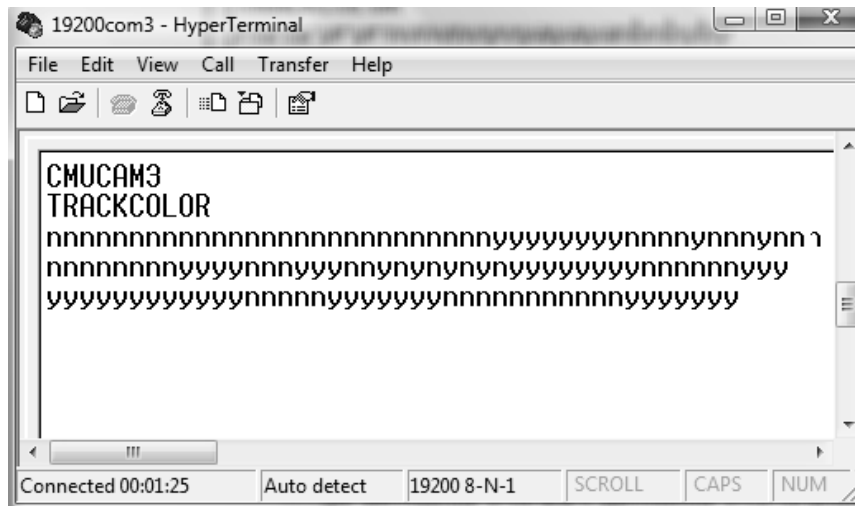


FIGURA 5.2: HYPERTERMINAL RECONOCIMIENTO DE COLORES

A continuación se muestra un pedazo de código que muestra como se trackeo el color rojo

```
printf("\n\rCMUCAM3");
printf("\n\rTRACKCOLOR");
while(true)
{
    simple_track_color(&t_pkt);
    if(t_pkt.centroid_x>10 && t_pkt.centroid_x<=174)printf('y');
    else printf('n' );
    cc3_timer_wait_ms(200);
}
```

5.3 PRUEBA DEL MOVIMIENTO DEL ROBOT Y EL SISTEMA DE VISIÓN

En un principio para asegurarnos de que los sistemas de visión y el sistema embebido de control, trabajaran perfectamente, se probaron por separado, es decir en ambos se simulaba mediante cualquier hyperterminal la respuesta o pregunta que recibía del otro sistema. Creímos eficiente esta prueba porque luego se vuelve complicado saber qué sistema es el que está fallando y en que parte, por ejemplo si llegaba a pasar que el sistema de control no respondiera o no transmitiera podíamos checar tanto el código como los cables del puerto serie que se están utilizando hasta checar el circuito integrado que permite la comunicación serie, es decir ir de lo general a lo particular.

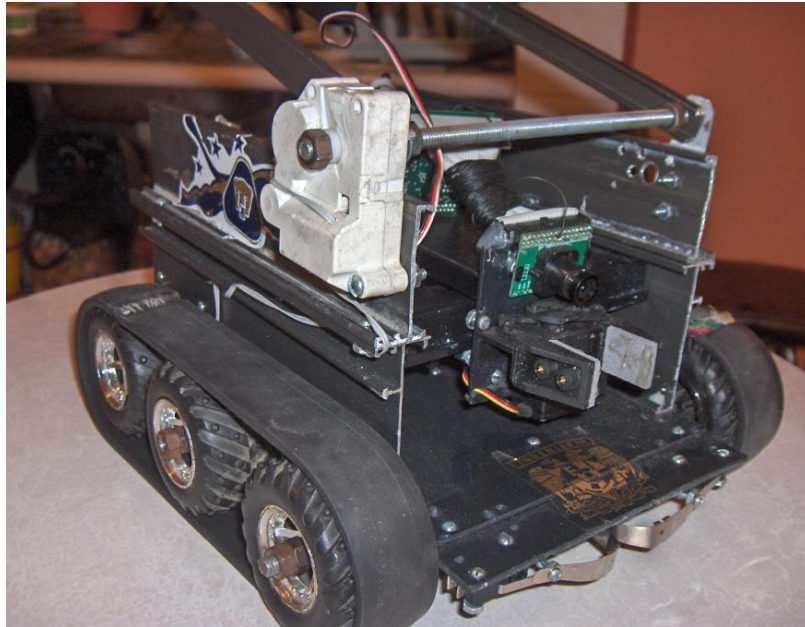


FIGURA 5.3: IMAGEN REAL DEL MOVIMIENTO REAL

5.3.1 PRUEBA DEL MOVIMIENTO DEL ROBOT CON VISIÓN SIMULADA

En esta prueba lo que se hizo fue simular el sistema de visión para poder checar todos los movimientos tanto de la base como del brazo y el gripper, y así también comprobar si el programa que se hizo funciona como debería.

En este caso se le contestó al PIC por medio de la hyperterminal, los caracteres que debía recibir de la cámara.

En la siguiente figura se hizo la rutina de la búsqueda del cubo amarillo, donde el PIC preguntó por dirección, si el cubo era amarillo, si veía cable rojo ó cable verde.

Servomotores	800	3	2400	0
Cmucam	130	1	130	0
Kit_picII	140	1	140	0
Motores	900	2	----	1800
Motor brazo	200	1	-----	200
Total			2670	2000

TABLA 5.1: CONSUMO DE ENERGÍA

Cabe mencionar que en la tabla anterior se muestran los valores máximos de corriente por dispositivo, y máximo total en el robot, sin embargo, no todos estos elementos actúan al mismo tiempo, por lo tanto en las pruebas realizadas, durante la prueba se registro un consumo máximo alrededor de un 1000 [mA], por lo que nuestra fuente de energía nos puede durar alrededor de 4 a 5 horas.

5.4 RESULTADOS

Una vez que se terminaron estas pruebas, se prosiguió a juntar los dos sistemas tanto el de visión como el de movimiento y pudimos comprobar que la búsqueda del cubo amarillo la realizaba sin ningún problema, sin embargo en la búsqueda de los cables nos vimos en la necesidad de reajustar el código, así como la calibración de la cámara, ya que no se previó que el robot hacía sombra en el cubo, sin embargo esta corrección no tuvo mayor problema, además se corrigió la distancia entre el robot y el cubo, ya que se tuvo problemas en la distancia a la que bajaba el brazo y esto ocasionaba que no pudiera girar bien el cubo y lo llegaba a separarla de su base o tirarla.

Por otro lado también se ajusto la velocidad a la que caminaba, porque no podía ir a la misma velocidad con la que busca en la parte de abajo, que al subir o bajar la rampa, o cuando se está alienando.

También nos dimos cuenta que en la calibración de los colores, se le estaba sumando y restando dos unidades porque así se nos indicaba en el manual de la cmucam, sin embargo nos dimos cuenta que si se le aumentaba mas el rango teníamos una mejor resolución y menor número de error porque si se ponía el robot a navegar sobre una superficie blanca con mucha luz alrededor podía ver las paredes como amarillo, incluso había valores como el verde claro que caían en el rango del amarillo y llegaba a confundirse. Es por eso que estos valores para nosotros no fueron los ideales así que decidimos tener una diferencia de 40 unidades a partir de la media obtenida en el trackeo, para las 2 primera componentes y 8 unidades para la tercera, ya que a este rango reconocía mejor el color amarillo de la bomba y así pudimos descartar ruido de otros tonos de amarillo que estuvieran presentes en su rango de visión, también nos dimos cuenta que ocupando un filtro del número 2 se nos metía menos ruido.

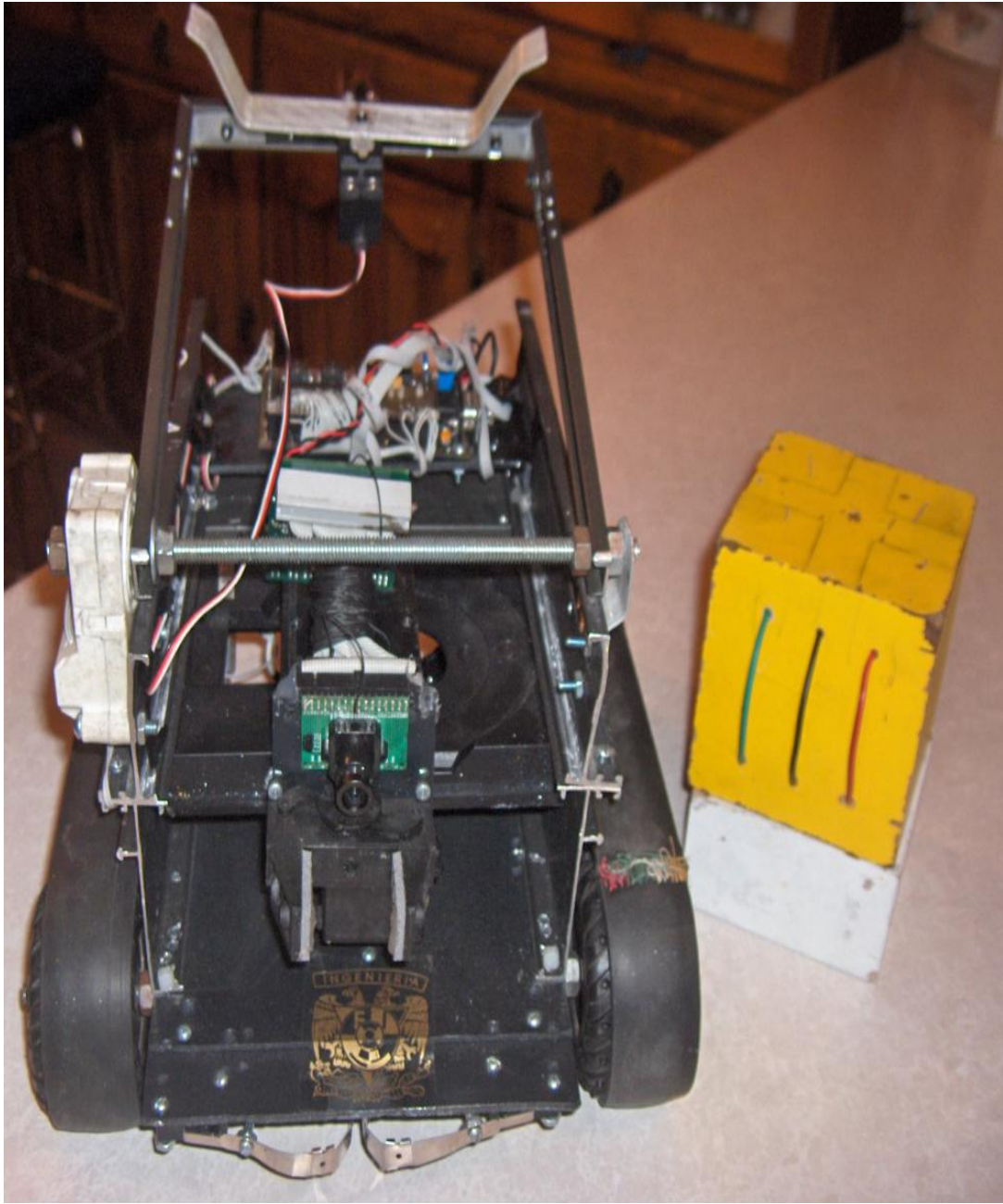


FIGURA 5.6: ROBOT RESULTADOS

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se muestran las conclusiones a la que se llegó y los trabajos que a futuro pueden ayudar a un mejor desempeño del robot.

5.1 CONCLUSIONES

Para la parte mecánica nos pudimos dar cuenta que a la hora de diseñar el robot, se tenía que considerar la limitantes que tenemos por no tener bases sólidas en mecánica, ya que esto nos llevó la mayor parte del tiempo, en el diseño, pese a esto consideramos que se hizo un buen trabajo, ya que podía subir la rampa y sus movimientos eran muy precisos, y afortunadamente se pudieron acoplar tanto la estructura de aluminio, como los motores y los ejes de las llantas.

El robot utiliza ocho pilas de níquel-metal, de 3.6 [V], a 1800 [mA], con las cuales pudimos proveer a nuestro robot de una fuente de energía de 14.4 [V] a 3.6 [A], lo cual pudimos comprobar que fue suficiente en todas las pruebas que se realizaron. Pudimos estar trabajando sin necesidad de recargar las baterías durante una jornada de aproximadamente 8 horas, lo que es de gran ayuda ya que no se pierde tiempo en estar cargando las baterías. Esto nos dio una gran seguridad en las competencias, ya que no teníamos que preocuparnos porque se quedará sin energía durante la prueba.

Los motores que se ocuparon en este trabajo, fueron ideales en el diseño, ya que por el gran torque que tienen pudo moverse el robot, sin ningún problema, ya que por ser un movimiento por orugas se requiere de más potencia, porque mucho de este movimiento se pierde en el arrastre, aun cuando los motores no trabajaron a toda su potencia. Una desventaja que se vio en estos motores es que son muy pesados; sin embargo, por la potencia que tenían no se vio reflejado en el movimiento.

La tarjeta de control KITPIC se diseño principalmente como un sistema de desarrollo y aprendizaje para los microcontroladores PIC de 40 pines, y para su fácil implementación en diferentes proyectos de electrónica, entre ellos la robótica como son las plataformas móviles o un robot completo como es en este caso, en este proyecto se cumplieron los objetivos para los que fue diseñada la tarjeta, ya que se montó también en proyectos que se trabajaron en paralelo con este.

La tarjeta de potencia o puente H, que se diseño a base del integrado L298, cumplió con todas nuestras expectativas, ya que por sus características de diseño fue muy fácil su implementación, y su comunicación con el KITPIC, además de que nunca se tuvo la necesidad de sustituirla.

En el diseño del brazo, y el gripper creemos que fue suficiente, con esa estructura porque no se necesitaba un diseño más sofisticado como una mano robótica, gracias a que solo bastaba girar el

cubo, lo único que se requería calibrar a qué distancia bajaba y que la pinza quedara exactamente en medio del cubo y en el caso de los cables solo bastaba con tirar de ellos, con poca fuerza.

Para el sistema de visión la cámara CMUCAM3, fue una herramienta muy poderosa y muy fácil de implementar ya que tiene un gran número de librerías que nos facilitaron el trabajo, y con solo modificar algunas partes del código, pudimos implementarlo de manera efectiva para la detección de colores primarios en el espacio RGB, sin embargo se le invierte mucho tiempo, al estar calibrando cada color, y si cambian las condiciones de luz se tiene que volver a calibrar, es por eso que antes de cada competencia alrededor de 15 minutos antes se tenía que hacer esta calibración, aún así fue muy eficiente frente a los sensores reflectivos en los que se había pensado por primera vez ya que estos son excelentes cuando se ocupan a nivel de piso como en el caso de los seguidores de línea, pero al estar expuestos a una gran cantidad de luz no trabajan como se desea y se les mete mucho ruido.

Teniendo en cuenta que este proyecto de tesis se diseñó para un concurso, y que es una simulación de un hecho real, se trató de manejar el concepto de autonomía y el que el robot pudiera navegar en diferentes terrenos sin mayor problema obteniendo el primer lugar en el torneo Mexicano de Robótica celebrado en el Palacio de Minería en el 2008 y el cuarto lugar en el torneo Latinoamericano de robótica celebrado en Brasil del mismo año.

5.2 TRABAJOS FUTUROS

En este trabajo de tesis todos los componentes que se ocuparon fueron suficientes para el buen funcionamiento del mismo, sin embargo, y como en todo trabajo de robótica, siempre hay la posibilidad de mejorarlo, para esto se puede sugerir, una serie de ideas como las siguientes:

Todos los movimientos del robot se realizan a base de retardos de tiempo, lo que hace muy inexacto el robot cuando la carga de las pilas es demasiado baja, lo que nos lleva a sugerir el uso de encoders, en el cual los movimientos pueden ser controlados por vueltas del motor y no por tiempo, se sugiere para mayor exactitud un dispositivo acoplado directamente al eje del motor que nos pueda determinar cuánto ha girado.

Otra de las mejoras que se puede hacer es el uso de un DSPIC, ya que estos por su estructura de hardware nos pueden ayudar a desarrollar algoritmos más complejos y más eficientes, ya que podríamos conectar y controlar más motores de corriente directa, ya que el PIC que se utiliza en ésta tarjeta solo puede manejar dos.

Para el caso de la visión, como se mencionó, la CMUCAM es muy eficiente, ya que en este caso sólo se requirió de encontrar colores primarios, pero para tener una mejor precisión se puede ocupar el reconocimiento de formas por medio de otra cámara y el uso de otra computadora para procesar toda esta información.

APÉNDICE A

A. CÓDIGO DEL MOVIMIENTO DEL ROBOT

```
/* **** */
/*                                     Tesis                                     */
/* **** */
/* **** */
/* // Definiciones PIC */
/* **** */
#include <18f452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=1000000)
#use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7)
/* **** */
/* Area de include */
/* **** */
#include <stdlib.h>
#include <stdio.h>
#include <input.c>
#include <LCD416.C>
#include <chacharas_brazo.c>
/* **** */
//RESERVA DE MEMORIA PARA TINYBOOTLOADER
#define LOADER_SIZE 0xFF //tinybld size + a bit more (200 bytes is enough)
#org getenv("PROGRAM_MEMORY")-LOADER_SIZE , getenv("PROGRAM_MEMORY")-1 void
boot_loader(void) {}
/* **** */
/* Zona de define */
/* **** */
#define derecha 1
#define izquierda 0
#define amarillo 1
#define azul 0
#byte portb=0xf81
/* **** */
/* Inicializa interrupciones y temporizadores*/
/* **** */
void inicializa_interrupciones() {

    port_b_pullups(true); //habilita resistencias de pullups
    setup_adc(ADC_OFF); //puerto analogico off
}
/* **** */
/* Inicializa puertos*/
/* **** */
void inicializa_puertos() {
    //set_tris_c(0x80);
    set_tris_b(0x0f);
}
/* **** */
/* Programa principal */
/* **** */
void main() {
    char comand;
    int push;
    int i;
```

```

int posicion;
int cont=0;
int1 side;
int1 bomba;
lcd_init();
init_robot();
inicializa_interrupciones();
inicializa_puertos();
centro();
center_posicion_servo();

while(true) {
push=~portb;
push=(push&0x0f);
lcd_putc("\fready");
    for(i=0;i<3;i++){
        lcd_gotoxy(i+6,1);
        lcd_putc(".");
        delay_ms(250);
    }
    if(push)break;
}
delay_ms(1000);
lcd_gotoxy(1,1);
lcd_putc("\fbusca bomba");

    if(push==0x04)
    {
        bomba=amarillo;
        lcd_gotoxy(5,2);
        lcd_putc("amarilla");
        delay_ms(1000);
    }
    else if(push==0x01)
    {
        side=izquierda;
        bomba=azul;
        lcd_gotoxy(7,2);
        lcd_putc("azul");
        delay_ms(1000);
    }
    else if(push==0x02)
    {
        side=derecha;
        bomba=azul;
        lcd_gotoxy(7,2);
        lcd_putc("azul");
        delay_ms(1000);
    }

    while(true){
        if(!bomba)
        {
            mv_robot(1000,1000);
            delay_ms(1100);
            mv_robot(0,0);
            delay_ms(2000);
            if(side)
            {

```

```

        mv_robot(-1000,1000);
        delay_ms(900);
        mv_robot(0,0);
        delay_ms(10);
    }
    else
    {
        mv_robot(1000,-1000);
        delay_ms(900);
        mv_robot(0,0);
        delay_ms(10);
    }
    mv_robot(-1000,-1000);
    delay_ms(2900);
    mv_robot(0,0);
    delay_ms(10);
}

while(true){
    output_toggle(pin_c3);
    push=~portb;
    push=(push&0x0f);
    switch(push)    {

        case 0:    printf("D");
                  delay_ms(10);
                  comand=getc();
                  delay_ms(10);
                  switch(comand)  {

                      case 'C': mv_robot(350,350);
                                break;

                      case 'I': mv_robot(300,400);
                                break;

                      case 'D': mv_robot(400,300);
                                break;

                      case 'n': mv_robot(300,300);
                                delay_ms(100);
                                mv_robot(0,0);
                                break;

                  }

                  break;

        case 1:    mv_robot(0,350);
                  break;

        case 2:    mv_robot(350,0);
                  break;

        case 3:    mv_robot(0,0);
                  delay_ms(10);
                  mv_robot(-300,-300);
                  delay_ms(500);
                  mv_robot(0,0);
    }
}
////////////////////////////////////

```

```

while(true)
{
output_high(pin_c3);
lcd_putc("\f");
lcd_putc("Bomba azul");
delay_ms(2000);
lcd_putc("\f");
lcd_gotoxy(2,1);
lcd_putc("Busca la cara ");
lcd_gotoxy(2,2);
lcd_putc("del cable rojo");
printf("R"); //pregunta cara de la bomba azul con
              cable rojo
comand=getc();

              if(comand=='y') break;
              else
              {
                  if(side)
                  {
                      lcd_putc("\f");
                      lcd_putc(" Activa brazo");
                      ejecuta_brazo(side);
                      posicion=1;
                  }
                  else
                  {
                      lcd_putc("\f");
                      lcd_putc(" Activa brazo");
                      ejecuta_brazo(side);
                      posicion=0;
                  }
              }
}
quita_cables('R');
//////////////////////////////////////////////////////////////////
adelante();
center_posicion_servo();
center_pos_izq();
//////////////////////////////////////////////////////////////////
atras();
while(true)
{
lcd_putc("\f");
lcd_gotoxy(2,1);
lcd_putc("Busca la cara ");
lcd_gotoxy(2,2);
lcd_putc("del cable verde");
printf("V");
comand=getc();
              if(comand=='y') break;
              else
              {
                  lcd_putc("\f");
                  lcd_putc(" Activa brazo");
                  ejecuta_brazo(side);
              }
}
//////////////////////////////////////////////////////////////////
quita_cables('V');

```

```

        bomba=amarillo;
        break; //case 3(swiches)
    } //switch frontales
        if(bomba)break;
    } //fin while

mv_robot(-500,-500);
delay_ms(2000);
mv_robot(0,0);
delay_ms(10);
    if(side) {
        mv_robot(-500,500);
        delay_ms(700);
        mv_robot(0,0);
        delay_ms(10);
    }
    else
        {
        mv_robot(500,-500);
        delay_ms(700);
        mv_robot(0,0);
        delay_ms(10);
        }
    lcd_putc("\f");
    lcd_gotoxy(4,1);
    lcd_putc("Bomba azul ");
    lcd_gotoxy(3,2);
    lcd_putc("Desactivada");
    delay_ms(3000);

} //fin if bomba azul desactivada
else //indica si ya desactivo la boma azul siga con amarilla
    {
    lcd_putc("\f");
    lcd_gotoxy(2,1);
    lcd_putc("busca bomba ");
    lcd_gotoxy(2,2);
    lcd_putc("amarilla");
    mv_robot(500,500);
    delay_ms(1000);
    mv_robot(0,0);

    while(true) //checa estado de los swiches para empezar a
        buscar la bomba amarilla
        {
        output_toggle(pin_c3);
        push=~portb;
        push=(push&0x0f);
        switch(push){ //principal bomba amarilla

            case 0: printf("d");
                    comand=getc();
                    delay_ms(10);
                    switch(comand){

                        case 'c': mv_robot(325,325);
                                    break;

                        case 'i': mv_robot(200,400);

```

```

        break;

    case 'd': mv_robot(400,200);
             break;

    case 'n': mv_robot(600,600);
             delay_ms(500);
             mv_robot(0,0);
             for(i=0;i<25;i++)
             {
                 mv_robot(-800,800);
                 delay_ms(150);
                 mv_robot(0,0);
                 delay_ms(80);
                 printf("d");
                 comand=getc();
                 if(comand!='n')break;
             }
             break;
        }

    break; //case cero

case 1: mv_robot(0,600);
       break;

case 2: mv_robot(600,0);
       break;

case 3: mv_robot(0,0);
       delay_ms(10);
       mv_robot(-300,-300);
       delay_ms(500);
       mv_robot(0,0);////////////////////////////////////
       centro();
       while(true) //aqui choca con algo y checamos si es
                  //la bomba, porque solo se sabe q se
                  //activaron los 2 switches
       {
           printf("a");
           comand=getc();
           if(comand=='y')
           { //checa si no choca
             //con la pared
             lcd_putc("\f");
             lcd_putc(" Bomba amarilla");
             cont++;
             if(cont==2)
             {
                 cont=0;
                 //////////////////////////////////////
                 while(true)
                 {
                     output_high(pin_c3);
                     lcd_putc("\f");
                     lcd_putc("busca cables");
                     printf("c");

```



```

comand=getc();
if(comand=='y')break;

else {
    lcd_putc("\f");
    lcd_putc(" Activa brazo");
    ejecuta_brazo(side);
}

}

////////////////////////////////////
quita_cables('r');
quita_cables('v');

lcd_putc("\f");
lcd_gotoxy(2,1);
lcd_putc("Bomba amarilla ");
lcd_gotoxy(3,2);
lcd_putc("Desactivada");
delay_ms(3000);
while(true){
    lcd_putc("\f");
    lcd_gotoxy(7,1);
    lcd_putc("Bombas ");
    lcd_gotoxy(3,2);
    lcd_putc("Desactivadas");
    output_toggle(pin_c3);
    delay_ms(1000);
}

else
{
    mv_robot(-400,-400);
    delay_ms(800);
    mv_robot(0,0);
    mv_robot(200,200);
    break;
} //fin de if que verifica si es la bomba amarilla
////////////////////////////////////

else
{
    mv_robot(-400,-400);
    delay_ms(2000);
    mv_robot(0,0);
    delay_ms(400);
    mv_robot(-400,400);
    delay_ms(1000);
    mv_robot(400,400);
    delay_ms(2000);
    mv_robot(0,0);
    break;
}

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

```

```
        } //cierre de while de case 3
    } // cierre de switch
} //cierre de while donde va a empezar a checar estado de los sw

} //cierre else bomba amarilla
}
```

APÉNDICE B

B. LIBRERÍA CREADA PARA EL ROBOT

```
//////////////////////////////////////
//Libreria para el uso del robot desactivador de bombas //
//////////////////////////////////////
#define abre 1
#define cierra 0
#define left 1
#define right 0

int cont;

void init_robot() { //Inicializa robot
    // Se inicializa el temporizador 2 para generar el PWM
    setup_timer_2(T2_DIV_BY_4, 255, 1);
    // Se inicializan los pines para generar los pwms
    setup_ccp1(CCP_PWM);
    setup_ccp2(CCP_PWM);
    set_pwm1_duty(0);
    set_pwm2_duty(0);
}

void mv_robot(signed long wi,signed long wd){
//lanta derecha
    if(wd==0) //Si es cero, frena
    {
        OUTPUT_HIGH(pin_E0);
        OUTPUT_HIGH(pin_A5);
        wd=wd;
    }
    if(wd>0) //valor positivo avanza
    {
        wd=wd;
        OUTPUT_LOW(pin_A5);
        OUTPUT_HIGH(pin_E0);
    }
    else //valor negativo retrocede
    {
        wd=-wd;
        OUTPUT_HIGH(pin_A5);
        OUTPUT_LOW(pin_E0);
    }

//lanta izquierda
    if(wi==0) //Si es cero, frena
    {
        OUTPUT_HIGH(pin_E2);
        OUTPUT_HIGH(pin_E1);
        wi=wi;
    }
    if(wi>0) //valor positivo avanza
    {
        wi=wi;
        OUTPUT_LOW(pin_E1);
        OUTPUT_HIGH(pin_E2);
    }
}
```

```

else //valor negativo retrocede
{
wi=-wi;
OUTPUT_HIGH(pin_E1);
OUTPUT_LOW(pin_E2);
}

set_pwm1_duty(wi); //Coloca el valor de PWM izquierdo
set_pwm2_duty(wd); //Coloca el valor de PWM derecho
}

void mv_brazo(signed long wd){

if(wd==0)
{
OUTPUT_HIGH(pin_A0);
OUTPUT_HIGH(pin_A1);
wd=wd;
}

if(wd>0) //valor positivo levanta brazo
{
wd=wd;
OUTPUT_LOW(pin_A1);
OUTPUT_HIGH(pin_A0);
}

else //valor negativo baja brazo
{
wd=-wd;
OUTPUT_HIGH(pin_A1);
OUTPUT_LOW(pin_A0);
}

set_pwm1_duty(wd);
set_pwm2_duty(0);
}

void atras() {
mv_robot(-300,-300);
delay_ms(300);
mv_robot(0,0);
delay_ms(100);
}

void adelante() {
mv_robot(300,300);
delay_ms(300);
mv_robot(0,0);
delay_ms(100);
}

void centro() { //posiciona el gripper al centro del robot
int m;
for(m=0;m<50;m++) {
output_high(PIN_b6);
delay_us(1500);
output_low(PIN_b6);
}
}

```

```

        delay_us(18500);
    }
}

void gira(side) { //Gira la bomba dependiendo el valor de side
    int m;
    for(m=0;m<40;m++) {
        if(!side)
        {
            output_high(PIN_b6);
            delay_us(500);
            output_low(PIN_b6);
            delay_us(19500);
        }
        else
        {
            output_high(PIN_b6);
            delay_us(2300);
            output_low(PIN_b6);
            delay_us(17700);
        }
    }
}

void levanta_brazo() {
    mv_brazo(-1000);
    delay_ms(2000);
    mv_brazo(-200);
    delay_ms(300);
    mv_brazo(1);
    delay_ms(700);
}

void baja_brazo() {
    mv_brazo(1000);
    delay_ms(2000);
    mv_brazo(200);
    delay_ms(200);
    mv_brazo(1);
}

void ejecuta_brazo(side){
    levanta_brazo();
    mv_brazo(0,1000);
    delay_ms(50);
    mv_brazo(0,0);
    delay_ms(100);
    gira(side);
    baja_brazo();
}

void center_posicion_servo(){
    long i;
    while(input(pin_b2))
    {
        output_high(PIN_b5);
        delay_us(900);
        output_low(PIN_b5);
    }
}

```

```

        delay_us(19100);
    }
        for(i=0;i<13;i++)
            {
                output_high(PIN_b5);
                delay_us(2100);
                output_low(PIN_b5);
                delay_us(17900);
            }
    }

void center_pos_izq() {
    long i;
    for(i=0;i<18;i++)
        {
            output_high(PIN_b5);
            delay_us(2100);
            output_low(PIN_b5);
            delay_us(17900);
        }
    }

void PWM_GENERATION_SERVO1(int side){

    if(side)
        {
            output_high(PIN_b5);
            delay_us(1900);
            output_low(PIN_b5);
            delay_us(18100);
        }
    else
        {
            output_high(PIN_b5);
            delay_us(1150);
            output_low(PIN_b5);
            delay_us(1850);
        }
    }

void PWM_GENERATION_SERVO2(int side){ //Funcion abre cierra pinza

    if(side)//abre
        {
            output_high(PIN_b4);
            delay_us(1000);
            output_low(PIN_b4);
            delay_us(19000);
        }
    else //cierra
        {
            output_high(PIN_b4);
            delay_us(2100);
            output_low(PIN_b4);
        }
    }

```

```

        delay_us(17900);
    }
}

void ejecuta_griper(){
    mv_robot(400,400);
    delay_ms(150);
    mv_robot(0,0);
    for(cont=0;cont<25;cont++) {
        PWM_GENERATION_SERVO2(cierra);
    }

    mv_robot(-350,-350);
    delay_ms(150);
    mv_robot(0,0);
    for(cont=0;cont<25;cont++) {
        PWM_GENERATION_SERVO2(abre);
    }
}

void quita_cables(char comando) {
    char comand;
    center_posicion_servo();
    center_pos_izq();
    while(true)
    {
        lcd_putc("\f");
        lcd_putc(" busca cable");
        if(comando=='r'||comando=='R')
        {
            lcd_gotoxy(7,2);
            lcd_putc("rojo");
        }
        else
        {
            lcd_gotoxy(7,2);
            lcd_putc("verde");
        }
        printf("%c",comando); //cable rojo
        comand=getc();
        if(comand=='y')
        {
            lcd_putc("\f");
            lcd_putc(" ejecuta griper");
            delay_ms(1000);
            ejecuta_griper();
            break;
        }
        else
        {
            PWM_GENERATION_SERVO1(right);
            if(!pin_b2)
            {
                center_posicion_servo();
                center_pos_izq();
            }
        }
    }
}

```

```
        }  
while(true)  
{  
    printf("%c",comando);  
    comand=getc();  
    if(comand=='y')ejecuta_griper();  
    else    break;  
}  
}
```


APÉNDICE C

C. LIBRERÍA CREADA PARA EL ROBOT

```
//nombre del programa: bomba azul y amarilla
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
#include <ctype.h>
#include <cc3.h>
#include <cc3_ilp.h>
#include <cc3_color_track.h>

void simple_track_color(cc3_track_pkt_t* t_pkt,cc3_track_pkt_t* t_pkt2,cc3_track_pkt_t* t_pkt3,
                        cc3_track_pkt_t* t_pkt4,cc3_track_pkt_t* t_pkt5,cc3_track_pkt_t* t_pkt6);

int main(void) {

    char c;
    uint32_t start_time;
    cc3_filesystem_init ();
    cc3_track_pkt_t t_pkt;
    cc3_track_pkt_t t_pkt2;
    cc3_track_pkt_t t_pkt3;
    cc3_track_pkt_t t_pkt4;
    cc3_track_pkt_t t_pkt5;
    cc3_track_pkt_t t_pkt6;

    cc3_uart_init (0,CC3_UART_RATE_19200,CC3_UART_MODE_8N1,
                  CC3_UART_BINMODE_TEXT);

    cc3_camera_init ();

    cc3_camera_set_resolution(CC3_CAMERA_RESOLUTION_LOW);
    cc3_camera_set_auto_white_balance (true);
    cc3_camera_set_auto_exposure (true);

    cc3_led_set_state (0, false);
    cc3_led_set_state (1, false);
    cc3_led_set_state (2, false);

    cc3_timer_wait_ms (1000);
    cc3_led_set_state (0, true);

    cc3_led_set_state (0, true);
    cc3_led_set_state (0, false);
    cc3_led_set_state (0, true);
    cc3_led_set_state (0, false);

    start_time = cc3_timer_get_current_ms ();
    while (!cc3_button_get_state ());
    cc3_led_set_state (1, true);
    /*// sample showing how to use timer
    printf ("It took you %dms to press the button\n",
           cc3_timer_get_current_ms () - start_time);*/
```

```

// init pixbuf with width and height
cc3_pixbuf_load();

// Load in your tracking parameters

//Valores que se obtuvieron del trackeo del color amarillo
t_pkt.lower_bound.channel[CC3_CHANNEL_RED] = 80;
t_pkt.upper_bound.channel[CC3_CHANNEL_RED] = 167;
t_pkt.lower_bound.channel[CC3_CHANNEL_GREEN] =85;
t_pkt.upper_bound.channel[CC3_CHANNEL_GREEN] =165;
t_pkt.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
t_pkt.upper_bound.channel[CC3_CHANNEL_BLUE] = 16;
t_pkt.noise_filter = 2;

//Valores que se obtuvieron del trackeo del color rojo sobre el amarillo
t_pkt2.lower_bound.channel[CC3_CHANNEL_RED] =150;
t_pkt2.upper_bound.channel[CC3_CHANNEL_RED] = 230;
t_pkt2.lower_bound.channel[CC3_CHANNEL_GREEN] =0;
t_pkt2.upper_bound.channel[CC3_CHANNEL_GREEN] =70;
t_pkt2.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
t_pkt2.upper_bound.channel[CC3_CHANNEL_BLUE] = 50;
t_pkt2.noise_filter = 1; //luz

//Valores que se obtuvieron del trackeo del color verde sobre el amarillo
t_pkt3.lower_bound.channel[CC3_CHANNEL_RED] =18;
t_pkt3.upper_bound.channel[CC3_CHANNEL_RED] =98;
t_pkt3.lower_bound.channel[CC3_CHANNEL_GREEN] =42;
t_pkt3.upper_bound.channel[CC3_CHANNEL_GREEN] =122;
t_pkt3.lower_bound.channel[CC3_CHANNEL_BLUE] = 41;
t_pkt3.upper_bound.channel[CC3_CHANNEL_BLUE] = 121;
t_pkt3.noise_filter = 0; //luz

//Valores que se obtuvieron del trackeo del color rojo sobre el azul
t_pkt4.lower_bound.channel[CC3_CHANNEL_RED] =155;
t_pkt4.upper_bound.channel[CC3_CHANNEL_RED] = 225;
t_pkt4.lower_bound.channel[CC3_CHANNEL_GREEN] =0;
t_pkt4.upper_bound.channel[CC3_CHANNEL_GREEN] =75;
t_pkt4.lower_bound.channel[CC3_CHANNEL_BLUE] = 0;
t_pkt4.upper_bound.channel[CC3_CHANNEL_BLUE] = 60;
t_pkt4.noise_filter = 1; //luz

//Valores que se obtuvieron del trackeo del color verde sobre el azul
t_pkt5.lower_bound.channel[CC3_CHANNEL_RED] =22;
t_pkt5.upper_bound.channel[CC3_CHANNEL_RED] =100;
t_pkt5.lower_bound.channel[CC3_CHANNEL_GREEN] =47;
t_pkt5.upper_bound.channel[CC3_CHANNEL_GREEN] =125;
t_pkt5.lower_bound.channel[CC3_CHANNEL_BLUE] = 47;
t_pkt5.upper_bound.channel[CC3_CHANNEL_BLUE] = 125;
t_pkt5.noise_filter = 0; //luz

//Valores que se obtuvieron del trackeo del color azul
t_pkt6.lower_bound.channel[CC3_CHANNEL_RED] =0;
t_pkt6.upper_bound.channel[CC3_CHANNEL_RED] =85;
t_pkt6.lower_bound.channel[CC3_CHANNEL_GREEN] =0;
t_pkt6.upper_bound.channel[CC3_CHANNEL_GREEN] =88;
t_pkt6.lower_bound.channel[CC3_CHANNEL_BLUE] = 42;
t_pkt6.upper_bound.channel[CC3_CHANNEL_BLUE] = 250;
t_pkt6.noise_filter = 0; //luz

```

```

while(true)
{
    simple_track_color(&t_pkt,&t_pkt2,&t_pkt3,&t_pkt4,&t_pkt5,&t_pkt6);
    c=getchar();

    switch(c)
    {
    case 'd':
        if(t_pkt.centroid_x>2 && t_pkt.centroid_x<=78) printf("i");
        else if(t_pkt.centroid_x>78 && t_pkt.centroid_x<=96)printf("c");
        else if(t_pkt.centroid_x>96 && t_pkt.centroid_x<=174) printf("d");
        else printf("n");
        break;

    case 'a':
        if(t_pkt.centroid_x>0 && t_pkt.centroid_x<=116) printf("y");
        else printf("n");
        break;

    case 'c':
        if((t_pkt2.centroid_x>29 && t_pkt2.centroid_x<=145)||(t_pkt3.centroid_x>29 &&
t_pkt3.centroid_x<=145)) printf("y");
        else printf("n");
        break;

    case 'r':
        if(t_pkt2.centroid_x>58 && t_pkt2.centroid_x<=116) printf("y");
        else printf("n");
        break;

    case 'v':
        if(t_pkt3.centroid_x>58 && t_pkt3.centroid_x<=116) printf("y");
        else printf("n");
        break;

    case 'D':
        if(t_pkt6.centroid_x>2 && t_pkt6.centroid_x<=78) printf("I");
        else if(t_pkt6.centroid_x>78 && t_pkt6.centroid_x<=96)printf("C");
        else if(t_pkt6.centroid_x>96 && t_pkt6.centroid_x<=174) printf("D");
        else printf("n");
        break;

    case 'R':
        if(t_pkt4.centroid_x>58 && t_pkt4.centroid_x<=116) printf("y");
        else printf("n");
        break;

    case 'V':
        if(t_pkt5.centroid_x>58 && t_pkt5.centroid_x<=116) printf("y");
        else printf("n");
        break;

    }//fin de switch
}fin while true

//fin de main

```

```

void simple_track_color(cc3_track_pkt_t * t_pkt,cc3_track_pkt_t * t_pkt2,cc3_track_pkt_t *
t_pkt3,cc3_track_pkt_t * t_pkt4,cc3_track_pkt_t * t_pkt5,cc3_track_pkt_t * t_pkt6)
{
    cc3_image_t img;
    img.channels = 3;
    img.width = cc3_g_pixbuf_frame.width;
    img.height = 1; // image will hold just 1 row for scanline processing
    img.pix = cc3_malloc_rows (1);
    if (img.pix == NULL) {
        return;
    }

    cc3_pixbuf_load ();
    if (cc3_track_color_scanline_start (t_pkt) != 0&& cc3_track_color_scanline_start (t_pkt2) != 0&&
    cc3_track_color_scanline_start (t_pkt3) != 0&&cc3_track_color_scanline_start (t_pkt4) != 0&&
    cc3_track_color_scanline_start (t_pkt5) != 0&& cc3_track_color_scanline_start (t_pkt6) != 0) {

        while (cc3_pixbuf_read_rows (img.pix, 1)) {
            // This does the HSV conversion
            // cc3_rgb2hsv_row(img.pix,img.width);
            cc3_track_color_scanline (&img, t_pkt);
            cc3_track_color_scanline (&img, t_pkt2);
            cc3_track_color_scanline (&img, t_pkt3);
            cc3_track_color_scanline (&img, t_pkt4);
            cc3_track_color_scanline (&img, t_pkt5);
            cc3_track_color_scanline (&img, t_pkt6);
        }

        cc3_track_color_scanline_finish (t_pkt);
        cc3_track_color_scanline_finish (t_pkt2);
        cc3_track_color_scanline_finish (t_pkt3);
        cc3_track_color_scanline_finish (t_pkt4);
        cc3_track_color_scanline_finish (t_pkt5);
        cc3_track_color_scanline_finish (t_pkt6);
        free (img.pix);
        return;
    }
}

```

APÉNDICE D

CÓDIGO DEL MOVIMIENTO DE LA BASE

```

/*****
/* Prueba de movimiento
/*****
/* // Definiciones PIC */
/*****
#include <18f452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=1000000)
#use rs232(baud=19200, xmit=PIN_C6, rcv=PIN_C7)
/*****
//RESERVA DE MEMORIA PARA TINYBOOTLOADER
#define LOADER_SIZE 0xFF //tinybld size + a bit more (200 bytes is enough)
#org getenv("PROGRAM_MEMORY")-LOADER_SIZE , getenv("PROGRAM_MEMORY")-1 void
boot_loader(void) {}
/*****
/* Area de include */
/*****
#include <stdlib.h>
#include <stdio.h>
#include <input.c>
/*****
/* Zona de define */
/*****
#byte portb=0xf81
void init_robot();
void inicializa_interrupciones();
void inicializa_puertos();
void mv_robot(signed long wi,signed long wd);
/*****
/* Programa principal */
/*****
void main(){

    char comand;
    int push;
    init_robot();
    inicializa_interrupciones();
    inicializa_puertos();

        while(true)
        {
            output_toggle(pin_c3);
            push=~portb;
            push=(push&0x0f);
            switch(push){

                case 0: printf("d");
                    delay_ms(10);
                    comand=getc();
                    delay_ms(10);
                    switch(comand) {

                        case 'c': mv_robot(350,350);

```

```

        break;

        case 'i': mv_robot(300,400);
        break;

        case 'd': mv_robot(400,300);
        break;

        case 'n': mv_robot(600,600);
        delay_ms(500);
        mv_robot(0,0);
                for(i=0;i<25;i++)
                {
                        mv_robot(-800,800);
                        delay_ms(150);
                        mv_robot(0,0);
                        delay_ms(80);
                        printf("d");
                        comand=getc();
                        if(comand!='n') break;
                }
        break;
    }
    break;

    case 1: mv_robot(0,350);
    break;

    case 2: mv_robot(350,0);
    break;

    case 3: mv_robot(0,0);
    delay_ms(10);
    mv_robot(-300,-300);
    delay_ms(500);
    mv_robot(0,0);
    }
}

// Fin programa principal //
/*****
/* Inicializa interrupciones y temporizadores*/
*****/
void inicializa_interrupciones(){

    port_b_pullups(true); // habilita resistencias de pullups
    setup_adc(ADC_OFF); // puerto analogico off
    }
/*****
/* Inicializa puertos*/
*****/
void inicializa_puertos() {
    //set_tris_c(0x80);
    set_tris_b(0x0f);
    }
/*****
/* Inicializa robot*/
*****/

```

```

void init_robot() {
    // Se inicializa el temporizador 2 para generar el PWM
    setup_timer_2(T2_DIV_BY_4, 255, 1);

    // Se inicializan los pines para generar los pwms
    setup_ccp1(CCP_PWM);
    setup_ccp2(CCP_PWM);
    set_pwm1_duty(0);
    set_pwm2_duty(0);
}
/*****/
/* Esta funcion coloca los valores de PWM y direccion de las llantas*/
/*****/
void mv_robot(signed long wi,signed long wd){
    //llanta derecha
    if(wd==0)          // cero detiene los motores
    {
        OUTPUT_HIGH(pin_E0);
        OUTPUT_HIGH(pin_A5);
        wd=wd;
    }
    else
    {
        if(wd>0)      //valor positivo avanza
        {
            wd=wd;
            OUTPUT_LOW(pin_A5);
            OUTPUT_HIGH(pin_E0);
        }

        else          //valor negetivo retrocede
        {
            wd=-wd;
            OUTPUT_HIGH(pin_A5);
            OUTPUT_LOW(pin_E0);
        }
    }

    //          llanta izquierda
    if(wi==0)        // cero detiene los motores
    {
        OUTPUT_HIGH(pin_E2);
        OUTPUT_HIGH(pin_E1);
        wi=wi;
    }
    else
    {
        if(wi>0)      //valor positivo avanza
        {
            wi=wi;
            OUTPUT_LOW(pin_E1);
            OUTPUT_HIGH(pin_E2);
        }
    }
}

```

```
else          //valor negetivo retrocede
{
wi=-wi;
OUTPUT_HIGH(pin_E1);
OUTPUT_LOW(pin_E2);
}
}
set_pwm1_duty(wi); //coloca los valores de PWM
set_pwm2_duty(wd);
}
// Fin de programa//
```


APÉNDICE E

El software de Eagle se utilizó para diseñar las tarjetas electrónicas como son el sistema embebido y los drivers para los motores.

Para posteriormente imprimir el diseño y plasmarlo en las tarjetas de cobre y poder soldar sus componentes.

A. Eagle

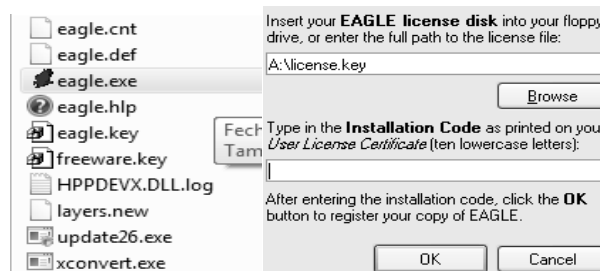
Este programa se puede descargar de la siguiente pagina www.cadsoftusa.com existe una versión libre pero con sus limitanes en la herramientas de diseño que generalmente es una versión para estudiantes.

i Instalación del software para Windows

Este software no necesita instalación, solo hay que ejecutarlo bajo el ambiente Windows. Se copia la carpeta de Eagle 4.11 en la raíz del sistema, de preferencia generalmente C o en la partición en la que se instalan los programas.

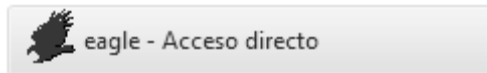


Posteriormente se abre esta carpeta y dentro de ella esta una carpeta llamada bin del cual ejecutamos el programa Eagle.exe.

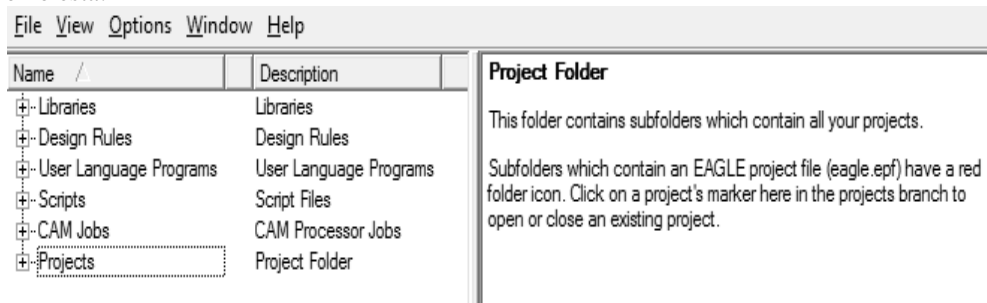


En la pantalla que aparece después de ejecutarlo, le damos click en el botón Browse y escoges la opción de freeware.key, en caso de que no aparezca esta ventana, buscas la carpeta bin, y dentro de esa carpeta se localiza el archivo freeware.key

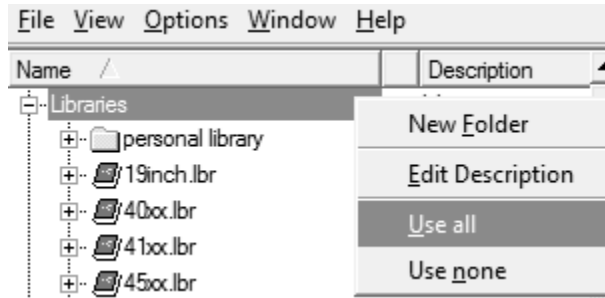
ii CREACIÓN DE UN PROYECTO



El programa lo podemos abrir desde el acceso directo el cual nos mostrará una pantalla llamada panel de control como esta:

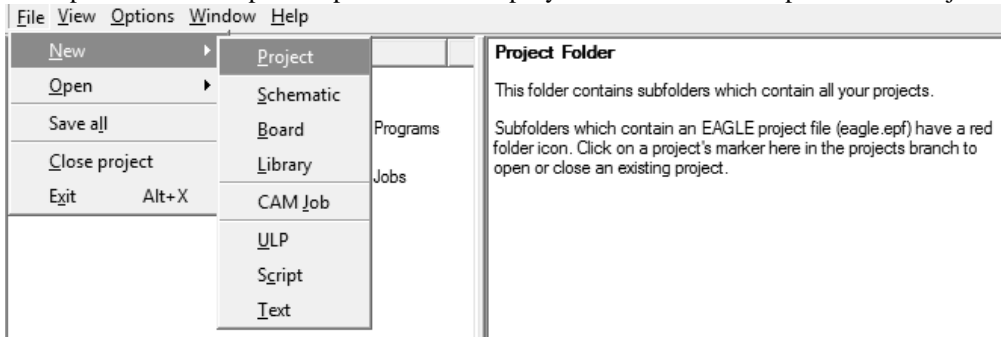


Dentro del panel de control vamos a dar de alta las librerías que necesitamos, en este caso vamos a dar de alta todas, en la parte donde dice Libraries le damos click derecho y escogemos Use all.

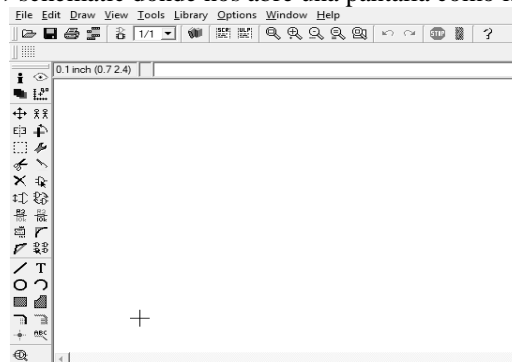


iii DESARROLLO DEL DIAGRAMA ESQUEMÁTICO

Una vez en el panel de control para empezar a crear un proyecto abrimos File después new->Project.



El cual creara una carpeta y nos pedirá ponerle un nombre, dicha carpeta está localizada dentro de la carpeta projects. Una vez asignado el nombre sobre esa carpeta le damos click derecho y nos desplegara un menú donde abriremos el menú new->schematic donde nos abre una pantalla como la siguiente:

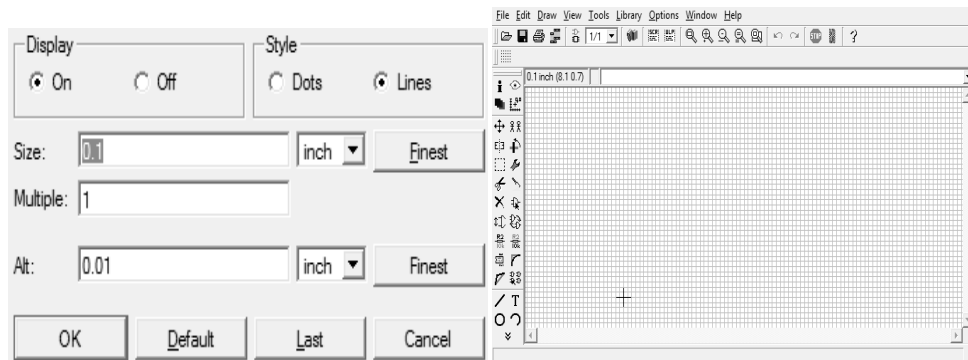


En dicha pantalla podremos hacer el esquemático de nuestra tarjeta o proyecto a diseñar.

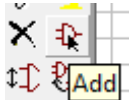
GRID:

Generalmente se activa el grid para poder tener más exactitud en la distancia de los componentes que vamos a conectar, de la siguiente manera:

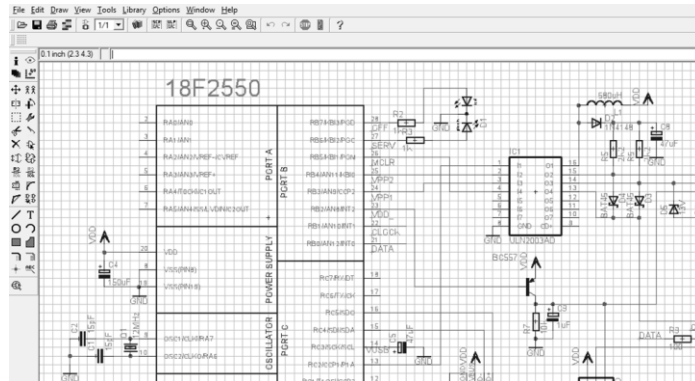
Le damos click en la tercer barra que nos aparece, justo abajo del folder amariilo, o la podemos sacar de la barra view->grid y en la pantalla que nos abra activamos en on el grid y le podemos cambiar las medidas como mejor nos acomodemos estas medidas definen que tan grande son los cuadros, una vez activado nuestra pantalla se verá así:



A partir de esa pantalla podremos empezar a agregar los objetos que necesitamos.

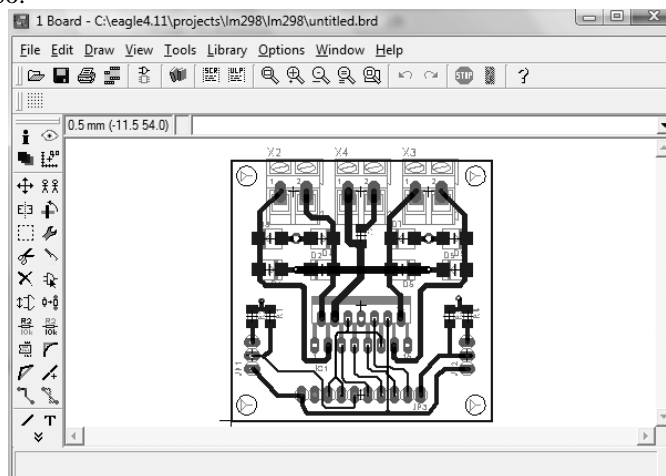


Con el icono add colocado en la barra de herramientas localizada a la izquierda de nuestra pantalla, también podremos encontrar iconos muy útiles, como eliminar, ponerle nombre a nuestros componentes y su valor en caso de que estén repetidos, rotar elementos, mover un conjunto de elementos y no uno por uno, ponerlos en espejo, conectarlos con nodos y/o líneas.



iv CREACIÓN DE LA PLACA (BOARD)

Una vez que vayamos creando el esquemático se irá creando el board, ya que este terminado y guardado podemos empezar a routear o acomodar las pistas de nuestra tarjeta, existe una herramienta que se llama autoruta la cual la podemos seleccionar del menú de herramientas(tolos), escogemos la opción auto y la pestaña route, esto es un poco ineficiente porque hay unas pistas que se enciman o quedan algunas sin conectar y no queda un bonito diseño, lo más recomendable es acomodar las pistas según nos convenga más aunque lleva más tiempo.



APÉNDICE F

El Mplab se utilizó para escribir los programas y compilarlos para que posteriormente se graben en el microcontrolador.

El ccs es un compilador para que en el mplab se pueda programar en lenguaje C, porque Mplab viene predefinido para programar en lenguaje ensamblador.

Y para que se puedan ocupar ambas interfaces se necesita de un plugin.

B. MPLAB Y COMPILADOR CCS PARA WINDOWS

i INSTALACIÓN MPLAB

El programa se puede descargar de la página de www.microchip.com la cual es de uso libre no se necesita licencia.

La última versión para Windows que se usó fue mplab_v8.20.

De la carpeta que bajamos la descomprimos y nos queda una carpeta como la siguiente:

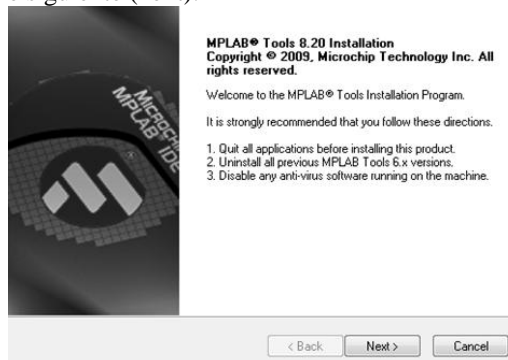


1. Ejecutar el archivo llamado **Install_MPLAB_v820**



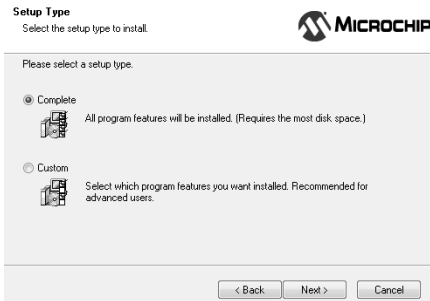
2. Cuando se ejecuta el archivo, comenzará con la instalación de un wizard o ayudante que le hará más fácil la instalación del programa.

3. Posteriormente aparece una pantalla como la que se muestra abajo, para empezar la instalación, lo único que hay que hacer es darle siguiente (next).

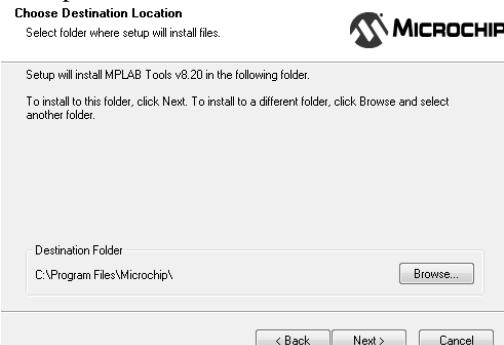


4. Lo primero que nos pide es si aceptamos los términos de la licencia. En caso de que no haya inconveniente en los términos de la licencia seleccionamos Aceptar.

5. Después seleccionamos el tipo de instalación para este caso elegimos el completo, para que no nos haga falta ninguna aplicación.



6. Lo siguiente que te pide es que le des la ubicación en donde se va instalar, por default los instala en C:\archivos de programa, si se desea puede cambiar la ruta en donde instalarlos.



7. En esta pantalla nos vuelve a pedir que aceptemos la licencia software maestro. Aceptamos y le damos siguiente.



8. En el siguiente paso empieza a copiar todos los archivos que se necesiten y le damos siguiente para que se copien



9. Ésta última version del Mplab te pide si quieres instalar otro programa llamado HITECH, al cual le vamos a decir que no, porque es otro aplicación que no vamos a ocupar. Dependiendo el tipo de maquina y sus actualizaciones puede pedirnos que la reiniciemos el equipo para terminar la instalación del MPLAB.

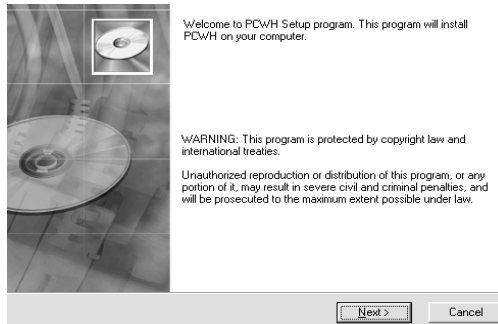
ii INSTALACIÓN DEL CCS Y PLUG IN PARA MPLAB

En este caso el CCS no es software libre, y se tiene que pagar una licencia una vez obtenido el CCS debemos tener una carpeta llamada CCS-PCWHupd

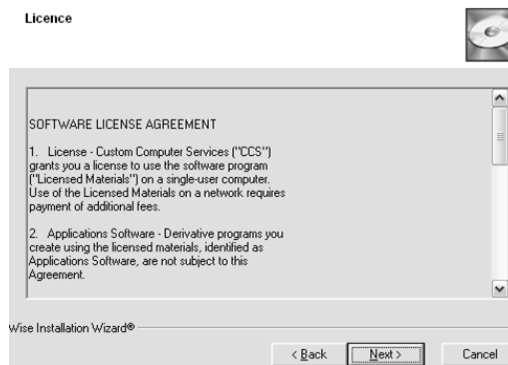
1. Ejecutar el archivo llamado pcwhupd para iniciar con la instalación



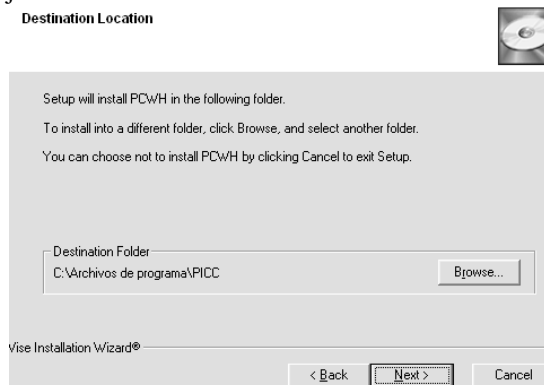
2. Posteriormente nos encontramos con la pantalla que nos guiará a través de la instalación y solo le damos siguiente:



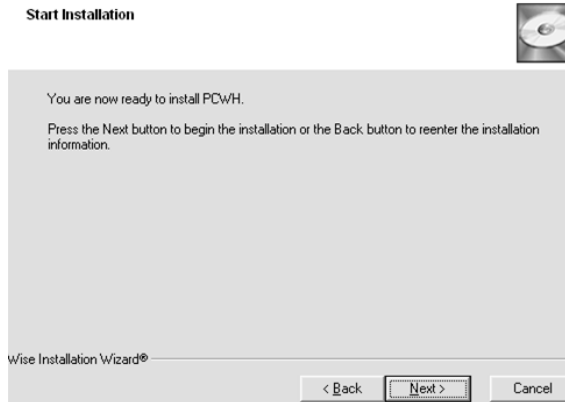
3. En el siguiente paso nos dejara continuar si estamos de acuerdo con los terminos de la licencia si es correcto le damos siguiente



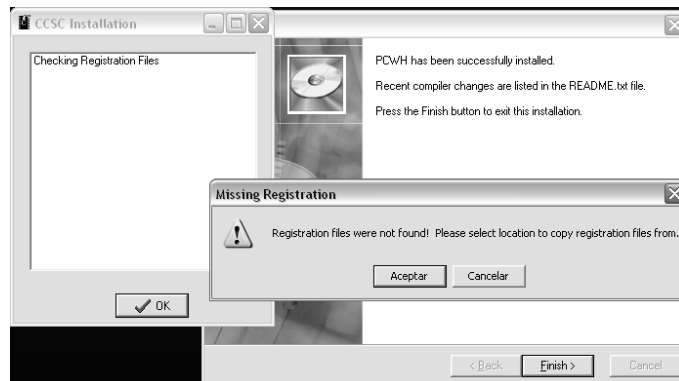
4. En el siguiente punto nos aparece la opción en donde se va a instalar por default lo pone en c:\archivos de programa, en caso de que se quiera cambiar la ruta no importa, solo es necesario recordar en que parte del disco duro lo dejes.



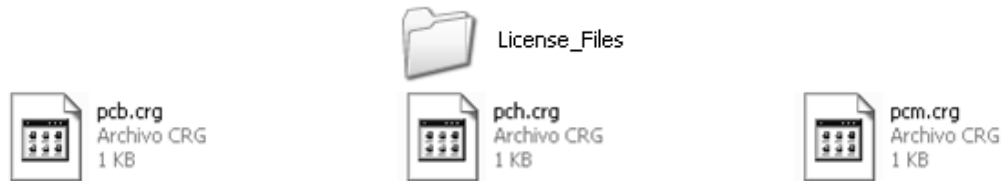
5. Da comienzo la instalación y esta es la pantalla de bienvenida y hay que darle en siguiente:



6. En este punto te pide los archivos de la licencia. En el cuadro de dialogo donde aparece el signo amarillo le damos aceptar y nos abrirá un cuadro donde tenemos que buscar los archivos de licencia los cuales se localizan en la carpeta CCS-PCWHUPD



7. Así que buscamos la carpeta llamada CCS-PCWHupd y dentro de esta localizamos la carpeta llamada License_Files



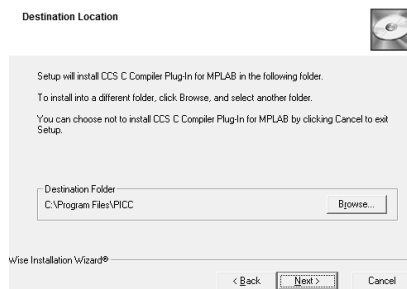
8. De estos 3 archivos que se encuentran podemos seleccionar cualquiera de los tres dándole doble click.

Finalmente queda instalar el plug-in de matlab para que pueda comunicarse con el CCS y poder programar y compilar en lenguaje c.

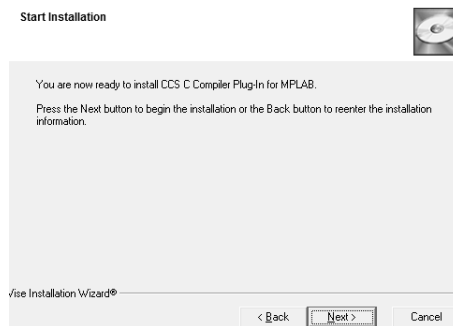
1. De la carpeta donde se encuentran los programas ejecutamos el archivo llamado: setup_mplab_plugin o se puede obtener de la pagina del ccs.



2. En esta primer pantalla de bienvenida le damos siguiente posteriormente se elige en qué dirección se va a guardar, aquí tenemos que verificar que se instale en C:archivos de programa\PICC para poder hacer la comunicación con Mplab y CCS.



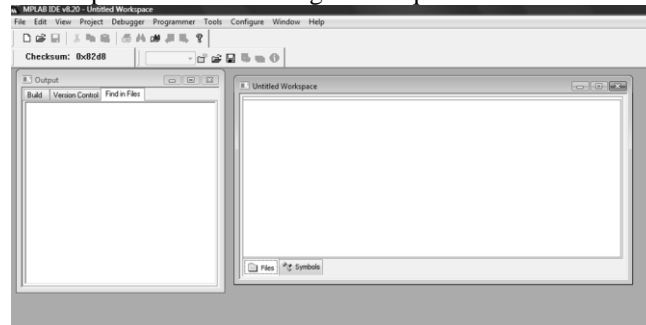
3. La siguiente pantalla nos pide iniciar la instalación por lo que hay que darle siguiente, hasta que nos aparezca el mensaje de finalizado.



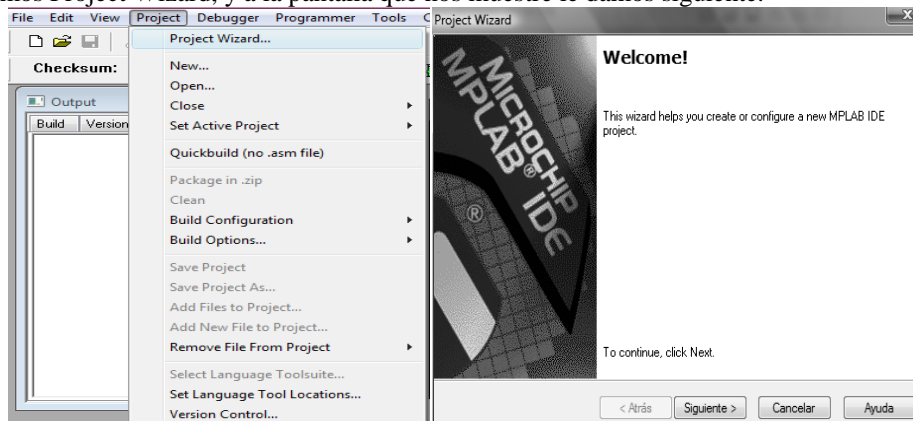
iii CREACIÓN DE UN PROYECTO EN MPLAB:

Por default cuando se instala el programa te crea un icono en el escritorio y abrimos el programa, o en los programas buscamos la carpeta de Microchip en esa buscamos la que diga MPLABIDE v8.20, y abrimos el MPLABIDE.

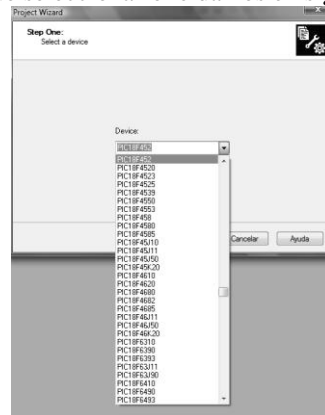
En cuanto abrimos el programa nos aparece una ventana igual a la que se muestra:



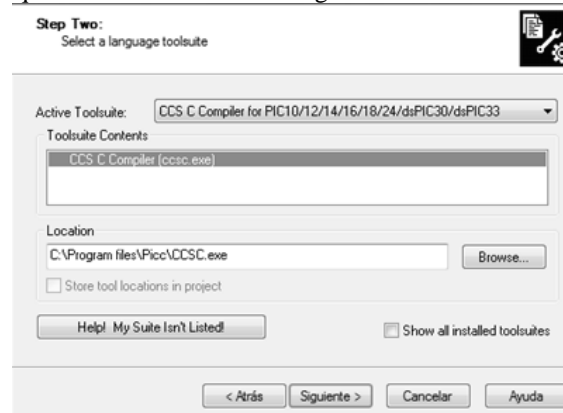
Para crear un programa necesitamos crear un proyecto, para esto nos vamos a la barra de herramientas en Project abrimos Project Wizard, y a la pantalla que nos muestre le damos siguiente.



En la siguiente nos pide seleccionar el dispositivo en nuestro caso vamos a trabajar con el PIC 18F452 que es el cual vamos a seleccionar y después de seleccionarlo le damos en siguiente:



En el siguiente paso nos pide el compilador que vamos a usar en nuestro caso es el CCS C Compiler, y nos fijamos que este correcto en la parte que dice Location(dirección), que la ruta corresponda en donde instalamos el CCS y una vez que sea correcto le damos siguiente:

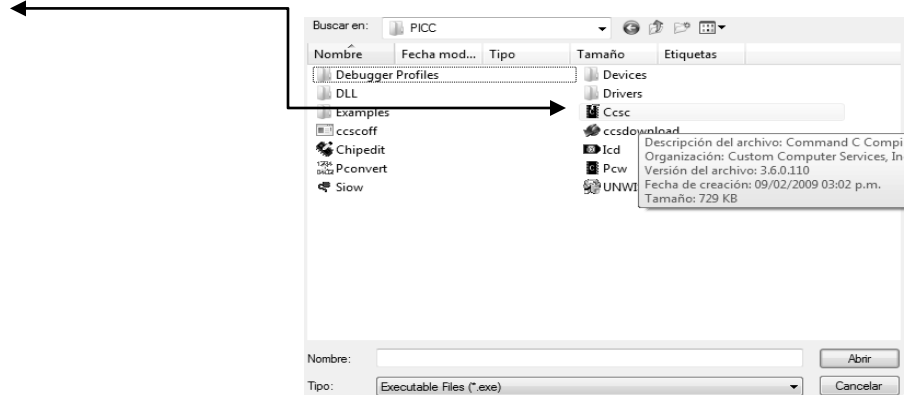


En esta parte como es la primera vez que utilizamos el Wizard tenemos que dar de alta el ccs con el Mplab así que esta primera vez lo más común es que la parte sombreada con azul donde dice:

CCS C Compiler(ccsc.exe) este marcada con un tache rojo, lo que indica que hay que activar el CCS así que solo vamos donde dice BROWSE y buscamos la carpeta donde se instaló el CCS que por default tuvo que ser: c:\archivos de programa\PICC\CCSC.EXE

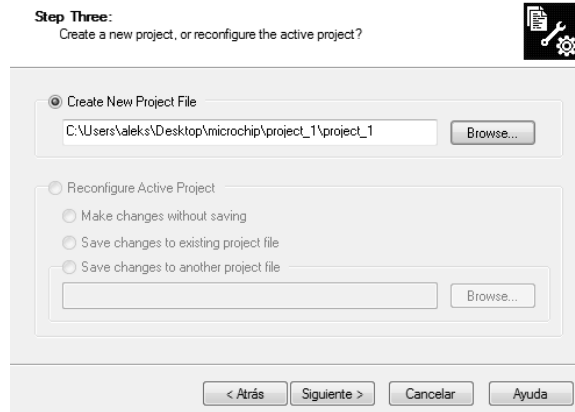
Y escogemos el icono llamado CCSC.EXE y le damos abrir y después solo le damos siguiente y tenemos que darnos cuenta que el tache se haya quitado una vez hecho lo anterior.

CCSC.EXE

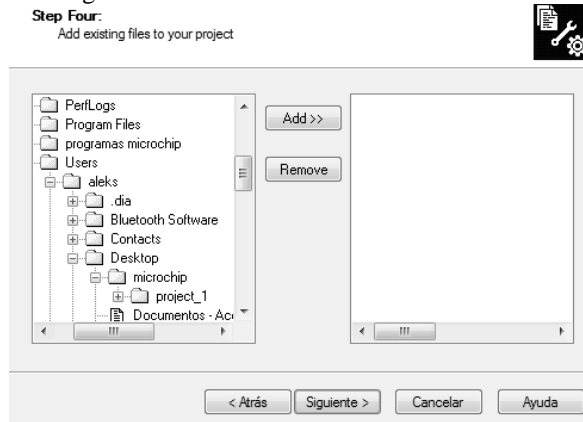


En la siguiente opción nos pide la dirección en donde vamos a guardar el proyecto así como el nombre del proyecto, como una sugerencia es que la ruta donde este la carpeta no sea demasiado larga, porque al compilar nos va a marcar errores porque la ruta es demasiada larga.

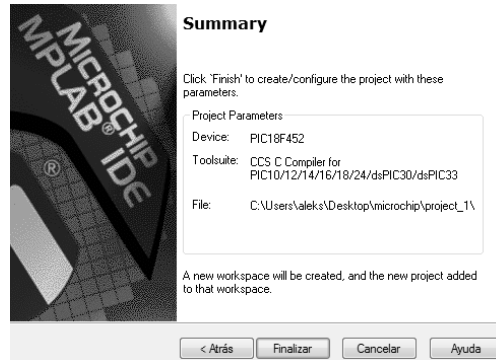
Por ejemplo el proyecto se creó en una carpeta llamada Project_1, que a su vez esta se llama Project_1. Ya que le damos el nombre y la ruta le damos siguiente:



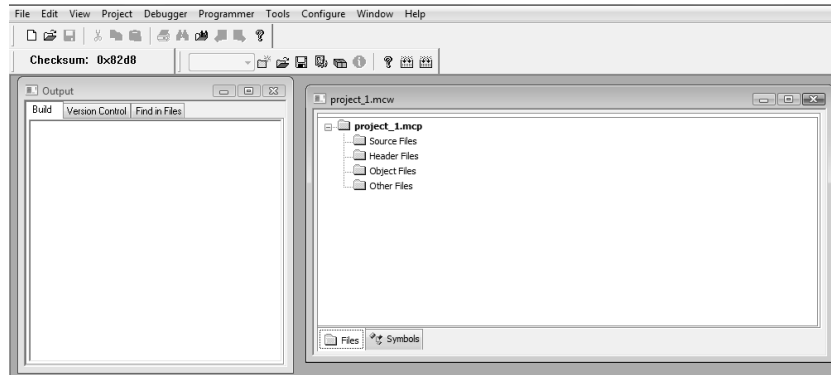
En el siguiente paso podemos ver que en lado izquierdo se ve la carpeta de nuestro proyecto, aquí no modificamos nada solo le damos siguiente:



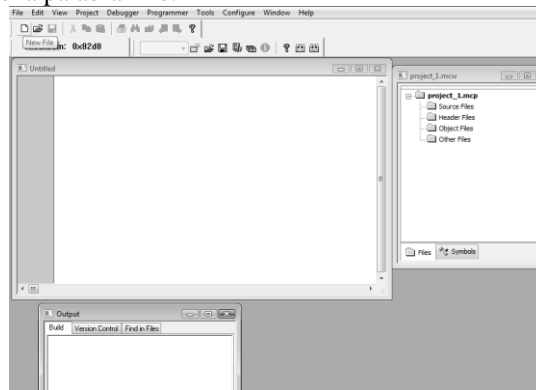
Al encontrarnos con esta última pantalla terminamos de crear el ambiente donde vamos a trabajar.



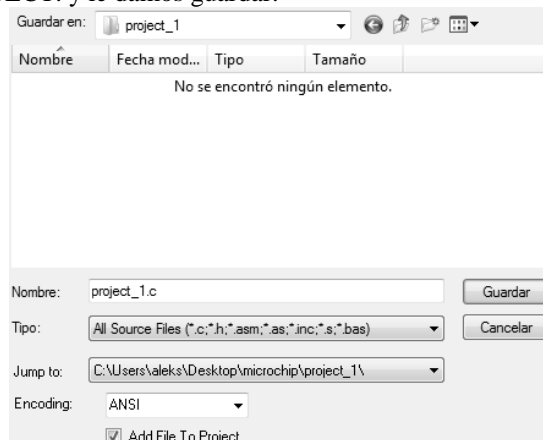
Al darle finalizar vemos esta pantalla, y reducimos un poco estas dos pantallas para seguir viendo lo que estamos trabajando:



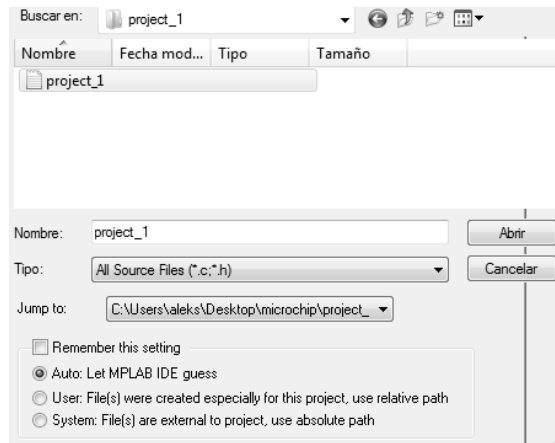
El paso siguiente es abrir una hoja nueva dando en la barra de herramientas en File → new, o en el icono de la hoja blanca que está debajo de la palabra File.



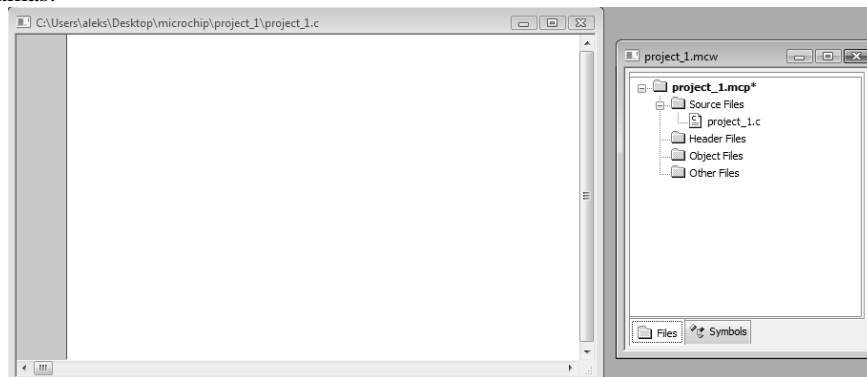
Y en seguida guardamos este archivo, donde creamos el proyecto, en el ejemplo que estamos viendo es en la carpeta Project_1. (File->save as). Y le damos el nombre que se desee pero guardado como punto c en el ejemplo lo guardamos con el mismo nombre de la carpeta Project_1.c y habilitamos la casilla de abajo llamada ADD FILE TO PROJECT. y le damos guardar.



En caso de que no activemos la casilla y solo le demos guardar, podemos agregar nuestro archivo, seleccionando de la barra de herramientas PROJECT-> ADD FILES TO PROJECT y nos sale la siguiente pantalla:



Y solo seleccionamos el archivo que acabamos de guardar y le damos abrir. En seguida de haberlo guardado por cualquiera de los dos métodos anteriores podemos ver que en la pantalla de la derecha en la carpeta SOURCE FILES, aparece nuestro archivo agregado a nuestro proyecto, en caso de que no apareciera hay que checar los pasos anteriores porque es indispensable para que compilen bien nuestros programas.



En este punto ya está listo nuestro ambiente para empezar a programar el PIC. Lo principal que hay que saber es que la primera línea en todos los programas tenemos que incluir la librería que hace referencia al pic18f45(en caso de ocupar otro PIC solo se cambia el modelo).

```
#include <18f452.h> // librería para el manejo del pic18f452
#fuses HS,NOWDT,NOPROTECT,NOLVP //HS:
//NOWDT:
//NOPROTECT:
//NOVOLP:

#use delay(clock=10000000) //se declara la frecuencia del cristal
//se declara la frecuencia del cristal
//que estamos usando en la tarjeta en
//nuestro caso la tarjeta tiene un cristal
//de 10mhz.
```

Los fuses son bits de configuración del micro y vienen explicados en la datasheet de cada micro. Los que tú has puesto en tu ejemplo son:
 -HS: High Speed Crystal. Se suele usar con cristales de más de 4MHz, en este caso se tiene que poner porque usamos un cristal de 10 mhz ya que así está diseñada la tarjeta de desarrollo.
 -NOWDT: No WatchDog. Desactiva el perro guardián.
 -NOPROTECT: No proteger el código. Permite que pueda leerse el programa
 -NOLVP: No Low Voltage Program. Impide que el micro pueda grabarse en modo de bajo voltaje.

```
C:\Users\aleks\Desktop\microchip\project_1\project_1.c*
#include <18f452.h>          /// libreria para el manejo del pic18f452
#fuses HS,NOWDT,NOPROTECT,NOLVP  //HS:
                                //NOWDT:no se usa el watchdog
                                //NOPROTECT:
                                //NOVOLP:

#use delay(clock=10000000)    /// declara la frecuencia del cristal
```

Posteriormente empezamos con nuestro main y ya podemos programar.

```
C:\Users\aleks\Desktop\microchip\project_1\project_1.c
#include <18f452.h>          /// libreria para el manejo del pic
#fuses HS,NOWDT,NOPROTECT,NOLVP  //HS:
                                //NOWDT:no se usa el watchdog
                                //NOPROTECT:
                                //NOVOLP:

#use delay(clock=10000000)    /// declara la frecuencia del cris

//primer programa donde se prende y se apaga un led en el puerto C
// que es donde se localizan los leds

void main(void)
{

//aquí empezamos a programar.....
} // fin de programa
```

iv COMPILACIÓN, ERRORES Y DEPURACIÓN DE UN PROGRAMA

Hasta este punto hemos visto como empezar a programar nuestro PIC. Al igual que cuando programamos en otro ambiente de c, tenemos que poner las librerías comunes como stdio,h, stdlib.h, y siempre deben ir debajo de la línea que hace referencia al PIC, los fusibles y el reloj que estamos usando.

Por ejemplo:

```
#include <18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=12000000)
#use rs232(baud=19200, xmit=PIN_C6,rcv=PIN_C7)
#include<stdio.h>
#include<stdlib.h>
```

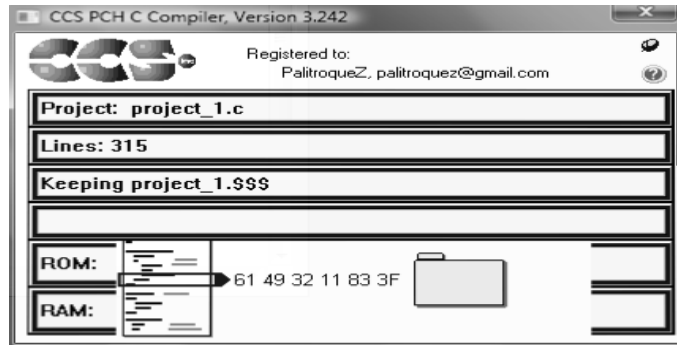
A continuación se muestran estas líneas de código para poder ver como se compila un programa sin errores y con errores,

```
#include <18f452.h>          /// libreria para el manejo del
                                pic18f452

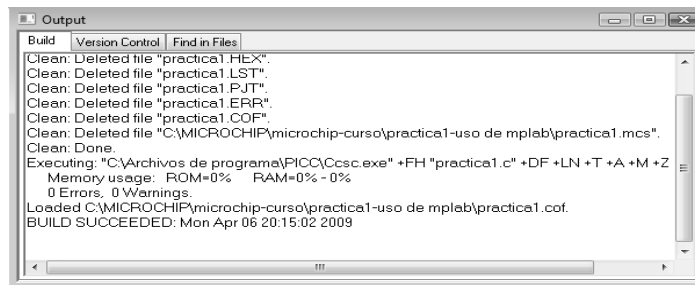
//apartir de aquí podemos incluir librerías.
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=10000000)    /// declara la frecuencia del cristal
//programa de creación un proyecto
void main(void)
{
//apartir de aquí podemos empezar a escribir nuestros programas
} // fin de programa
```

Lo que sigue es ver si está bien nuestro código, es decir vamos a compilar nuestro programa dándole click al icono llamado Build all.

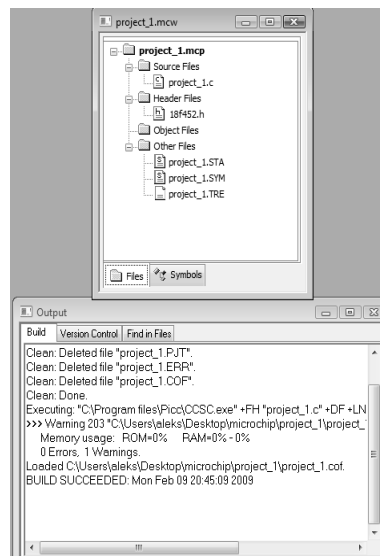
Donde veremos una pantalla cuando se esté compilando el programa:



Donde veremos en la pantalla de output los errores y los warnings que tengamos en nuestro programa, en este caso nuestro programa no tiene ningún error. Por lo que nos creará en la carpeta que estamos trabajando un archivo hexadecimal que es el que tenemos que grabar en nuestro microcontrolador, en caso de tener errores no lo podrá crear.



En el caso de la pantalla que aparecía a nuestra derecha aparecen en la carpeta de OTHER FILES los archivos que se crearon al compilar.



Cuando se tiene un error al darle doble click, te muestra donde se localiza el error en tu programa. Por ejemplo vamos a alterar el código anterior en la línea de fuses y el reloj y en el main.

```
#include <18f452.h> // libreria para el manejo del pic18f452

//apartir de aquí podemos incluir librerías.
#fuses HS,NOWDT,NOPROTECT,NOVP
#use delay(clok=1000000) // declara la frecuencia del cristal
```

```
//programa de creación un proyecto
void main(void)
{
//apartir de aquí podemos empezar a escribir nuestros programas
// fin de programa
```

En seguida se compila como se describe en la parte de arriba y nos sale la siguiente pantalla:

```
Output
Build | Version Control | Find in Files
Clean: Deleted file "practica1.PJT".
Clean: Deleted file "practica1.ERR".
Clean: Deleted file "practica1.COF".
Clean: Done.
Executing: "C:\Archivos de programa\PICC\Ccsc.exe" +FH "practica1.c" +DF +LN +T +A +M +Z +Y=9 +EA
Error 111 "C:\MICROCHIP\microchip-curso\practica1-uso de mplab\practica1.c" Line 3(7,35): Unknown keyword in #FUSES: "NOVP"
Error 99 "C:\MICROCHIP\microchip-curso\practica1-uso de mplab\practica1.c" Line 8(5,76): Option invalid: Expecting "CLOCK"
Error 79 "C:\MICROCHIP\microchip-curso\practica1-uso de mplab\practica1.c" Line 16(0,1): Expect }
4 Errors, 0 Warnings.
Halting build on first failure as requested.
BUILD FAILED: Mon Apr 06 20:35:55 2009
```

En la línea de errores nos muestra la línea donde está el error y te marca la posible falla por lo que hay que saber interpretar estos avisos.

Cuando leemos la pantalla de los errores nos muestra que la palabra NOVP es desconocida, en el siguiente, nos muestra que en la declaración del reloj hay una opción inválida, y en el siguiente nos muestra que falta una llave.

```
C:\MICROCHIP\microchip-curso\practica1-uso de mplab\practica1.c
#include <18f452.h> // libreria para el manejo del pic18f452
//apartir de aquí podemos incluir librerías.
#fuses HS, NOWDT, NOPROTECT, NOVP //HS:
//NOWDT:
//NOPROTECT:
//NOVOLP:

#use delay(clok=10000000) // declara la frecuencia del cristal en Hz
//programa de creación un proyecto
void main(void)
{
//apartir de aquí podemos empezar a escribir nuestros programas
// fin de programa
```

Una vez que se leyeron los errores, al darle doble click en los errores nos regresa a la pantalla de el programa y nos indica la línea donde está el error y empezamos a corregir.

Los errores que nos marca son:

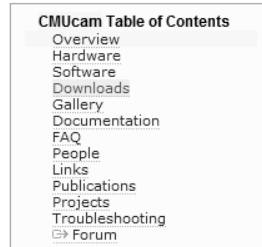
En la parte de #fuses no está bien escrito NOPROTECT, NOVOLP por eso los marca como desconocidos, al igual que en la declaración del reloj (clock=10000000), y en el main falta cerrar una llave. Y una vez corregidos lo volvemos a compilar y ya no nos debe marcar errores.

APENDICE G

G. SOFTWARE PARA EL USO DE LA CÁMARA CMUCAM V3

El software de la cámara CMUCAM puede descargarse del sitio oficial: <http://www.cmucam.org/> de la siguiente forma:

Por lo general aparece en lado derecho de la página un cuadro llamado tabla de contenidos y le damos click donde dice downloads (descargas)



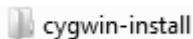
Una vez en la sección de downloads nos vamos a la parte de abajo donde dice Attachments y le damos click a la liga `cmucam3_install_r556` el cual es un paquete completo de todos los programas que necesitamos para poder manejar la cámara, de lo contrario tenemos que bajara uno por uno, en caso de que se desee hacer de esa forma se necesitan bajar de la misma página: CYGWIN INSTALLER, GNU ARM GCC,LPC210X FLASHUTILITY, CC3.

i INSTALACIÓN DEL CYGWIN

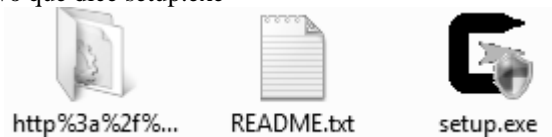
Este programa nos va a servir para poder compilar los programas hechos para la Cmucam.

Una vez bajada la carpeta con todos los paquetes la descomprimimos, y vamos a tener una carpeta llamada `cmucam3_install`, la cual tenemos que abrir y vamos a tener otra carpeta llamada `Windows`, y dentro de ella abrimos otra llamada `cygwin-install`:

`Cmucam3_install\Windows\cygwin-install`



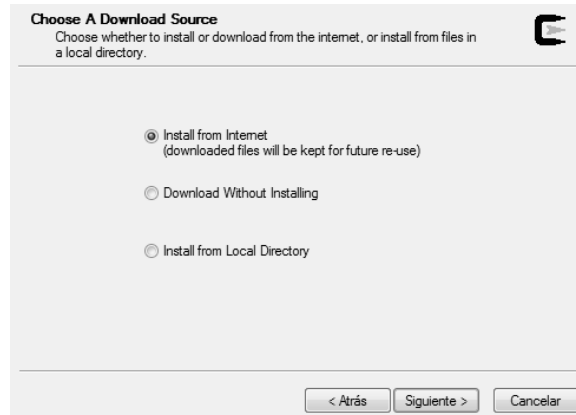
Y de ahí ejecutamos el archivo que dice `setup.exe`



Nos desplegará una pantalla de bienvenida a la cual daremos siguiente:



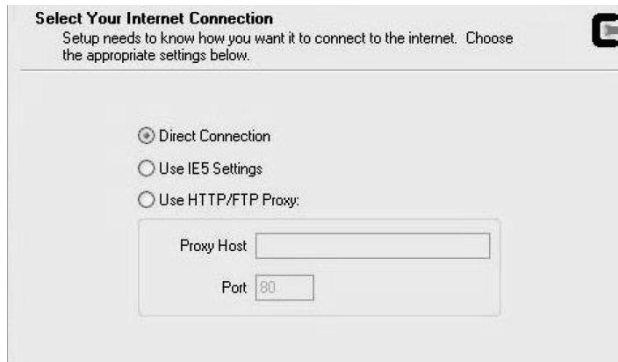
Y nos pedirá la forma en que queremos instalar el programa, y escogemos la primera que es instalarla desde internet y le damos siguiente, de preferencia tiene que ser de banda ancha porque se puede detener la instalación o tardarse mucho.



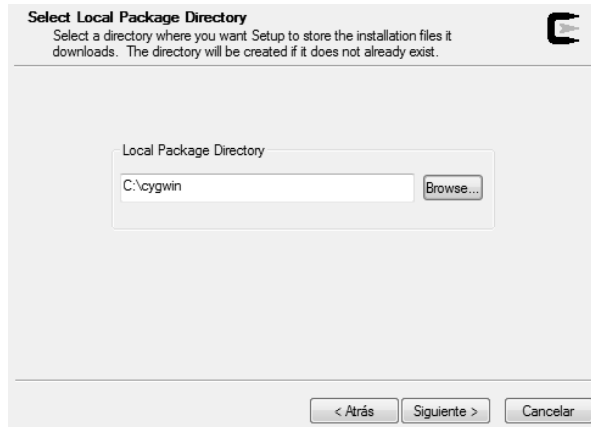
El siguiente paso es seleccionar la ruta en donde se va a instalar y por default te lo instala en c:\cygwin y dejamos así la ruta y le damos siguiente:



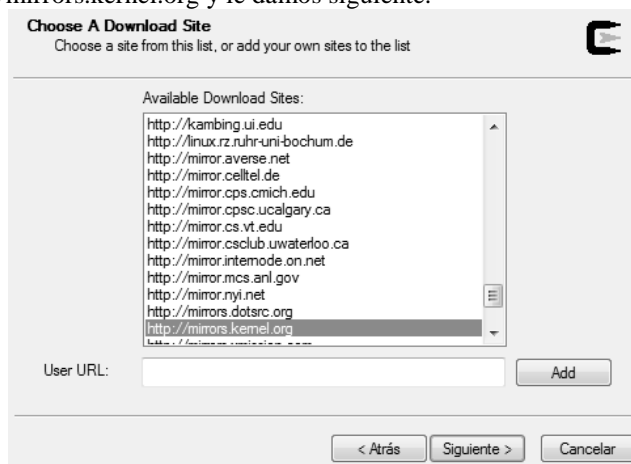
En el siguiente paso tenemos que seleccionar como es nuestra conexión a internet. Este paso no en todas las instalaciones lo requiere.



En el siguiente paso nos pedirá la dirección en donde se van a guardar los archivos de instalación que se van a bajar a continuación. Por lo que se recomienda darle la misma dirección de c:\cygwin

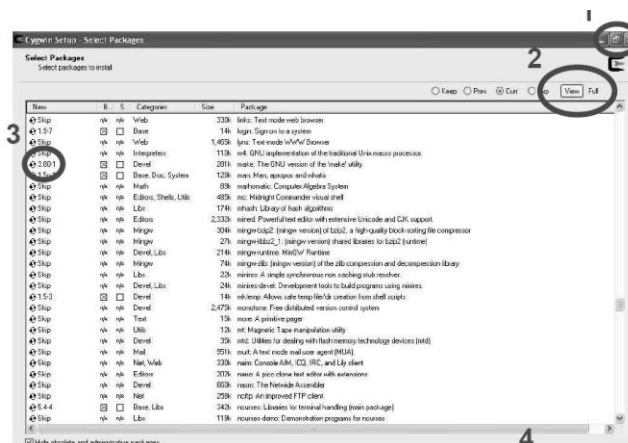


En el siguiente paso nos pedirá escoger un sitio de donde bajar el resto del software que necesita cygwin. Y tenemos que elegir <http://mirrors.kernel.org> y le damos siguiente.



En la siguiente pantalla que nos aparece:

1. primero maximizamos la pantalla
2. damos click donde dice view
3. y después donde aparece skip buscamos los siguientes paquetes y los seleccionamos:
 - make
 - subversión
 - open-ssl



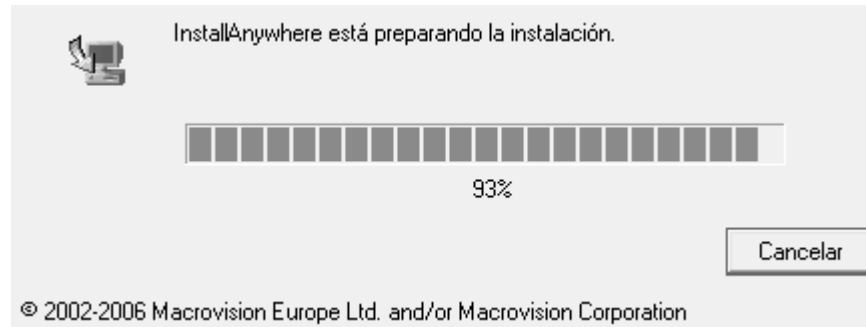
Y a continuación le damos siguiente y vemos como va avanzando la descarga.

Al concluir la instalación nos pide si queremos que cree un icono en el escritorio, y si queremos un icono en el menú de inicio, escogemos el que se desee y le damos finalizar.

ii INSTALACIÓN DEL GNU ARM GCC

En la carpeta Cmucam3_install\Windows\ejecutamos el programa llamado: arm-2006q3-27-arm-none-eabi.exe

Donde veremos una pantalla en donde se prepara la instalación:



Y después a la pantalla siguiente le daremos finalizar.

iii INSTALACIÓN DEL LPC210X FLASH UTILITY

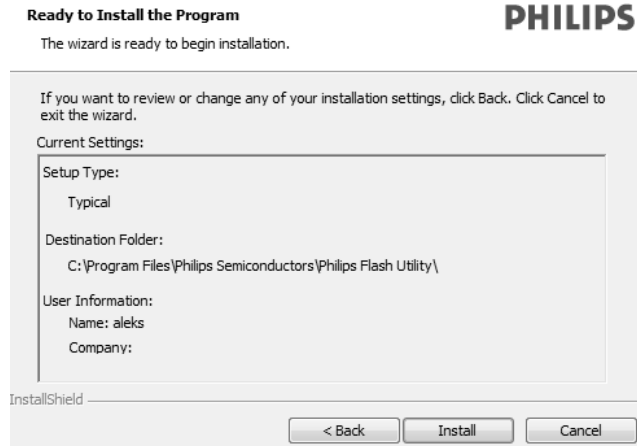
Para la instalación del siguiente programa debemos abrir la carpeta Cmucam3_install\Windows\flash.isp.utility.lpc2000; Y después ejecutamos el programa llamado: Philips flash utility installation.exe



Primero nos va a dar una pantalla de bienvenida y le damos siguiente, el siguiente paso es darle la dirección en donde queremos instalarlo, y le podemos dejar la dirección que da por default que es c:\archivos de programa\philips semiconductor\philips flash utility.



En la siguiente pantalla nos da una aviso de que si estamos seguros en donde se va a instalar y si es el caso le damos siguiente.



Y por ultimo le damos finalizar.

iv INSTALACIÓN DEL CC3

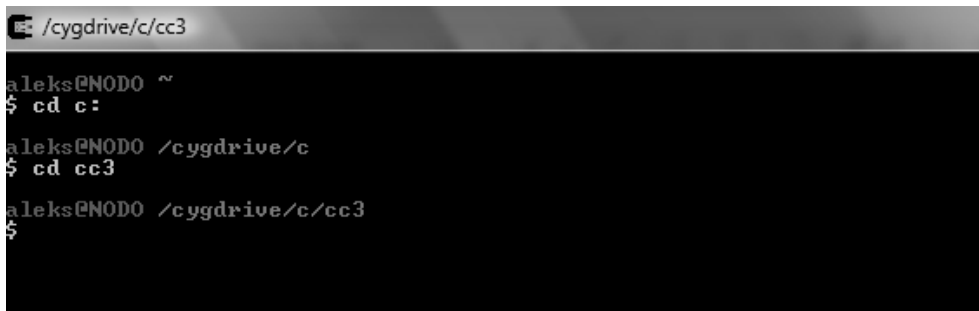
La carpeta CC3 se puede bajar del sitio <http://www.cmucam.org/wiki/Downloads> o de la carpeta que bajamos y descomprimos de la carpeta Cmucam3_install\CC3 y copiamos la carpeta completa cc3 y la pegamos de preferencia en la raíz del disco duro en nuestro caso C:\CC3



Antes de empezar a compilar tenemos que compilar nuestro sistema es decir hacerles un make a las librerías que vamos a utilizar.

Por lo que hay que abrir cygwin y colocarnos en la carpeta de cc3 que se encuentra en disco C de la siguiente manera:

Cd c:
Cd cc3:



Una vez colocados en la carpeta cc3 teclear la palabra “make” y damos enter

```
aleks@NODO /cygdrive/c/cc3
$ make
make[1]: Entering directory `/cygdrive/c/cc3/hal'
make[2]: Entering directory `/cygdrive/c/cc3/hal/lpc2106-cmucam3'
make -f hal.mk
make[3]: Entering directory `/cygdrive/c/cc3/hal/lpc2106-cmucam3'
make[3]: Nothing to be done for `all'.
make[3]: Leaving directory `/cygdrive/c/cc3/hal/lpc2106-cmucam3'
make -C ../../lib
make[3]: Entering directory `/cygdrive/c/cc3/lib'
make[4]: Entering directory `/cygdrive/c/cc3/lib/cc3-ilp'
make[4]: Nothing to be done for `all'.
make[4]: Leaving directory `/cygdrive/c/cc3/lib/cc3-ilp'
make[4]: Entering directory `/cygdrive/c/cc3/lib/jpeg-6b'
make[4]: Nothing to be done for `all'.
make[4]: Leaving directory `/cygdrive/c/cc3/lib/jpeg-6b'
```

Posteriormente nos movemos a la carpeta projects ubicada dentro de la carpeta cc3 y le damos el comando make. Y así sucesivamente en todas las carpetas que se encuentran dentro de la carpeta CC3 donde encontremos el archivo de texto MAKEFILE.

Como en las carpetas:

1. C:\cc3\hal
2. C:\cc3\hal\lpc2106-cmucam3
3. C:\cc3\hal\virtual-cam
4. C:\cc3\lib

Y así ya queda preparado nuestro compilador para nuestros proyectos y así obtener el archivo hexadecimal que le vamos a grabar a la cámara.

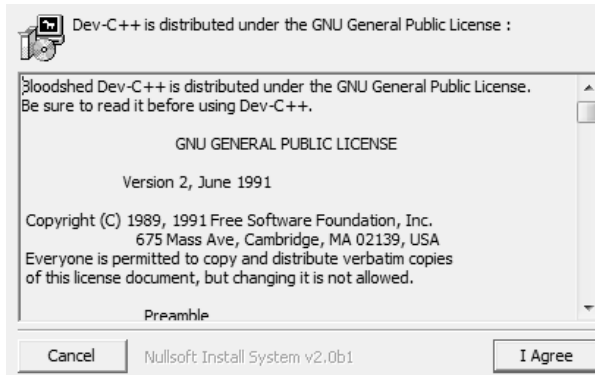
v **INSTALACIÓN DEL PROGRAMA DEV-C++**

Este programa lo ocupamos para poder escribir y editar nuestros programas para la cámara los cuales los escribimos en lenguaje C, pero se puede ocupar cualquier editor de texto.

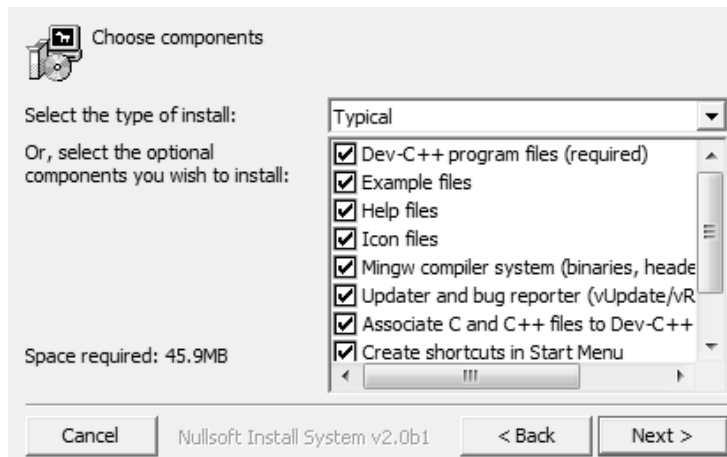
El programa es una versión libre y se puede obtener de la siguiente página:

<http://www.bloodshed.net/dev/devcpp.html>.

De la carpeta que bajemos ejecutamos el programa llamado devcpp4980.exe, y lo primero que nos pide es aceptar los términos de licencia pública.

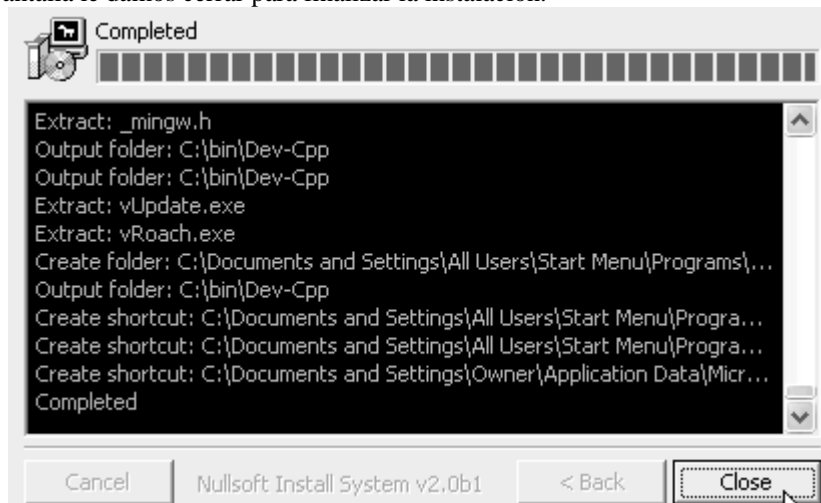


En la siguiente pantalla nos pide escoger los componentes a instalar por default dejamos los que están seleccionados y le damos siguiente:



En la siguiente pantalla escogemos la ubicación donde lo queremos instalar, y una vez elegida le damos en el botón instalar.

Y en la última pantalla le damos cerrar para finalizar la instalación.



El siguiente paso es configurar el ambiente donde vamos a trabajar en DEV-C++, pero como solo se va a ocupar para editar y no como compilador basta con las configuraciones que trae por default.

vi CREACIÓN DE UN PROYECTO PARA LA CÁMARA

Abrimos el programa DEV-C++ y en la barra de herramientas de Archivo abrimos en nuevo un archivo fuente. En la cual nos desplegará una hoja en blanco en la cual empezaremos a escribir nuestro código.

Y como en todo programa primero incluimos las librerías, damos de lata variables globales si es necesario, ponemos las funciones necesarias y después vendría nuestro main con el programa principal.

Para cualquier programa que hagamos debemos guardarlo en una carpeta la cual este dentro de la carpeta projects dentro de cc3.

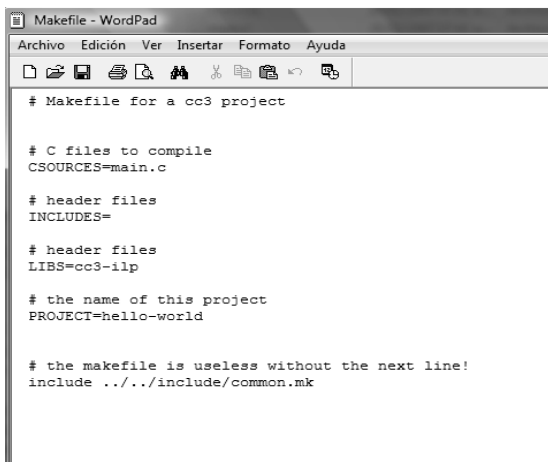
Por ejemplo vamos a crear una carpeta llamada crear_proyecto

C: cc3\projects\crear_proyecto

Una vez creada la carpeta tenemos que darle el nombre de main.c a todos nuestro programas, y no debe de haber confusión con que todos se llamen igual porque cada programa debe estar guardado en una carpeta diferente.

El siguiente paso es abrir la carpeta hello-world ubicada en projects y copiar los archivos llamados: Makefile y lpc2106_cmucam3_buildfiles.

Posteriormente hay que editar el Mekefile de acuerdo a los datos de nuestro programa por lo que abrimos el archivo Makefile con el programa wordpad y nos despliega lo siguiente:



```
Makefile - WordPad
Archivo Edición Ver Insertar Formato Ayuda

# Makefile for a cc3 project

# C files to compile
CSOURCES=main.c

# header files
INCLUDES=

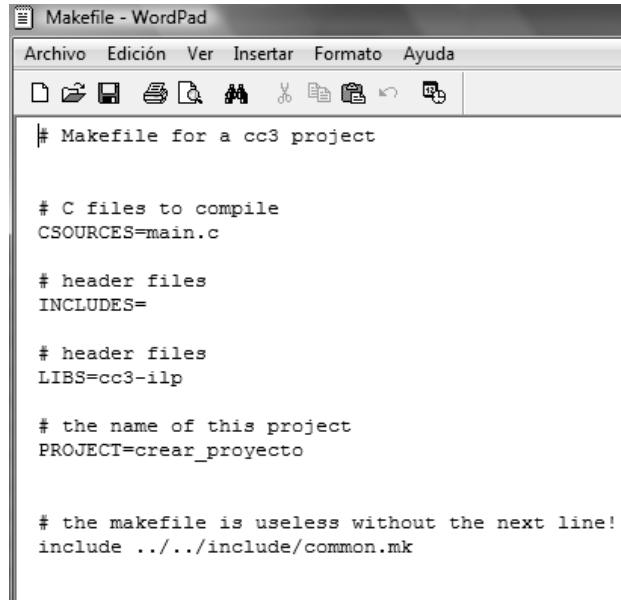
# header files
LIBS=cc3-ilp

# the name of this project
PROJECT=hello-world

# the makefile is useless without the next line!
include ../../include/common.mk
```

Lo que hay que modificar en este archivo es el nombre del proyecto el cual es el nombre de la carpeta donde estamos guardando el programa de la siguiente forma:

```
# the name of this project
PROJECT=crear_proyecto
```



```
Makefile - WordPad
Archivo Edición Ver Insertar Formato Ayuda

# Makefile for a cc3 project

# C files to compile
CSOURCES=main.c

# header files
INCLUDES=

# header files
LIBS=cc3-ilp

# the name of this project
PROJECT=crear_proyecto

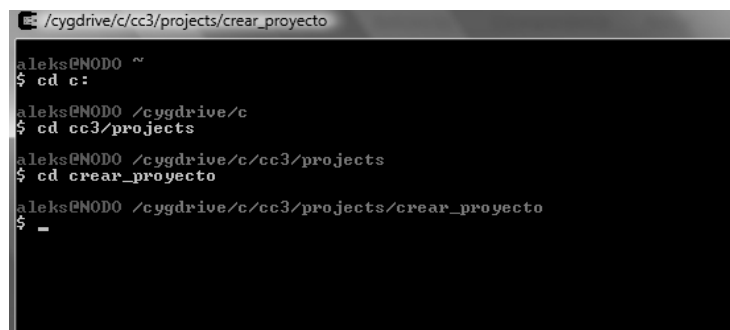
# the makefile is useless without the next line!
include ../../include/common.mk
```

Y lo guardamos, el otro archivo que se copió lpc2106_cmucam3_buildfiles se queda intacto.

vii COMPILACIÓN, ERRORES Y DEPURACIÓN DE UN PROGRAMA

El siguiente paso es abrir cygwin y nos vamos a la carpeta que estamos trabajando en este caso siguiendo el ejemplo es:

```
C:/cc3/projects/crear_proyecto
```

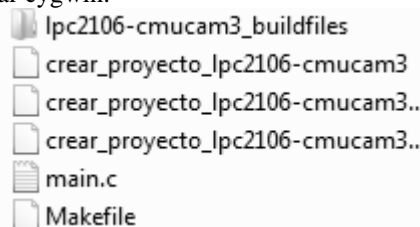


```
/cygdrive/c/cc3/projects/crear_proyecto
aleks@NODO ~
$ cd c:
aleks@NODO /cygdrive/c
$ cd cc3/projects
aleks@NODO /cygdrive/c/cc3/projects
$ cd crear_proyecto
aleks@NODO /cygdrive/c/cc3/projects/crear_proyecto
$ -
```

Una vez ubicados en la carpeta de trabajo escribir make y le damos enter

```
aleks@NODO ~
$ cd c:
aleks@NODO /cygdrive/c
$ cd cc3/projects
aleks@NODO /cygdrive/c/cc3/projects
$ cd crear_proyecto
aleks@NODO /cygdrive/c/cc3/projects/crear_proyecto
$ make
CC      crear_proyecto_lpc2106-cmucam3
OBJCOPY crear_proyecto_lpc2106-cmucam3.hex
text   data      bss      dec      hex filename
80824  2204    9392   92420   16904 crear_proyecto_lpc2106-cmucam3
aleks@NODO /cygdrive/c/cc3/projects/crear_proyecto
$
```

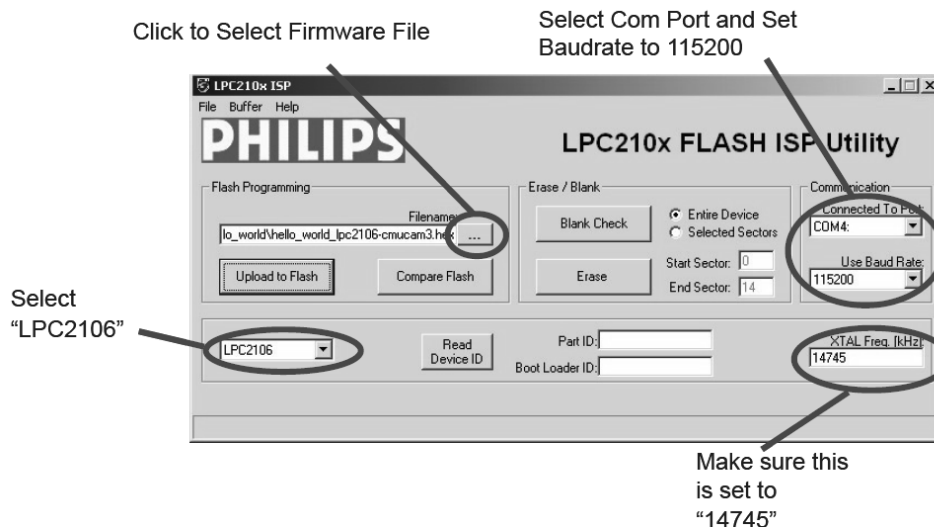
Después de darle make, vemos que si no tiene errores crea diferentes archivos entre ellos el hexadecimal que necesitamos para poder grabarlo en la cámara. Y si vemos la carpeta en la que estamos trabajando aparecen estos archivos que nos acaba de crear cygwin.



En el caso de que nos marque errores nos dirá que línea del código es en la que se encuentra así como el error que podemos tener.

viii DESCARGAR EL PROGRAMA A LA CÁMARA

El siguiente paso es grabarlo mediante el programa LPC200 FLASH UTILITY

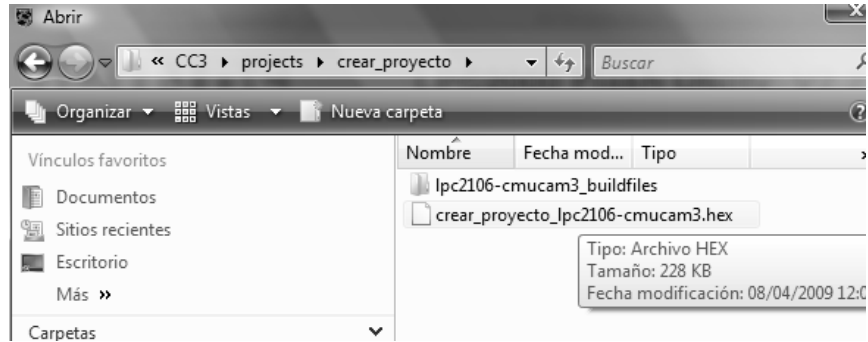


En la parte de comunicación asignarle el com con el que estamos trabajando, en la velocidad de baudaje por lo general se usa 115200.

En dispositivo seleccionamos el LPC2106.

EN XTAL FREQ[KHZ]:14745

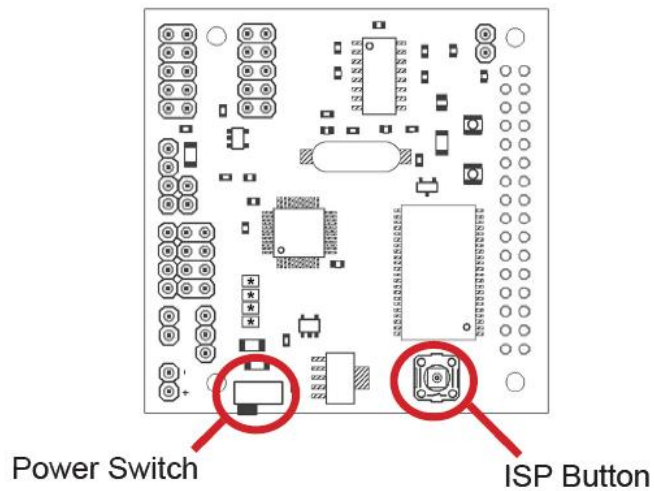
En filename seleccionamos la carpeta siguiendo el ejemplo anterior crear_proyecto y de ahí seleccionamos el hexadecimal que nos creo.



Finalmente en el botón de uploadflash le damos click y nos aparece una pantalla como la siguiente.



Después de esta advertencia tenemos que tener presionado el botón ISP de la cámara mientras presionamos el botón de encendido, para que entre el bootloader y podamos programar.



Finalmente le damos ok a la advertencia que nos apareció y así grabamos los programas a la cámara.

BIBLIOGRAFÍA

LIBROS:

- I. DIGITAL DESIGN PRINCIPLES & PRACTICES. WAKERLY, JOHN F. PRENTICE HALL, 2000.
- II. DISEÑO DE SISTEMAS DIGITALES UN ENFOQUE INTEGRADO. UYEMURA, JOHN P. THOMSON, 2000
- III. AN ENGINEERING APROACH TO DIGITAL DESIGN. FLETCHER, WILLIAM I. PRENTICE HALL, 1990
- IV. SOFTWARE AND HARDWARE ENGINEERING. FREDRICK M. CADY., JAMES M. SIBIGTROGH. OXFORD, 2000.
- V. MICROPROCESSORS AND INTERFACING PROGRAMING AND HARDWARE. DOUGLAS V. HALL. MCGRAW-HILL, 1986.
- VI. MICROCONTROLLER. ARQUITECTURE, IMPLEMENTATION & PROGRAMMING HINTZ TABAK, MCGRAW-HILL 1992
- VII. THE 8051 MICROCONTROLLER ARQUITECTURE, PROGRAMMING & APPLICATIONS. KENNETH J. AYALA, WEST PUBLISHING COMPANY 1991
- VIII. DISEÑO DE MECANISMOS. ANÁLISIS Y SÍNTESIS. ARTHUR G. ERDMAN & GEORGE N. SANDOR. TERCERA EDICIÓN. PRENTICE HALL, ISBN: 970-17-0163-1.
- IX. MECÁNICA DEL AUTOMÓVIL. JM ALONSO PÉREZ.PARANINFO, 1996. ISBN:84-283-1584-1
- X. INTERFACING PIC MICROCONTROLLERS. MARTIN P. BATES 2006. ISBN 0750680288
- XI. DESIGNING EMBEDDED SYSTEMS WITH PIC MICROCONTROLLERS. TIM WILMSHURST
- XII. PIC MICROCONTROLLERS SECOND EDITION: AN INTRODUCTION TO MICROELECTRONICS. MARTIN P. BAT. 2004 EDITOR : NEWNES.
- XIII. ADVANCED PIC MICROCONTROLLER PROJECTS IN C: FROM USB TO RTOS WITH THE PIC 18F SERIES. LUCIO DI JASIO. NEWNES (MARCH 30, 2007). ISBN-10: 0750682922.
- XIV. ADVANCED PIC MICROCONTROLLER PROJECTS IN C: FROM USB TO RTOS WITH THE PIC 18F SERIES. IBRAHIM, DOGAN. NEWNES BOOKS(2008). ISBN: 978-0-7506-8611-2.
- XV. DATA STRUCTURES AND ALGORITHMS IN C++. ADAM, DROZDEK. THOMSON, 2005
- XVI. ESTRUCTURAS DE DATOS Y ALGORITMOS. AHO, A. V., HOPCROFT, J., ULLMAN, J. ADDISON-WESLEY IBEROAMERICANA, 1998
- XVII. CIRCUITOS MICROELECTRÓNICOS. SEDRA, SMITH. OXFORD UNIVERSITY PRESS, 1998
- XVIII. EMBEDDED MICROCONTROLLERS. TODD D. MORTON. PRENTICE HALL, 2001.
- XIX. PROGRAMMING MICROCONTROLLERS IN C. VAN SICKLE TED. HIGHTEXT, 2000.
- XX. ROBOTIC: CONTROL, SENSING, VISION AND INTELLIGENCE. FU, K. S., R. C. GONZÁLEZ Y C. S. G. LEE. MCGRAW-HILL, 1989
- XXI. A MATHEMATICAL INTRODUCTION TO ROBOTIC MANIPULATION. MURRAY, R. M., Z. LI Y S. S. SASTRY. CRC PRESS, 1994.

PAGINAS CONSULTADAS EN INTERNET:

- I. <http://robots.net/robomenu/index-robot.html>www.revistaciencias.com/publicaciones/
- II. http://www.jeuazarru.com/docs/Libro_de_Tesis.pdf
- III. <http://www.interempresas.net/Quimica/Articulos/Articulo.asp?A=1464>
- IV. <http://robotica.uv.es/castellano/home.html>
- V. <http://proton.ucting.udg.mx/materias/robotica/r166/r63/r63.htm>
- VI. http://www.industriaynegocios.cl/Academicos/AlexanderBorger/Docts%20Docencia/Seminario%20de%20Aut/trabajos/2004/Rob%C3%B3tica/seminario%202004%20robotica/Seminario_Robotica/Documentos/DEFINICION%20DE%20LA%20ROBOTICA.htm
- VII. <http://www.robotics.org/>
- VIII. <http://www.ifr.org/>
- IX. <http://www.ieee-ras.org/>
- X. http://cfievalladolid2.net/tecno/ctrl_rob/robotica/
- XI. <http://robotec11.tripod.com/id4.html>
- XII. http://www.tendencias21.net/Los-Robots-Inteligentes-Autonomos-son-la-nueva-generacion_a744.html
- XIII. <http://axxon.com.ar/not/176/c-1760017.htm>
- XIV. <http://robots-argentina.com.ar/not/07/1720035.htm>
- XV. http://www.torneomexicanoderobotica.org.mx/downloads/Open2008_en.pdf
- XVI. <http://www.neoteo.com/microcontroladores.neo>
- XVII. <http://www.scribd.com/doc/19083598/ARQUITECTURA-DE-COMPUTADORAS>
- XVIII. <http://www.conozcasuhardware.com/quees/micro1.htm>
- XIX. http://es.wikipedia.org/wiki/Unidades_de_E/S
- XX. <http://geneura.ugr.es/~gustavo/ec/teoria/tema2-1x2.pdf>
- XXI. <http://www.gui.uva.es/udigital/02.html>
- XXII. www.infowarehouse.com.ve
- XXIII. www.unicrom.com

Referencias:

- [1] <http://www.ilustrados.com/publicaciones/EpyuZIVyZIsWULVYYw.php>
- [2] <http://www.brazorobot.juguetronica.com/Una%20historia%20sobre%20Robots.pdf>
- [3] http://www.industria.uda.cl/Academicos/AlexanderBorger/Docts%20Docencia/Seminario%20de%20Aut/trabajos/2004/Rob%C3%B3tica/seminario%202004%20robotica/Seminario_Robotica/Documentos/DEFINICION%20DE%20LA%20ROBOTICA.htm
- [4] <http://www.roboticspot.com/especial/historia/his2004b.php>
- [5] http://www.roboticajoven.mendoza.edu.ar/rob_tip2.htm
- [6] <http://www.quizma.cl/robotica/tipos.htm>
- [7] <http://www.roboserv.net/asendro-eod/>
- [8] <http://www.irobot.com/sp.cfm?pageid=171>
- [9] <http://www.sandia.gov/media/NewsRel/NR2001/bombbot.htm>
- [10] <http://crapitolio.wordpress.com/2008/02/12/un-robot-desactivador-de-bombas-aplastando-a-un-terrorista/>
- [11] <http://www.oem.com.mx/elsoldetlaxcala/notas/n632178.htm>
- [12] http://www.torneomexicanoderobotica.org.mx/downloads/Open2008_en.pdf
- [13] <http://upload.wikimedia.org/wikipedia/commons/c/cb/Microcontrolador.jpg>
- [14] http://es.wikipedia.org/wiki/Microcontroladores#Familias_de_microcontroladores
- [15] <http://www.olimex.cl/tutorial/tutorial1.pdf>
- [16] <http://www1.microchip.com/downloads/en/DeviceDoc/39564c.pdf>
- [17] http://www2.ate.uniovi.es/fernando/Doc2003/SED/SSP_SPI_BREVE.pdf
- [18] http://www2.ate.uniovi.es/fernando/Doc2003/SED/SCI_asincrono.pdf
- [19] http://www.unicrom.com/Tut_MotorCC.asp
- [20] http://es.wikipedia.org/wiki/Motor_de_corriente_continua
- [21] <http://www.st.com/stonline/books/pdf/docs/1773.pdf>
- [22] <http://www.disam.upm.es/~cybertech/Nacional/Documentos/Talleres/Taller03.pdf>
- [23] <http://www.TiposDePlataformas.php>
- [24] <http://www.disam.upm.es/~cybertech/Nacional/Documentos/Otros/configuracionesmovimiento.pdf>
- [25] <http://www.robotshop.ca/lynxmotion-little-grip-kit-no-servo.html>
- [26] <http://omarsanchez.net/vision.aspx>

- [27] <http://www.cognex.com/ProductsServices/InspectionSensors/default.aspx?id=188>
- [28] <http://www.edmundoptics.com/onlinecatalog/Browse.cfm?categoryid=246>
- [29] <http://www.bannerengineering.com/en-US/products/sub/213>
- [30] <http://www.edmundoptics.com/onlinecatalog/displayproduct.cfm?productID=2680>
- [31] www.cmucam.org
- [32] http://www.convertronic.net/index.php?option=com_content&task=view&id=2010&Itemid=21
- [33] http://es.wikipedia.org/wiki/Modelo_RGB
- [34] www.cimat.mx/~gil/tcj/2002/optica/notas_curso/sesion2.ppt
- [35] <http://www.robotshop.ca/lynxmotion-tri-track-chassis-kit-2.html>