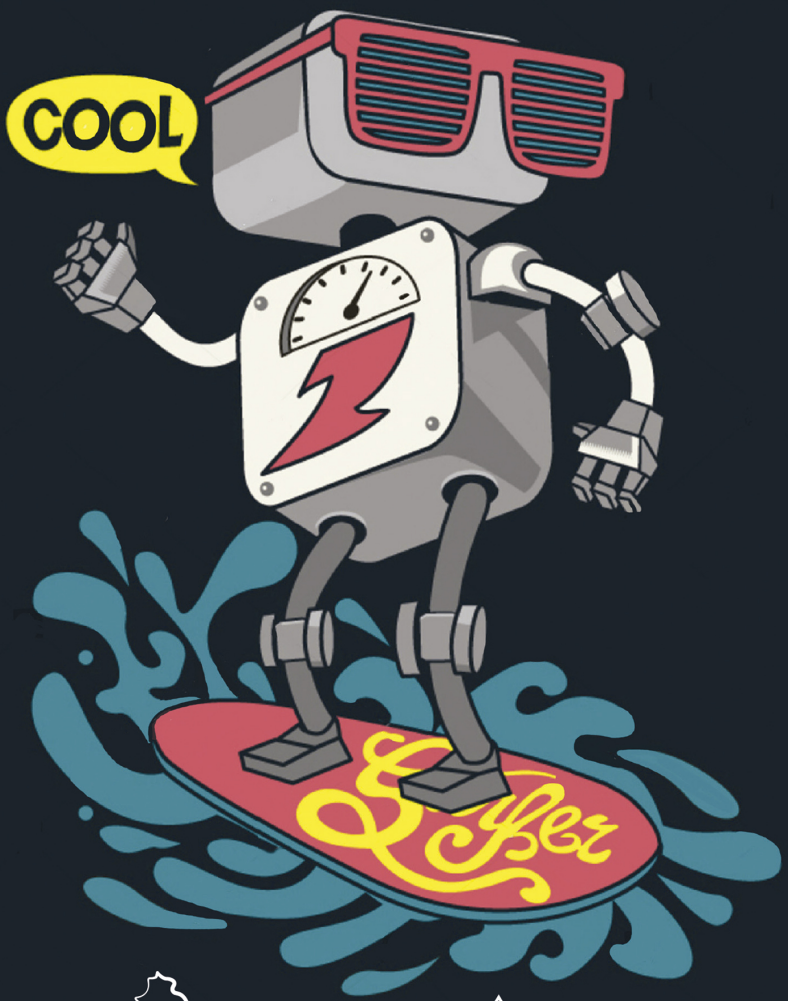
ROBÓTICA

DISEÑO Y APLICACIÓN

Alberto Rocha Díaz



Marcombo

Alfaomega

ROBÓTICA - DISEÑO Y APLICACIÓN

ROCHA DÍAZ ALBERTO (YIYE)

ROBÓTICA - DISEÑO Y APLICACIÓN

ROCHA DÍAZ ALBERTO (YIYE)

Marcombo

Alfaomega

Robótica: diseño y aplicación

Alberto Rocha Díaz

Derechos reservados © Alfaomega Grupo Editor, S.A. de C.V., Argentina

Primera edición: 2019

ISBN: 978-987-3832-36-9

Primera edición: MARCOMBO, S. L. 2020

© 2020 MARCOMBO, S. L.

www.marcombo.com

«Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra sólo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. Diríjase a CEDRO (Centro Español de Derechos Reprográficos, www.cedro.org) si necesita fotocopiar o escanear algún fragmento de esta obra».

ISBN: 978-84-267-2814-2

D.L.: B-26539-2019

Impreso en Servicepoint

Printed in Spain

Al amor de mis hijos, Máximo Rocha Díaz y Juliano Rocha Díaz Presali.

Mis padres Rubén Ariel Rocha (el Vasquito) y Nélida Esther Díaz (la Coca).

Yiye

Agradecimientos: Inés Cabriza, Cristian Goria; Walter Salaberry, Hernán Bosco, José Antelo Solozábal, Marco Domenici, Damián Palacios (músico y productor) con quien estoy grabando los temas que he compuesto para mi banda "Escipión Africano" grupo de Pop Rock donde soy baterista.

Acerca del autor

ROCHA DÍAZ ALBERTO EDUARDO ARIEL (YIYE)

Es Profesor en Disciplinas Industriales Especialidad Computación; Profesor en Disciplinas Industriales Especialidad Electrónica; Profesor en Disciplinas Industriales, con Especialización en Informática Educativa; Especialización en Educación y TIC; Analista de Sistemas de Información; Técnico Superior en Electrónica; Instructor de Formación Profesional Niveles I, II, III.

Es autor del libro *Electrónica E Informática Aplicada*.

Es autor del Proyecto: RxD - 1ROBOTx1DÓLAR. El Proyecto aplica a la Robótica, y propone generar un nuevo paradigma o modelo, creando igualdad de oportunidades, y de acceso a la tecnología:

"Que cada niño en el mundo reciba su Primer Robot, el cual tendrá un coste final de fabricación – Comercialización, de un dólar".

Diseñó la primera carrera corta de Latinoamérica sobre *Robótica, Diseño y Aplicación*.

Desarrolló el primer trayecto en la Escuela de Maestros, *Robótica Práctica y Robótica Aplicada*.

Desarrolló en el Centro de Formación Profesional N˚36, de la CABA, el Primer Trayecto en la Rama de Formación Profesional, sobre *Arduino – Aplicación y Programación*.

Desarrolló el Primer curso de *Programación de Microcontroladores*, en la rama de Formación Profesional en la Provincia de Buenos Aires.

En las Olimpiadas Internacionales de Robótica, Bahía Blanca (2013), obtiene como Profesor a cargo por el Instituto Madero, el Primer Puesto en la modalidad Sumo Menores Inicial, logrando el ascenso a la Liga de Honor, o Liga de Campeones.

Se desempeña como Líder de Proyecto o PM, en el sector privado y la industria.

Es Profesor en los niveles Universitario, Escuela de Maestros, Terciario, Escuelas Técnicas, Formación Profesional, Media, Adultos y Adolescentes.

Coordinador de los distritos Florencio Varela y Quilmes, por el Plan Provincial de Robótica Educativa.

Contenido

Capítulo 1
Introducción a la Robótica 1
1.1 Introducción 1
1.2 Historia de la Robótica 2
1.3 Tipos de robots 4
1.3.1 Definiciones 5
1.3.2 Robots móviles 6
1.3.2.1 Tipos de estructura mecánica del robot móvil ... 9
1.3.3 Robots industriales 10
1.4 Drones 12
1.4.1 Clasificación de los drones 12
1.4.2 Historia 13
1.4.3 Ventajas y desventajas de los drones 13
1.4.4 Nuevos paradigmas de aplicación 13
1.4.5 Componentes de un drone 14
1.5 Robots a implementar en esta obra ... 15
1.5.1 Robots sin lógica adicional 16
1.5.2 Robot seguidor de luz 18
1.5.3 Robot seguidor de línea blanca 19
1.5.4 Robot seguidor de línea negra con lógica de circuitos integrados digitales no programables 20
1.5.5 Robots con lógica de microcontrolador 21
1.5.5.1 Robot seguidor de luz con lógica de microcontrolador en lenguaje C 21
1.5.5.2 Robot seguidor de luz con lógica de microcontrolador en lenguaje Assembler 22

Capítulo 2
Fundamentos y circuitos de aplicación en Robótica 23
2.1 Corriente eléctrica 23
2.1.1 Intensidad de la corriente eléctrica 23
2.1.2 Amperímetro 24
2.2 Corriente Alterna (CA) 25
2.2.1 Frecuencia 26
2.2.2 Período 26
2.2.3 Ciclo 26
2.3 Corriente Continua (CC) 27
2.4 Tensión eléctrica 27
2.4.1 Voltímetro 28
2.5 Resistencia eléctrica 28
2.6 Primera Ley de OHM 29
2.7 Triángulo de la Ley de OHM 29
2.8 Circuito eléctrico 30
2.9 Leyes de Kirchhoff 30
2.10 Configuraciones circuitales básicas 30
2.11 Circuitos en serie 30
2.11.1 Malla 32
2.11.2 Segunda Ley de Kirchhoff (LVK) 33
2.12 Circuitos paralelos 35
2.12.1 Primera Ley de Kirchhoff (LIK) 35
2.12.2 Nodo 36
2.13 Circuitos mixtos 36
2.14 Potencia eléctrica 37
2.15 Triángulo de potencia 38
2.16 Gráfico de sectores 39
2.17 Señales 40
2.18 Electrónica analógica 40
2.18.1 Señales analógicas 40
2.19 Electrónica digital 41
2.19.1 Señales digitales 41
2.20 Sistema de numeración binario 42
2.20.1 Conversión binario a decimal 43
2.20.2 Conversión decimal a binario 43
2.21 Complemento a 1 de un número binario 44
2.22 Niveles lógicos y niveles de tensión . 44
2.23 Resistencias de pull down 45
2.24 Resistencias de pull up 48
2.25 Tierra o ground 49

Capítulo 3
Componentes Empleados en Robótica 51
3.1 Introducción 51
3.2 Tecnologías de Componentes THT y SMD 51
3.3 Resistencias 52
3.3.1 Tipos y características 53
3.3.2 Resistencias fijas 54

3.3.2.1 Resistencias de carbón 54
3.3.3 Resistencias variables..... 56
3.3.4 Resistencias de metal film o precisión 58
3.3.5 Resistencias de encapsulados especiales 59
3.4 Diodos..... 59
3.4.1 Tipos y características..... 61
3.4.2 Diodos LED 61
3.5 Transistores..... 63
3.5.1 Bipolares..... 63
3.5.2 MOSFET..... 65
3.5.3 Fototransistores..... 65
3.5.4 Encapsulados más empleados en Robótica 66
3.5.5 Configuración Darlington..... 66
3.5.6 Método de Testeo de funcionamiento 68
3.5.7 Pin Out de los Transistores a emplear 69
3.5.8 Obtención del valor del βcc o hFE..... 70
3.5.9 Transistor 2N2222A. Características 71
3.5.10 Submúltiplos y notación científica 72
3.5.11 Configuraciones de un transistor..... 72
3.6 Capacitores 73
3.6.1 Tipos y características..... 75
3.6.2 Cerámicos..... 75
3.6.3 Cerámicos multicapa..... 76
3.6.4 Electrolíticos de aluminio 77
3.6.5 Electrolíticos de tantalio..... 77
3.6.6 Modos de funcionamiento 77
3.6.6.1 Capacitor en corriente continua 124
4.6.6.2 Capacitor en corriente alterna 124
3.7 Osciladores..... 78
3.7.1 Oscilador de cristal..... 79
3.8 Sensores..... 79
3.8.1 LDR – Resistencia dependiente de la Luz 80
3.8.2 Sensor óptico por infrarrojo 81
3.8.3 Sensor ultrasónico..... 85
3.8.4 Sensores de implementación con componentes discretos..... 87
3.9 Placas multipropósito o experimentales..... 88
3.10 Zócalos..... 89
3.11 Wire wrapping..... 90
3.11.1 Pinza de wire wrapping 91
3.11.2 Zócalos de wire wrapping 92
3.11.3 Cable de wire wrapping 92
3.12 Borneras para placa de 2 terminales..... 93
3.13 Baterías..... 94
3.14 Conector broche para batería..... 97
3.15 Imán de neodimio..... 97
3.16 Tornillos, tuercas, arandelas, bujes.. 98
3.17 Motores..... 100
3.17.1 Motor de corriente continua..... 100
3.17.2 Servomotor..... 101
3.17.3 Motores para aplicación en Robótica 101
3.17.4 Diferencia entre velocidad y torque..... 103
3.18 Esquinero ángulo..... 104
3.19 Ruedas..... 105
3.19.1 Modelos y características..... 105
3.19.2 Ruedas para motores 106
3.19.3 Omni wheel – Rueda omnidireccional 106
3.19.4 Rueda loca 106
3.19.5 Opciones avanzadas de diseño de ruedas..... 108
3.19.5.1 Modelado e impresión 3D ... 108
3.19.5.2 Diseño de moldes 108
3.20 Patas Metálicas..... 110
3.21 Termocontraibles 110
3.22 Estaño 111

Capítulo 4
Herramientas e instrumental para Robótica 113
4.1 Soldador y estación de soldado..... 113
4.2 Soporte de soldador 114
4.3 Cutter..... 115
4.4 Limas de matricero 115
4.5 Alicate 115
4.6 Pinza de punta plana 116
4.7 Set de destornilladores perilleros..... 116
4.8 Bruselas 117
4.9 Calibre 117
4.10 Torno de mano 118
4.10.1 Mechas..... 118
4.10.2 Discos de corte 119
4.10.3 Accesorios 120
4.11 Instrumental de medición 121
4.11.1 Introducción 121
4.11.2 Tester..... 121
4.11.3 Osciloscopio 124
4.11.3.1 Osciloscopio analógico..... 124
4.11.3.2 Osciloscopio digital 125
4.12 Elementos de seguridad personal.... 125

Capítulo 5
Diseño e implementación de Robots sin lógica adicional 127
5.1 Diseño y montaje de circuitos 127
5.1.1 Introducción 127
5.1.2 Métodos y técnicas de montaje 127
5.1.3 Método de montaje cableado 128
5.1.4 Montaje con wire wrapping 129
5.1.5 Diseño de circuitos impresos PCB 130
5.2 Robot seguidor de luz 131
5.2.1 Introducción 131
5.2.2 Circuito eléctrico 132
5.2.3 Principio de funcionamiento 133
5.2.4 Listado de materiales o BOM 134
5.2.5 Montaje y pasos de implementación 134
5.2.6 Testeo y pruebas de funcionamiento 137
5.3 Robot seguidor de línea blanca 139
5.3.1 Circuito eléctrico 139
5.3.2 Principio de funcionamiento 140
5.3.3 Listado de materiales o BOM 140
5.3.4 Montaje y pasos de implementación 141
5.3.5 Testeo y pruebas de funcionamiento 142
5.4 Otra forma de montaje de Robots 143
5.4.1 Introducción 143
5.4.2 Montaje y pasos de implementación 143
5.4.3 Otras modificaciones 148
5.4.4 Listado de materiales o BOM. Vista Ampliada 151

Capítulo 6
Introducción a la programación en Robótica 153
6.1 Introducción 153
6.2 Algoritmo 154
6.2.1 Definiciones 154
6.2.2 Tres etapas de diseño 154
6.2.3 Tres características 155
6.2.4 Etapas de un algoritmo 156
6.3 Lenguajes de programación 156
6.3.1 Clasificación 156
6.3.2 Lenguaje de máquina 156
6.3.3 Lenguaje de bajo nivel 157
6.3.4 Lenguajes de alto nivel 157
6.4 Compilador, enlazador e intérpretes .. 158
6.4.1 Compilador 158
6.4.2 Intérprete 158
6.5 Paradigmas de programación 158
6.5.1 Clasificación 158
6.5.2 Programación estructurada (PE) .. 159
6.6 Tipos de datos 160
6.7 Desarrollo de algoritmos 164
6.7.1 Diagramas 164
6.7.2 Diagramas de flujo 164
6.7.3 Diagramas de flujo símbolos básicos 165
6.7.4 Diagramas de Nassi-Shneiderman 167
6.7.5 Structorizer Software 169
6.8 Validación o verificación de algoritmos 171
6.8.1 Pruebas de escritorio (PE) 172
6.9 Lenguaje Ensamblador o Assembler .. 172
6.9.1 Set de Instrucciones 173
6.10 IDE - Entorno de Desarrollo Integrado 173
6.10.1 Introducción 173
6.10.2 IDE MPLAB - Assembler 173
6.10.3 IDE MPLAB X – Lenguaje C 177
6.10.4 IDE Dev-C++ 178
6.11 Lenguaje C 180
6.11.1 Historia 181
6.11.2 Biblioteca estándar de C 181
6.11.3 Funciones 181
6.11.4 Estructura básica de un programa en C 182
6.11.5 Aplicación de desarrollo de algoritmos 184

Capítulo 7
Diseño e implementación de robots con lógica de circuitos integrados digitales no programables 191
7.1 Diseño de robots con compuertas lógicas 191
7.2 Compuertas lógicas 192
7.3 Descripción y modo de funcionamiento 194
7.3.1 Compuerta Inversora (NOT) 194
7.3.2 Compuerta OR 195
7.3.3 Compuerta AND 196
7.3.4 Compuerta NOR 197
7.3.5 Compuerta NAND 198
7.3.6 Compuerta XOR 199
7.3.7 Compuerta XNOR 200
7.4 Circuitos Integrados 201
7.4.1 Clasificación 201
7.4.2 Tecnología TTL 202
7.4.3 Tecnología CMOS 205
7.5 Circuito Integrado 40106BE 206

7.6 Circuito Integrado ULN2803 207
7.7 Robot seguidor de línea negra con lógica de circuitos integrados digitales no programables 208
7.7.1 Introducción 209
7.7.2 Circuito eléctrico 209
7.7.3 Principio de funcionamiento 210
7.7.4 Listado de materiales o BOM 211
7.7.5 Montaje y pasos de implementación 212
7.7.6 Testeo y pruebas de funcionamiento 216

Capítulo 8
Microcontroladores y su Aplicación en Robótica 221
8.1 Introducción a los microcontroladores 221
8.2 Microcontroladores y microprocesadores 222
8.3 Arquitecturas de Harvard y Von Neumann 222
8.4 Arquitectura interna 224
8.5 Set de instrucciones CISC y RISC 224
8.5.1 Set de instrucciones CISC 225
8.5.2 Set de instrucciones RISC 225
8.6 Ciclos de máquina vs ciclos de Clk 225
8.7 Pin out 226
8.8 Puertos o ports 226
8.9 SFR – Registros de funciones especiales 227
8.10 Tipos de memoria 227
8.10.1 Mapa de memoria 229
8.11 Elección del microcontrolador 229
8.12 Herramientas de trabajo 233
8.13 Programación. Pasos a seguir 233
8.14 Programador – PICkit 3 de Microchip 234
8.14.1 ICSP – In Circuit Serial Programing 234
8.14.2 Implementación de la ICSP en Robótica 235
8.15 Implementación de placa de entrenamiento 236
8.15.1 Desarrollo de la aplicación 237
8.15.2 Explicación de la sintaxis del programa 240
8.16 Set de Instrucciones del µC 16F84A 242
8.17 Configuración básica de los µC a emplear 242
8.17.1 Configuración Básica del µC 16F84A 242
8.17.2 Configuración Básica del µC 18F4610 243

Capítulo 9
Diseño e implementación de robots con lógica de microcontrolador 245
9.1 Introducción 245
9.2 Robot seguidor de luz con lógica de microcontrolador en lenguaje C 245
9..2.1 Circuito eléctrico 246
9..2.2 Principio de funcionamiento 247
9..2.3 Listado de materiales o BOM 249
9..2.4 Montaje y pasos de implementación 250
9.2.5 Testeo y pruebas de funcionamiento 258
9.2.5.1 Testeo del hardware 258
9.2.5.2 Testeo del software 258
9.2.5.3 Testeo del software en la implementación del hardware 259
9.2.6 Desarrollo de programas 259
9.3 Robot seguidor de luz con lógica de microcontrolador en lenguaje Assembler 269
9.3.1 Circuito eléctrico 269
9.3.2 Principio de funcionamiento 270
9.3.3 Listado de materiales o BOM 270
9.3.4 Montaje y pasos de implementación 270
9.3.5 Testeo y pruebas de funcionamiento 271
9.3.6 Desarrollo de programas 272

Capítulo 10
Arduino - Otra forma de hacer Robótica . 287
10.1 Introducción a Arduino 287
10.2 Processing 291
10.3 Wiring 293
10.4 IDE Arduino 296
10.5 Lenguaje Arduino 298
10.6 Estructura de un programa 298
10.7 Desarrollo de programas 299
10.8 Placas Arduino 302
10.8.1 Arduino Uno 306
10.8.2 Arduino Mega 2560 307
10.9 Shields Arduino 307
10.10 Fritzing Software de diseño de PCB 309
10.11 Aplicaciones de Robótica con Arduino 312

10.11.1 Introducción 312
10.11.2 Arduino robots..... 312
10.11.3 Robots compatibles con Arduino..... 313
10.11.4 Desarrollo empleando Arduino..... 314

Capítulo 11
Impresoras 3D Aplicación en Robótica 317
11.1 Introducción 317
11.2 Tipos y características..... 318
11.2.1 Cartesianas 318
11.2.2 Delta 319
11.3 Electrónica - Hardware 320
11.4 Tipos de archivos de trabajo 321
11.5 Software empleado en impresión 3D 323
11.6 Diseño e impresión de componentes y chasis..... 324

Capítulo 12
Robótica Avanzada - FPGA 333
12.1 Introducción 333
12.2 FPGA – Placa a emplear..... 334
12.3 IDE – ISE..... 334
12.4 Descarga del programa 335
12.5 Creación de un proyecto 335
12.5.1 VHDL - Lenguaje..... 341
12.5.2 Estructura de un programa..... 343
12.5.3 Comportamiento lógico del componente 344
12.5.4 Sintaxis y sintetización del programa..... 344
12.5.5 Vista de esquemático..... 345
12.5.6 Simulación del proyecto..... 346
12.5.7 Generación de archivo ejecutable..... 352
12.5.8 Conexión de la placa Basys 2 a la PC 358

Introducción

Hemos sido invitados a participar de momentos mágicos en la historia de la humanidad y la tecnología, con el hombre y los robots espaciales; sean humanoidades de apoyo y *rovers*, abocados a la exploración espacial. La proyección e inminente construcción de ciudades bajo el mar, los micro robots en aplicaciones de medicina, los robots médicos que operan con precisión superior a la del hombre, los drones en aplicaciones de exploración; los coches no tripulados, voladores, anfibios, las mulas mecánicas, los robots futbolistas, concertistas, mascotas, e insectos, de servicio, o industriales.

La mecatrónica ha llegado y deja su huella en el día a día. La industria de la electrónica extiende sus límites sobre la limitación física de los componentes, impulsada por la evolución constante de la ingeniería de componentes y materiales. La industria del software recibe los desarrollos de la electrónica. Ellos se complementan para desarrollar potentes aplicaciones a nivel de software y hardware, como las FPGA, microcontroladores, los sistemas de Arduino, y recursos de aplicación para la robótica, como la impresión 3D.

El género ciencia ficción nacido en las primeras décadas del siglo pasado, admira y celebra esta maravilla, que hoy es una realidad: la llamada Robótica. Los invitamos en esta obra a formar parte de ella, a través de la integración de saberes, como: la electrónica, la informática, la electrotecnia, la mecánica, la física, la neumática, aplicadas al diseño e implementación de robots.

Yiye Rocha Díaz

CAPÍTULO 1
INTRODUCCIÓN A LA ROBÓTICA

1.1 INTRODUCCIÓN

La Robótica es una ciencia que integra saberes, como la Electrónica, Informática, Electrotecnia, Mecánica, Física, Inteligencia Artificial (IA), y el Modelado e Impresión 3D. Dependiendo de la necesidad e integración de estas áreas, será el tipo de Robot a implementar.

Es así como la robótica es una ciencia integrante de saberes, dado que etimológicamente ciencia proviene del latín *Scientia* (conocimiento), y alude al conocimiento exacto y razonado de ciertas cosas. A su vez, *Scientia* deriva del verbo *Scire*, que se traduce como saber.

Los **robots móviles** son el tipo de robots a implementar en esta obra, ya que forman la categoría de robots a la que acceden tanto aficionados como instituciones educativas durante su trabajo introductorio en el campo de la robótica.

1.2 HISTORIA DE LA ROBÓTICA

Al abordar la historia de la Robótica, se observa que es un tema tan interesante como extenso. Su enfoque depende del área específica de trabajo en la misma, tal cual se verá al conocer los distintos tipos de Robots.

Nuestro recorrido se remontará a la ciudad de Alejandría, donde Herón en el siglo primero de nuestra era, seguramente inspirado en el trabajo de Filón de Bizancio, quien tres siglos antes escribió, entre otros, un compendio de mecánica, abordó trabajos con motores neumáticos, y otros dispositivos cuya fuente de energía era el agua en combinación con el aire.

Dado entonces que estos visionarios no disponían de la corriente eléctrica como fuerza motriz, aprovecharon el potencial o presión del agua, para dar animación a variados tipos de figuras y máquinas. Algunas de estas como en el caso de Herón, se emplearon como elementos de entretenimiento o decoración, interactuando también con las propiedades del aire. Destacan en su trabajo la hidráulica y la neumática, tecnologías con gran presencia en la actualidad.

Este mismo principio se empleó a través de varios siglos, lo cual se refleja en el trabajo de Al Jazari, en el 1206 de nuestra era. Fue autor de un tratado que llega a nuestros días, sobre dispositivos mecánicos, llamado *El libro del conocimiento de dispositivos mecánicos ingeniosos*. No es casualidad, según se verá más adelante, que una de las obras más recordadas de Al Jazari, sea *El reloj del castillo*.

Luego, durante el Renacimiento en el siglo XVI, destacará para otro genio, Leonardo da Vinci, la concepción y el desarrollo de Autómatas. Autómata, proviene del latín *Autómata*, y su significado es máquina que imita los movimientos de un ser animado. También está asociado a dispositivo que tiene dentro de sí, un mecanismo que le permite ciertos movimientos. Es importante destacar que los autómatas son los antecesores de lo que hoy se llama robots. Pero se debe diferenciar la autonomía de estos, entre sus orígenes de movimiento en función del agua, y su operación mediante principios y partes mecánicas. Como llega a nuestros días la anécdota de *El León*, regalo otorgado por Leonardo al Rey de Francia, Francisco I en el año 1515. El manuscrito original de este autómata no ha sido hallado, pero pudo ser implementado de todas formas, a modo de muestra en nuestros días.

Sin embargo, como suele ocurrir en la ciencia, escalando en los grandes acontecimientos ya producidos o presentes, existe quien trasciende a la visión y el talento entre los talentosos, y deja su huella en la historia. Este es el caso de Jacques de Vaucanson, ingeniero e inventor francés, a quien se le atribuye, entre otros desarrollos, el primer robot.

Es actual el término robot, puesto que, en 1737, año en que Vaucanson desarrolla su obra, conocida como *El Flautista*, este término no existía aún.

Este Autómata superaba la gran cantidad de dispositivos mecánicos desarrollados para la época.

Luego como fruto de sus años de formación y estudio específico, desarrolla otras dos obras geniales: *El Tamborilero*, y su tal vez considerada gran obra maestra, *El Pato*. Esta última poseía como emulación el sistema digestivo de un pato real. El

autómata consumía un grano, lo digería, y reproducía el proceso natural de la digestión del animal.

Resulta interesante profundizar en este punto con el objetivo de analizar dos factores históricos, que marcarían un antes y un después en el desarrollo de autómatas y en el advenimiento de la robótica actual.

Para el primero de ellos, se debe destacar que la producción de autómatas a partir del siglo XVIII surge de una ciencia, que llevaba miles de años de evolución, La Relojería. Esto resulta claro si se considera que el empleo racional del tiempo era fundamental para la evolución y el control en todas las sociedades, y a esto lo entendieron tanto egipcios, como griegos, y quienes pueden ser considerados los padres de la Ingeniería, los romanos.

El segundo gran factor histórico acontece también en el siglo XVIII: la Revolución Industrial, que surge en Inglaterra, y luego ha de extenderse al resto de Europa y los Estados Unidos. Desde el Neolítico, la humanidad no había experimentado una transformación tan grande en su economía, tecnología, formas de producción, aparición de vacunas, crecimiento demográfico, ciudades, sus obras de infraestructura y, un factor clave dentro de las transformaciones sociales; el fortalecimiento de la Burguesía.

Fue la clase acomodada o rica la responsable de solicitar a los grandes y destacados ingenieros y artesanos de la época, el diseño y construcción de los autómatas. En la actualidad, se conservan y destacan obras como, *El Dibujante* de Pierre Jaquet Droz; *El Teatro Mecánico* en el palacio de Hellbrunn, Austria. Y la joya orfebre de Cox-Marlin, el *Cisne de Plata*.

Otro punto de análisis surge en el razonamiento de que los autómatas inicialmente formaban parte de las escenas presentes en los grandes relojes de las ciudades. Posteriormente, se concibió trabajar en la implementación o representación de estos, a mayor escala.

La mecánica y la micromecánica, presentes en dispositivos tales como los relojes, fueron para la tecnología aplicada a los autómatas, lo que la microelectrónica significa tecnológicamente hoy en día para la Robótica. Del mismo modo, fueron necesarios conocimientos sobre Anatomía. Así, destacan los estudios realizados por Vaucanson para el diseño de sus obras, o las del propio Leonardo.

La Homeostasis se aplicó como principio para la regulación de la temperatura, o forma de equilibrio dinámico, tal cual ocurre con los seres humanos. Estos conocimientos son similares a los que se emplean dentro de muchas de las aplicaciones de lo que se verá más adelante; la IA o Inteligencia Artificial.

Ya en el siglo XX, Karel Capek, dramaturgo y novelista checo, acuña la palabra robot, en su obra R.U.R., acrónimo de *Robots Universales Rossum*, escrita en 1920. Etimológicamente, la palabra robot proviene del vocablo checo, *robota*, y aplica a servidumbre, trabajo forzado o esclavitud.

Nuevamente es muy interesante aquí entender el contexto histórico que influye en las personas, a la hora de representar o transmitir sus ideas. Las imágenes que construye Karel desde su formación de base filosófica están ligadas a acontecimientos históricos producidos en su sociedad, y en el resto de Europa. Como el Ludismo inglés, movimiento caracterizado por la oposición de los trabajadores, a la introducción de maquinaria moderna en la producción, al ver a las mismas como una competencia y futuro reemplazo. En este movimiento social, eran los trabajadores quienes rompían las maquinas, en contrapartida del desenlace en la novela R.U.R. en la que, cobrando alma, los Robots comienzan su guerra contra la humanidad. Este tema ha de ser abordado por Issac Asimov, en sus tres leyes de la Robótica. Vale aquí analizar el significado del género en que están concebidas estas obras, la Ciencia Ficción.

La Ciencia Ficción es un género con una visión futurista e imaginaria, basada en los posibles logros de la tecnología, y las consiguientes transformaciones sociales, donde las temáticas abordadas son variadas: Robótica, exploración planetaria, mutaciones.

Uno de sus primeros grandes representantes fue Julio Verne, quien escribió, entre otras: *Veinte mil leguas de viaje submarino* del año 1870. Verne es considerado junto a Herbert George Wells, padre de este género.

Una de las grandes obras de la Ciencia Ficción del siglo XX, vaticinio de la interacción en las mega ciudades, entre humanos, maquinarias y androides, es la película Alemana *Metrópolis*, del año 1927, dirigida en los estudios UFO, por Fritz Lang, considerada obra del mundo por la UNESCO.

Dentro de este mismo género, es uno de sus mayores autores, Isaac Asimov, quien acuña la palabra robótica. Siendo celebres también, sus *Tres Leyes de la Robótica*, que son empleadas como paradigma en Inteligencia Artificial y en el desarrollo de robots.

Las tres leyes de la Robótica postulan:

1. Un robot no puede hacer daño a un ser humano o, por su inacción, dejar que un ser humano sufra daño.
2. Un robot debe obedecer las órdenes que le son dadas por un ser humano, excepto si estas órdenes entran en conflicto con la Primera Ley.
3. Un robot debe proteger su propia existencia, hasta donde esta protección no entre en conflicto con la Primera o Segunda Ley.

1.3 TIPOS DE ROBOTS

En la introducción del capítulo, se describió que la complejidad y tecnología a emplear para el diseño y aplicación de robots, sería dependiente del tipo de robot

a implementar. A continuación, se aborda el trabajo en la categoría de robots móviles, expuesta como la categoría de robots para aplicar en esta obra.

Por su importancia en el desarrollo de la robótica, son presentadas además las categorías de robots industriales y drones.

Es importante también destacar, respecto a la transversalidad o integración de las distintas ingenierías o áreas del conocimiento, el advenimiento de la Ingeniería en Mecatrónica, que conjuga las Ingenierías Electrónica, Informática y Mecánica. Uno de sus campos de acción es la robótica, y aplica también al desarrollo de dispositivos y procesos Electromecánicos Microcontrolados, con el fin de brindar soporte a aquellas áreas de la Industria, que han evolucionado fuertemente a la par de los ciclos de la tecnológica. Este es el caso de las automotrices, de las cuales se podría predecir que surja en breve, la próxima revolución tecnológica, social y económica del mundo, dados los logros alcanzados en aplicaciones consideradas como ciencia ficción en otros tiempos, y donde hoy la realidad muestra en las ciudades del primer mundo, un escenario compuesto por el histórico autor principal en el sistema de manejo, el chofer, y del otro lado, una de las nuevas maravillas o joyas de la tecnología moderna, los ***driverless car***, o **coches sin conductor**.

Podría también surgir la pregunta: ¿cuál es la mejora o beneficio en la introducción de estos coches? La respuesta nuevamente está en la tecnología misma, o en estas bellas obras mecatrónicas, ya que estos coches cuentan con sistemas de microcontroladores para la toma de decisiones. Los cuales pueden operar en el orden de millones de instrucciones por segundo, aplicando junto a la electrónica de sus sistemas, capacidades de respuestas extremadamente rápidas y confiables.

Se deja también como introducción para el conocimiento del lector, los ***flying cars*** o **coches voladores**, y los ***water cars*** o *coches de agua*, entre otros.

Llegados hasta aquí, es importante definir que es físicamente un Robot.

Del mismo modo, y a fin de generar un conocimiento integrador y formativo, se van a desarrollar otras definiciones, a fin de incorporar conceptos técnicos, sobre distintos principios de funcionamiento. Vale destacar que las mismas, no responden a un significado simplemente enciclopedista, sino que poseen, como valor agregado, el saber erudito; es decir, aquel que se le añade a una definición, desde el conocimiento amplio, práctico y profundo, sobre los temas tratados.

La norma ISO 8373:2012, define robótica y robot móvil de la siguiente forma.

1.3.1 DEFINICIONES

ROBÓTICA: Ciencia y práctica de diseñar, fabricar y aplicar robots.

ROBOT MÓVIL: Robot capaz de viajar bajo su propio control.

Ahora bien, se puede aplicar con basamento teórico, y afianzados en nuestra experiencia práctica, la siguiente definición de robot para nuestros dispositivos electrónicos:

Robot es Un dispositivo integrador de diversas tecnologías, cuya integración, es dependiente de su complejidad o categoría, presentando como mínimo, las tres etapas básicas de un sistema de control o informático, entrada (sensores), procesamiento (microcontrolador, lógica de compuertas, conmutación por transistor o relé), y salida (actuadores o motores).

DISPOSITIVO: Conceptualmente es un aparato.
En general, se emplea esta denominación para conferirle un léxico más técnico.

APARATO: Conjunto de componentes o piezas, dispuestos sistemáticamente, de forma tal, de cumplir funciones específicas.

MECANISMO: Del latín *mechanisma*, conjunto de piezas, que cumplen determinadas funciones, mediante sistemas coordinados de encastre o engranaje, transmitiendo fuerza y movimiento que resultan necesarios para producir una acción o trabajo.

ENGRANAJE: Mecanismo, empleado con el fin de transmitir, dentro de una máquina, una relación de potencia, de un componente a otro.

ENCASTRE: Es la introducción de los dientes de dos piezas, unos en otros.

ESLABÓN: Pieza con forma de anillo, que se enlaza con otras formando una cadena.

1.3.2 ROBOTS MÓVILES

De forma práctica para este trabajo, se pueden definir también como móviles:

Robots móviles son aquellos robots que poseen ruedas, patas u orugas, para su desplazamiento.

Como gran punto de referencia para esta categoría de Robots, se pueden citar los *rovers* o vehículos robóticos de exploración de la NASA. Uno de los más reconocidos y recordados a nivel general es Sojourner, empleado en la misión Pathfinder en Marte.

Este fue el primer vehículo rodado que se empleó en los planetas del sistema solar. De una planificación original de siete días, Sojourner estuvo 83 días explorando el terreno de marte, sacando fotografías, tomando muestras químicas y atmosféricas, entre otros tipos.

Estos son algunos datos descriptivos de este *rover*:

- Tipo: *lander/rover*.
- Estado: pasado.

- Fecha de lanzamiento: 04/12/1996.
- Lugar de lanzamiento: Cabo Cañaveral, Florida.
- Fecha de aterrizaje: 04/07/1997.
- Fecha de finalización de la misión: 27/09/1997.
- Objetivo: Marte.

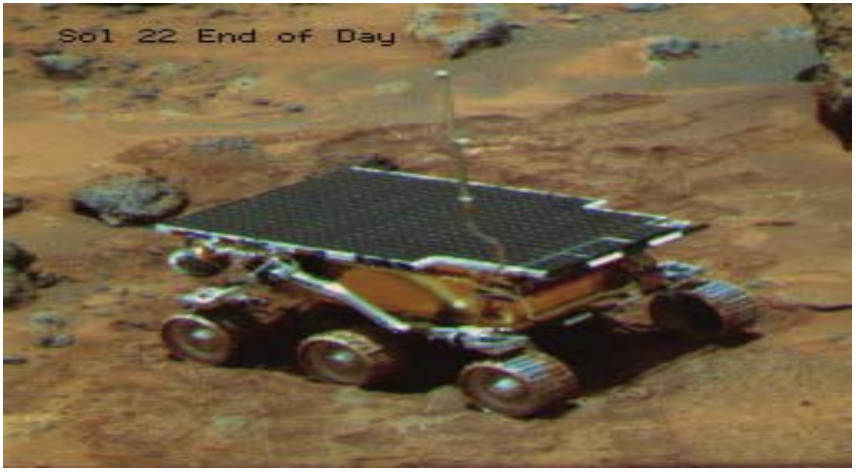


Figura 1.1. *Rover* Sojourner. Cortesía NASA.

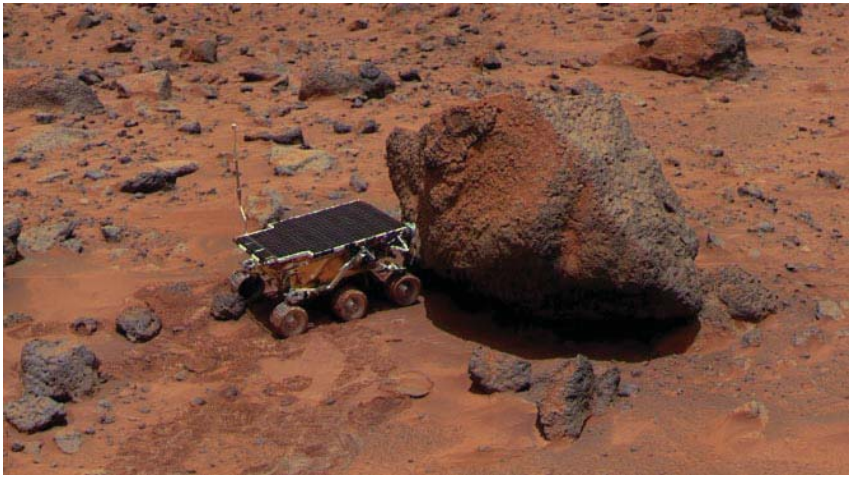


Figura 1.2. *Rover* Sojourner en exploración. Cortesía NASA.

Los robots móviles toman su denominación histórica, tanto para diferenciarlos por su principio de tracción básica, como así también por su contraste con los robots industriales, que están fijos.

ORUGA

Una oruga está formada por x cantidad de eslabones, generalmente en forma de anillos, que establecen una cadena.

Los materiales estándar empleados para la fabricación de orugas son el caucho, el acero, y las aleaciones de aluminio. Recientemente, y dado el crecimiento en el empleo de las impresoras 3D, es común el diseño e impresión de orugas empleando filamentos de material termo plástico, como se verá en el Capítulo 11.

Los sistemas de oruga son empleados en terrenos irregulares para dar estabilidad en el desplazamiento. Este sistema permite distribuir el peso en una superficie mayor y, consecuentemente, reducir la presión.

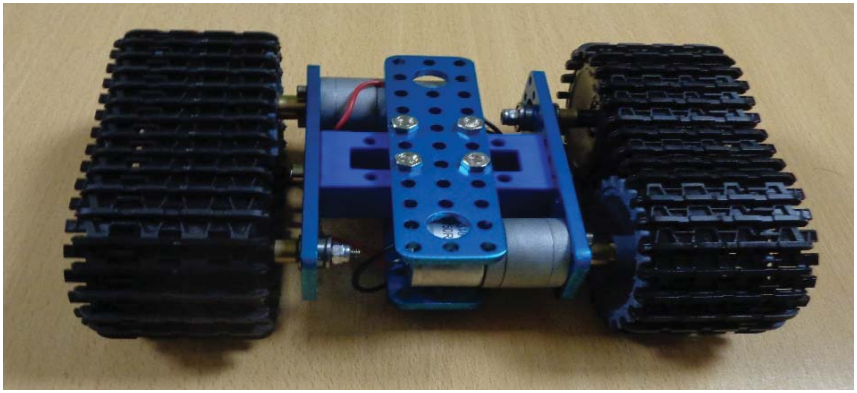


Figura 1.3. Orugas en base de robot mini tanque.

PRESIÓN

Se expuso antes que, al aumentar la superficie, disminuye la presión, y ello responde a que físicamente:

$$P\ (presión) = F\ (fuerza)\ /\ S\ (superficie)$$

En las siguientes figuras, se observan robots móviles seguidores de línea y luz, sin lógica adicional. Ambos tipos emplean sistema de montaje con ruedas.

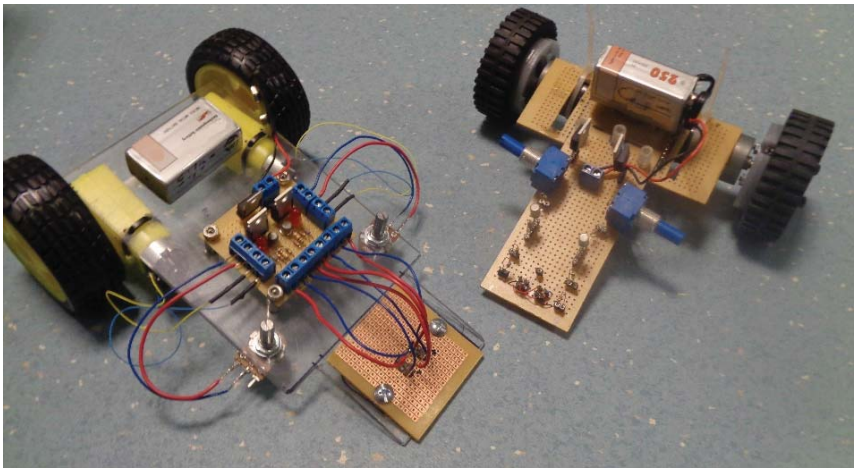


Figura 1.4. Robots móviles seguidores de línea.

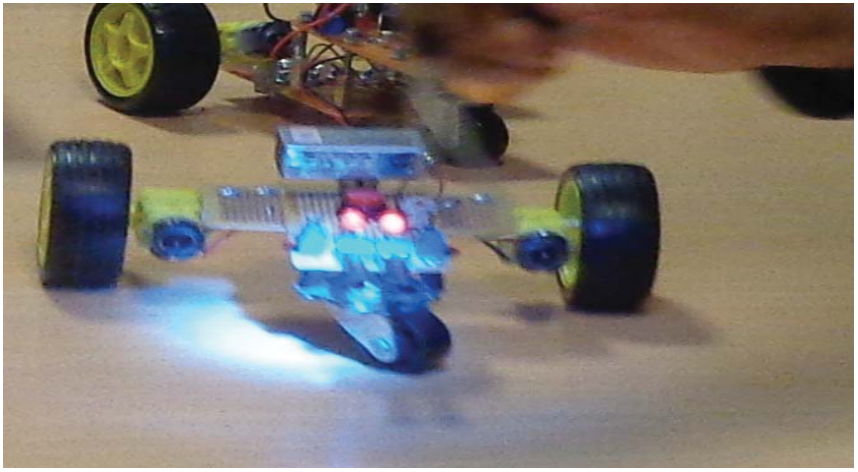


Figura 1.5. Robot Móvil. Seguidor de Luz.

1.3.2.1 TIPOS DE ESTRUCTURA MECÁNICA DEL ROBOT MÓVIL

ROBOT CON RUEDAS

Robot móvil que se desplaza utilizando ruedas.

ROBOT CON PATAS

Robot móvil que se desplaza utilizando una o más patas.

ROBOT BÍPEDO

Robot con patas que se desplaza usando dos patas.

ROBOT DE ORUGAS

Robot móvil que se desplaza sobre orugas/bandas de rodamiento.

ROBOT HUMANOIDE

Robot con cuerpo, cabeza y extremidades, mirando y moviéndose como un humano.

PLATAFORMA MÓVIL

Montaje de los componentes del robot móvil que permite la locomoción.

1.3.3 ROBOTS INDUSTRIALES

Este tipo o familia de robots fueron citados antes y, como su nombre lo indica, son el tipo de robots empleados en la actividad industrial.

Sin lugar a duda, desde su nacimiento en la década del cincuenta, su historia, evolución y aplicación es realmente fantástica, al punto de ser considerados pioneros dentro de esta disciplina.

La norma ISO 8373:2012, define robot industrial de la siguiente forma:

ROBOT INDUSTRIAL

Mecanismo actuador programable en dos o más ejes con un grado de autonomía, moviéndose dentro de su entorno, para realizar las tareas previstas.

- Un robot incluye el sistema de control y la interfaz del sistema de control.
- La clasificación del robot en Robot industrial o en el robot de servicio se realiza de acuerdo con la aplicación prevista.

Controlado automáticamente, reprogramable, multipropósito, manipulador, programable en tres o más ejes, que puede estar fijo en su lugar o móvil para uso en aplicaciones de automatización industrial.

ROBOT DE SERVICIO

Robot que realiza tareas útiles para los seres humanos o equipamientos, con exclusión de aplicaciones de automatización industrial.

Las aplicaciones de automatización industrial incluyen, pero no se limitan a, fabricación, inspección, empaque y ensamblaje.

Mientras que los robots articulados utilizados en líneas de producción son robots industriales, los robots articulados similares utilizados para servir comida son robots de servicio.

SISTEMA DE CONTROL

Conjunto de funciones lógicas de control y potencia, que permite monitorizar y controlar la estructura mecánica del robot y la comunicación con el entorno (equipos y usuarios).

EL ROBOT INDUSTRIAL INCLUYE

- El manipulador, incluyendo actuadores.

- El controlador, incluyendo la suspensión de aprendizaje y cualquier interfaz de comunicación (hardware y software).

Esto incluye otros ejes adicionales integrados.

MANIPULADOR

Máquina en la que el mecanismo consiste, normalmente, en una serie de segmentos, articulados o deslizantes uno con respecto al otro, con el fin de agarrar y/o mover objetos (piezas o herramientas), usualmente en varios grados de libertad.

Un manipulador puede ser controlado por un operador, un controlador electrónico programable, o cualquier sistema lógico (por ejemplo, dispositivo de cámara, cableado).

Un manipulador no incluye un efector final.

EFECTOR FINAL

Dispositivo diseñado específicamente para la fijación a la interfaz mecánica para permitir al robot realizar su tarea.

PINZA

Diseño del efector final para el agarre y la retención.

ACTUADOR DE LA MÁQUINA

Mecanismo de potencia utilizado para efectuar el movimiento del robot.

Ejemplo: un motor que convierte la energía eléctrica, hidráulica o neumática para efectuar el movimiento del robot.

BRAZO ROBÓTICO

Conjunto interconectado de eslabones y articulaciones motorizadas del manipulador, que comprende eslabones de forma longitudinal que posicionan la muñeca.

MUÑECA ROBÓTICA

Conjunto interconectado de eslabones y articulaciones motorizadas del manipulador entre el brazo y el efector final, que soporta, posiciona y orienta el efector final.

ESLABÓN

Cuerpo rígido que conecta las articulaciones vecinas.

Figura 1.6. Manipulador de la firma igus. Cortesía igus.

1.4 DRONES

De creciente empleo y demanda también en robótica, son los drones, empleados para seguridad, exploración, entretenimiento y producción.

Un Drone (Dron), VANT (Vehículo Aéreo No Tripulado o UAV (*Unmanned Aircraff Vehicle*) es una aeronave que vuela sin tripulación, y tiene la capacidad de mantener un nivel de vuelo controlado y sostenible.

Respecto a su empleo, cabe destacar que los drones son aeronaves, por lo que su uso está sometido a la legislación aeronáutica vigente para cada país.

1.4.1 CLASIFICACIÓN DE LOS DRONES

Los drones pueden ser clasificados teniendo en cuenta factores como:

- Su misión: reconocimiento, combate, logística, investigación y desarrollo, comerciales y civiles.
- Alcance: Handheld, close, nato, tactical, male, hale.
- Topología: *fix wings*, cuadricóptero, hexacóptero, octacóptero.

1.4.2 HISTORIA

Como proyecto, nace en la década del 40 acompañando al desarrollo de la industria aeronáutica. Sin embargo, no fue hasta fines del siglo XX cuando las primeras unidades fueron gobernadas por radiocontrol, ampliando significativamente la utilidad de estos pequeños aviones.

1.4.3 VENTAJAS Y DESVENTAJAS DE LOS DRONES

VENTAJAS

- Pueden ser empleados en áreas de alto riesgo o de difícil acceso.
- No requieren la actuación de pilotos.
- En la actualidad, son accesibles económicamente.

DESVENTAJAS

TÉCNICAS:

- Su funcionamiento es influenciado por fenómenos físicos.

AUTONOMÍA

- Litio: en promedio, la batería tiene 20 minutos de carga.
- Hidrógeno: se proyecta como material de empleo a futuro.

1.4.4 NUEVOS PARADIGMAS DE APLICACIÓN

COMERCIAL

- Agricultura de precisión.
- Transporte de carga.
- Construcción/inmobiliario.
- Drones recreativos/juguetes.

SALUD

- Drone ambulancia.

CIVIL

- Desastres naturales.
- Ayuda humanitaria.

CIENCIA

- Medioambiente.
- Aplicaciones climáticas.

OTROS

- Publicidad, eventos y producciones audiovisuales.
- Deportes.
- Recitales.

1.4.5. COMPONENTES DE UN DRONE

COMPONENTES DE UN DRONE
Fuselaje y estructura (motores, hélices).
Sensores: GPS, acelerómetro, giróscopo, brújula, ultrasonidos, barómetro, infrarrojo y proximidad.
Controladora Principal y Caja Negra.
Baterías.
Cámara de fotografía y video/cámara térmica.
Control remoto/Realidad Virtual.
Canal de comunicación.
Sistema de control de vuelo (Base de tierra + Operador.)

Figura 1.7. Componentes de un Drone.

Figura 1.8. Preparando la clase de Drones.

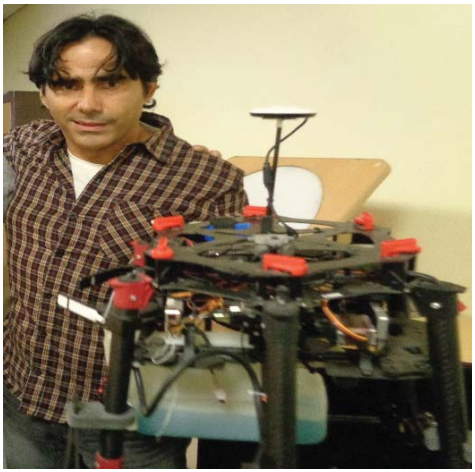


Figura 1.9. Yiye Rocha Diaz. Aplicaciones de Drones.

En la imagen se observa un drone fumigador.

1.5 ROBOTS A IMPLEMENTAR EN ESTA OBRA

Figura 1.10. Implementación de Robots.

Como se vio en las secciones anteriores, son muy variados en la actualidad los tipos de robots, las posibilidades tecnológicas, y las áreas de interés y aplicación.

En esta sección, se va a introducir la presentación y las características de los robots propuestos para el Diseño y Aplicación en este libro. Se expondrán también

de forma individual sus principios de funcionamiento, ya que luego se profundizará sobre ellos, cuando se trabaje en la implementación física.

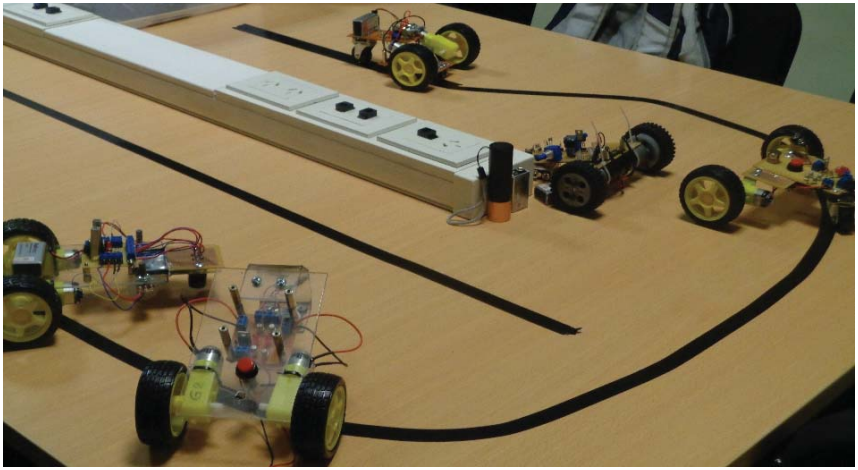


Figura 1.11. Implementación de Robots. Seguidores de Línea y Luz.

1.5.1 ROBOTS SIN LÓGICA ADICIONAL

La denominación **robots sin lógica adicional** responde a aquellos robots o dispositivos que no emplean lógica de CI, microcontrolador, FPGA, o Arduino para su funcionamiento. Por el contrario, emplean componentes discretos. Estos poseen tres etapas básicas para su funcionamiento: entrada-procesamiento-salida.

Donde:

- Entrada = sensores -> LDR o CNY 70.
- Procesamiento = Conmutación ante un evento. Transistor, relé.
- Salida = actuadores. Prender un motor, un LED testigo.

Discreto proviene del latín *discretus*, y significa separado. Hace referencia, a los componentes por separado (transistores, resistencias, diodos), o sin integración.

Como se verá más adelante, un microcontrolador es un circuito integrado digital programable, que posee x cantidad de pines o patas. Algunas de ellas pueden ser configuradas como entradas o salidas; de esta forma se obtendrían las tres etapas descritas anteriormente: entrada, procesamiento y salida.

Sin embargo, un efecto similar puede obtenerse con el empleo de transistores, componentes, que son la base tecnológica de los circuitos integrados.

Es posible aprovechar sus zonas de trabajo para producir una conmutación, que permita configurar el circuito, para que cada vez que se detecte un evento en la entrada, a través de los sensores. El transistor se comporte como si fuera una llave, *ON/OFF* (Prendido/Apagado o Abierto/Cerrado), y así, encender o apagar un motor. Los transistores, al igual que el resto de los componentes electrónicos, serán presentados en el Capítulo 3.

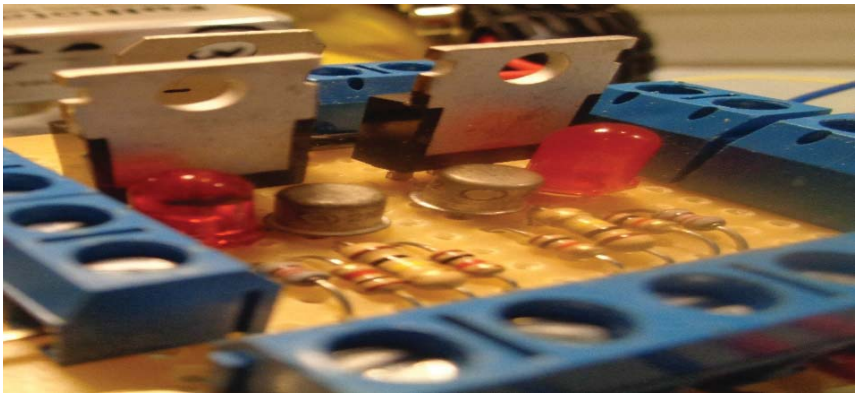


Figura 1.12. Placa de robot con componentes discretos.

Un componente empleado en todos los modelos de robots implementados en esta obra es el circuito integrado (CI) regulador de tensión positiva 7805.

Su empleo es fundamental, debido al tipo de motores utilizados, cuyo rango de tensión de trabajo es de 3 V a 6 V. Esto significa que, si los 9 V de tensión de la batería fueran aplicados de forma directa a los motores, estos se dañarían.

De esta forma, el CI regulador 7805 permite reducir la tensión de entrada, y entregar a la salida un valor de tensión de 5 V. Estos 5 V resultan también un valor seguro, para la especificación máxima, de la VBE (tensión base-emisor) de los transistores.

Su tipo de encapsulado (Figura 3.38) es el TO220:

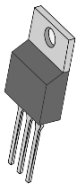


Figura 1.13. CI regulador 7805 de encapsulado TO220.

Visto de frente (*View Front*), la descripción de pines (*Pin Out*) de izquierda a derecha, es la siguiente:

PATA	DESCRIPCIÓN
1	Entrada (Input).
2	GND.
3	Salida (Output).

Figura 1.14. *Pin Out* del CI regulador 7805.

La corriente de trabajo máxima para este CI es de 1 A. Debe ser considerado, además, que la tensión de entrada no debe superar los 4 V respecto a la tensión de salida. Este margen (rango) entre las tensiones de entrada y salida, debe ser asegurado para que el CI pueda garantizar los 5 V de salida.

Para proteger el CI es empleado un disipador de calor y grasa siliconada.

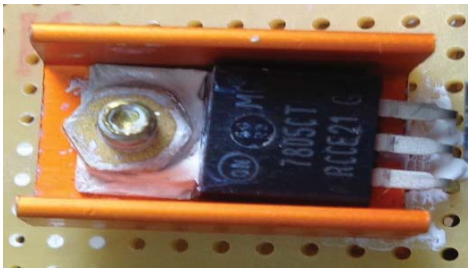


Figura 1.15. CI 7805 con disipador de calor y grasa siliconada.

1.5.2 ROBOT SEGUIDOR DE LUZ

Uno de los tipos de robots más interesantes para introducirse en el aprendizaje o trabajo en robótica, es el robot seguidor de luz. Esto se debe a su simpleza circuital, y la eficiencia de funcionamiento que posee.

Se verá a continuación que los circuitos eléctricos propuestos para los distintos tipos de robots son de desarrollo propio del autor: son diseñados para simplificar el montaje, reducir tamaño, y el coste final de los robots.

El robot seguidor de luz, al igual que el resto de los robots o dispositivos, emplea dos canales o lados, generalmente denominados izquierdo y derecho. Un detalle particular, es que los motores están cruzados.

Esto significa que visto de frente o *front view*, el motor que se vea a la izquierda corresponderá realmente al sensor que está del lado derecho. Esto permite que el robot gire o doble, para un lado u otro.

Figura 1.16. Robot seguidor de luz.

Cada uno de los canales emplea como sensor un LDR, o Resistencia Variable con la Luz, introducidos en un cilindro. Esto se debe a su forma original, y a modo de que no incida luz en ellos de forma indeseada. Como cilindros, se podrá emplear inicialmente termo contraíble (un material flexible), calculando el largo necesario desde el montaje del LDR en la placa.

Cuando se enfrenta el haz de luz de una linterna o móvil al robot, la luz incidente varía la resistencia del sensor (LDR), modificando el valor del divisor resistivo conectado a la entrada o base de un transistor.

Este efecto varía la entrada **base**, rompiendo la barrera de potencial (VBE) necesaria para conducir, y comportándose como una llave, cuya conmutación (cambio), deja a otra de sus patas, **emisor**, a un potencial de 0 V (-) o GND. El emisor queda unido internamente a su tercera pata, **colector**, que conecta en un nodo una resistencia llamada de *pull up*, con el polo negativo del motor. Así, permite que el motor gire, dado que su polo positivo (+), está fijo al de la batería.

1.5.3 ROBOT SEGUIDOR DE LÍNEA BLANCA

Si bien puede parecer que este tipo de robots posee simplemente aplicaciones de aprendizaje o hobby, su principio de funcionamiento es empleado también en la actividad industrial, donde son empleados para el traslado de materiales entre puestos de trabajo o depósitos.

Este robot emplea como sensor un CNY 70, que es un optoacoplador. Esto significa que, internamente, acopla por luz un diodo infrarrojo, con la base de un foto transistor.

Cuando el diodo infrarrojo (IR) emite luz, esta va a rebotar únicamente en presencia de una línea o superficie blanca, excitando, de esta forma, la base del foto transistor. Para este robot se emplean en la etapa siguiente, dos transistores

del tipo utilizado para el seguidor de luz, con el fin de formar una configuración llamada Darlington. En la cual, la ganancia de corriente de los transistores se multiplica.

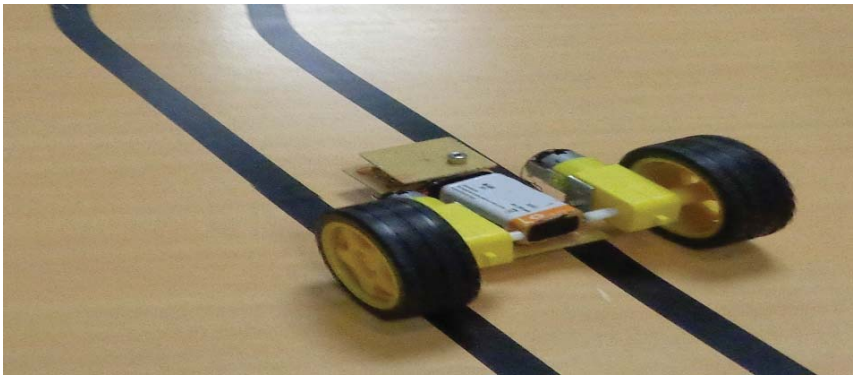


Figura 1.17. Robot seguidor de línea blanca.

1.5.4 ROBOT SEGUIDOR DE LÍNEA NEGRA CON LÓGICA DE CIRCUITOS INTEGRADOS DIGITALES NO PROGRAMABLES

El principio de funcionamiento de este robot es similar al anterior en la primera etapa (sensor CNY70). Luego, se diseña el mismo empleando dos circuitos integrados muy difundidos en robótica, requeridos también para el desarrollo de robot con microcontroladores.

Figura 1.18. Robot seguidor de línea negra.

Estos CI son el CD40106 Hex Schmitt Trigger, y el ULN2803 8ch Darlington Sink Driver el cual provee integrados 8 pares de transistores, del tipo empleados para el robot seguidor de línea blanca.

1.5.5 ROBOTS CON LÓGICA DE MICROCONTROLADOR

Será abordado en el Capítulo 9 el desarrollo de robots seguidores de luz, empleando microcontroladores programados en los lenguajes de programación C y Assembler.

Para la implementación de estos se parte de un modelo de robot ya conocido por el lector, (como el robot seguidor de luz) a fin de simplificar su introducción en las aplicaciones con Microcontroladores.

Para la programación de los microcontroladores será empleado el sistema ICSP (*in circuit serial programing* o programación serie en circuito), ver Capítulo 8. Este sistema permite descargar el archivo `.hex` al microcontrolador, sin necesidad de desconectarlo de la placa que controla.

1.5.5.1 ROBOT SEGUIDOR DE LUZ CON LÓGICA DE MICROCONTROLADOR EN LENGUAJE C

El principio de funcionamiento de este robot es similar al presentado en el Capítulo 5 para el robot seguidor de luz, A diferencia empleará un microcontrolador programable en lenguaje C, para la lectura proveniente de la etapa de entrada, compuesta por un divisor de tensión entre el LDR y una resistencia de 10 KΩ.

Cuando uno de los pines del microcontrolador detecte que su entrada está a nivel alto (seteada). Pondrá un 1 lógico a la salida de uno de sus pines, en el cual tendrá conectado un diodo LED. El mismo proceso será realizado para activar el motor.

Para permitir que el robot gire siguiendo la fuente de luz, no hará falta cruzar los motores respecto a la posición física de los sensores (cruzar los cables), ya que esta acción se realiza por software.

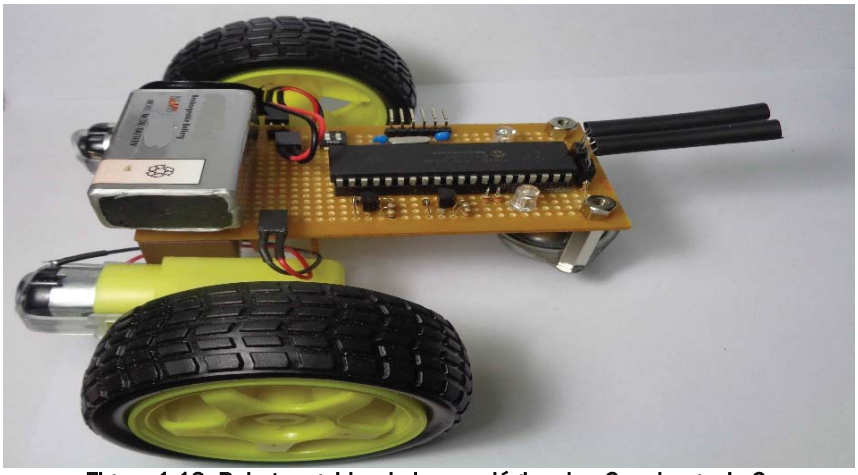


Figura 1.19. Robot seguidor de luz con lógica de µC en lenguaje C.

1.5.5.2 ROBOT SEGUIDOR DE LUZ CON LÓGICA DE MICROCONTROLADOR EN LENGUAJE ASSEMBLER

El principio de funcionamiento de este robot es similar al anterior con la diferencia de que es empleado un microcontrolador en lenguaje Ensamblador.

De esta forma, se introduce al empleo de los dos lenguajes de programación más empleados en las aplicaciones de microcontroladores y electrónica.

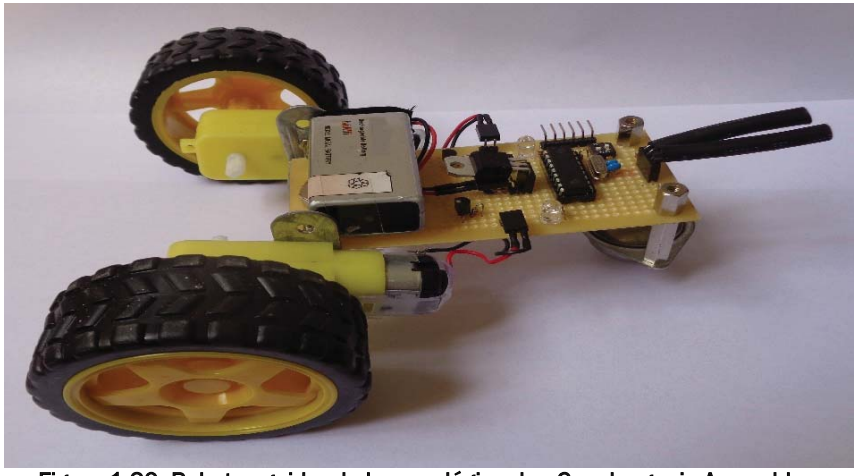


Figura 1.20. Robot seguidor de luz con lógica de µC en lenguaje Assembler.

CAPÍTULO 2
FUNDAMENTOS Y CIRCUITOS DE APLICACIÓN EN ROBÓTICA

2.1 CORRIENTE ELÉCTRICA

La corriente eléctrica es la circulación de cargas por el interior de un conductor.

Es una de las siete unidades básicas en el SI o Sistema Internacional de Unidades.

MAGNITUD	NOMBRE	SÍMBOLO
corriente eléctrica	ampere	A

Figura 2.1. Corriente eléctrica.

2.1.1 INTENSIDAD DE LA CORRIENTE ELÉCTRICA

La intensidad de corriente se representa por la letra I.

Es la cantidad de electricidad o carga eléctrica que recorre un conductor en la unidad de tiempo.

$$I=Q/t$$

I (intensidad en ampere)
Q (culombios)
t (segundos)

Figura 2.2. Intensidad de Corriente - Carga sobre tiempo.

De forma práctica se debe tener en claro que la corriente eléctrica es la que será demandada de acuerdo con las características circuitales de nuestros robots o dispositivos desarrollados. Entonces, como se verá más adelante, será condición necesaria contar con una batería que pueda entregar la cantidad Ah o Amper hora, requeridos fundamentalmente por los motores o actuadores.

Recuérdese también que la corriente eléctrica puede causar daño físico a las personas, por lo que se deben tomar en todo momento las precauciones necesarias; contar con las medidas de seguridad y protecciones establecidas en las reglamentaciones vigentes.

Por otra parte, si bien existen dos tipos de corriente eléctrica, **alterna**, y **continua**, se puede definir como guía práctica para el usuario, que la electrónica y sus dispositivos-salvo excepciones-trabajan o están diseñados para trabajar con corriente continua. No debe confundirse, por ejemplo, el hecho de que cuando se emplea una notebook, se conecta su fuente a la red eléctrica, que suministra corriente alterna. Esto se debe a que, a través de la fuente se produce el proceso de rectificación (conversión) de la corriente alterna a continua.

2.1.2 AMPERÍMETRO

El amperímetro es el instrumento empleado para medir la intensidad de la corriente eléctrica.

Tal cual se observa en las figuras 2.28 y 2.29 el amperímetro debe ser conectado en serie dentro del circuito, dado que como se verá, en un circuito serie, la corriente es la misma para todos sus componentes. Otra característica importante es que, para no afectar las características del circuito, los amperímetros son diseñados con una resistencia interna muy baja.

Pueden ser adquiridos como instrumento único, o emplear la función amperímetro del tester.

Los circuitos del capítulo han sido desarrollados con la versión demostración de Proteus. Potente software empleado para la simulación y el diseño de circuitos.

Figura 2.3. Proteus website.

El símbolo para el amperímetro es:

Figura 2.4. Símbolo eléctrico del amperímetro.

2.2 CORRIENTE ALTERNA (CA)

La corriente alterna es aquella en la cual tanto la magnitud como el sentido de la corriente y la tensión, varían a una determinada frecuencia.

La corriente alterna es el tipo de corriente empleada en la red eléctrica. Su prototipo de señal es la señal senoidal, tal cual se observa en la siguiente imagen:

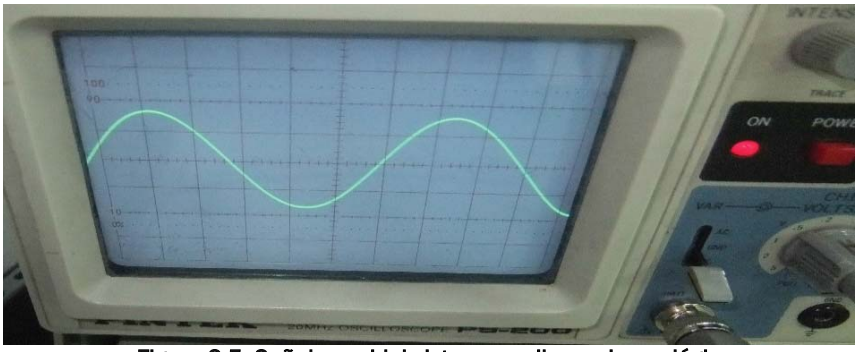


Figura 2.5. Señal senoidal vista en osciloscopio analógico.

Símbolo:

Figura 2.6. Señal senoidal. Símbolo.

Otras características de la señal senoidal son que representa una función continua, y posee infinitos valores.

2.2.1 FRECUENCIA

La frecuencia es el número de ciclos que se producen en un segundo.

Su unidad es el Hertz (Hz), y se representa con la letra f.

Un dato habitual, presente en los equipos de electrónica, es la frecuencia de la red eléctrica: para la Argentina es de 50 Hz, y para los EE.UU. 60 Hz. Se presentan también los rangos de tensión de trabajo y consumo de potencia.

Figura 2.7. Equipos electrónicos especificaciones.

2.2.2 PERÍODO

El período es el intervalo de tiempo entre repeticiones sucesivas de una forma de onda periódica.

Su unidad es el segundo (s), y se representa con la letra T.

A su vez, el período es inverso a la frecuencia. Esto significa que cuando el período aumenta, la frecuencia disminuye, y viceversa.

La ecuación característica es la siguiente:

$$T=1/f$$

2.2.3 CICLO

El ciclo es la forma de onda contenida en un período.

2.3 CORRIENTE CONTINUA (CC)

Como se describió, la corriente continua es el tipo de corriente que emplean los componentes y dispositivos electrónicos para su trabajo.

La corriente continua posee la característica de que los electrones circulan siempre (se mueven) en el mismo sentido con un valor constante.

Se encuentra disponible en dínamos, pilas y baterías. Pero, en general y dependiendo de la actividad, la corriente continua se obtiene de la corriente alterna, empleando un puente de diodos. Este proceso se describe como rectificación.

Símbolo:

Figura 2.8. Corriente continua símbolo.

2.4 TENSIÓN ELÉCTRICA

Se define la tensión eléctrica entre dos puntos de un conductor como el trabajo necesario para desplazar la unidad de carga entre un punto y el otro. A esta tensión se la llama también diferencia de potencial (d.d.p) entre dichos puntos.

La unidad de tensión eléctrica o diferencia de potencial es el volt, y se la representa por la letra V.

En el SI o Sistema Internacional de Unidades, es una unidad derivada.

MAGNITUD	NOMBRE	SÍMBOLO
Diferencia de potencial eléctrico, Fuerza electromotriz.	Volt	V

Figura 2.9. Tensión eléctrica.

EXPRESADA EN TÉRMINO DE OTRA UNIDAD SI
W/A

Figura 2.10. Tensión eléctrica en función de unidades del SI.

2.4.1 VOLTÍMETRO

El voltímetro es el instrumento empleado para medir tensión.

A diferencia del amperímetro, el voltímetro debe ser conectado en paralelo, dado que como se verá, en un circuito paralelo, la tensión es la misma para todos sus componentes. Otra característica importante es que, para no afectar las características del circuito, los voltímetros son diseñados con una resistencia interna muy Alta.

Pueden ser también adquiridos como instrumento único, o emplear la función voltímetro del tester.

Símbolo del voltímetro:

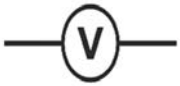


Figura 2.11. Símbolo eléctrico del voltímetro.

2.5 RESISTENCIA ELÉCTRICA

Figura 2.12. Resistencia. Símbolo eléctrico.

La resistencia eléctrica es la oposición que ofrece un cuerpo a la circulación de la corriente eléctrica.

Su unidad de medida es el ohm.

MAGNITUD	NOMBRE	SÍMBOLO
Resistencia eléctrica	Ohm	Ω

Figura 2.13. Resistencia eléctrica.

EXPRESADA EN TÉRMINO DE OTRA UNIDAD SI
V/A

Figura 2.14. Resistencia eléctrica en función de unidades del SI.

2.6 PRIMERA LEY DE OHM

A los fines prácticos, la ley de Ohm es tan simple como imprescindible, en el trabajo con circuitos de corriente continua. Se verá a continuación que la ley de Ohm vincula las tres unidades básicas desarrolladas anteriormente, y permite, conociendo el valor de dos de sus magnitudes, obtener el (valor) de la tercera.

La ley de Ohm establece que:

La corriente que circula por un conductor es directamente proporcional a la tensión V e inversamente proporcional a la resistencia R.

Expresado en fórmula:

$$I = V / R$$

Donde:

I = intensidad.
V = tensión.
R = resistencia.

Figura 2.15. Ley de Ohm: magnitudes.

2.7 TRIÁNGULO DE LA LEY DE OHM

El triángulo de Ohm se aplica con fines mnemotécnicos y como su nombre lo indica, integra las tres magnitudes comprendidas en la ley de Ohm.

Nuevamente, conociendo el valor de dos de las tres magnitudes, se tapa la magnitud incógnita y se obtiene como resultado la operación que se debe realizar (división o multiplicación) para obtener el resultado de esta.

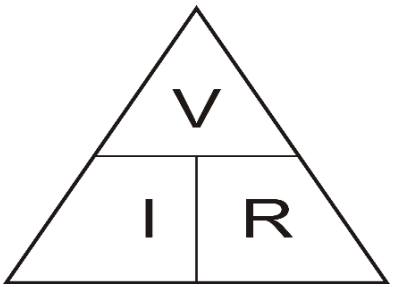


Figura 2.16. Triángulo de Ohm.

Por lo que la fórmula para el cálculo de un valor de tensión será:

$$V = I \cdot R$$

Y la fórmula para el cálculo del valor de una resistencia será:

$$R = V / I$$

2.8 CIRCUITO ELÉCTRICO

Un circuito eléctrico consta como mínimo de una malla, una fuente de tensión o de corriente, y al menos dos o más componentes eléctricos y/o electrónicos (diodo LED, interruptor, resistencia, capacitor, inductor, motor...), los cuales han sido dispuestos y calculados a fin de cumplir y/o desarrollar una o varias funciones, en o como parte de una aplicación.

2.9 LEYES DE KIRCHHOFF

Las leyes de Kirchhoff son dos y aplican al principio de conservación de la energía.

2.10 CONFIGURACIONES CIRCUITALES BÁSICAS

Conocer y dominar las diferentes configuraciones circuitales es fundamental en el trabajo en robótica, debido a que hay componentes y circuitos, que deben ser conectados empleando las tres configuraciones básicas:

Serie	Paralelo	Mixto

Figura 2.17. Configuraciones circuitales básicas.

2.11 CIRCUITOS SERIE

Características:

En un circuito serie, sus componentes comparten solamente un terminal, es decir, están seguidos uno de otro.

En un circuito serie la resistencia total es igual a la suma de todas las resistencias.

$$RT = R1 + R2 + ... + RN$$

En un circuito serie la corriente es la misma para todos sus elementos.

En un circuito serie, la caída de tensión en cada componente depende de su coeficiente resistivo.

Ejemplo - Fuentes de tensión en serie:

Se está construyendo un robot de sumo para competir, y entre las características de los motores elegidos, se encuentra que la tensión de trabajo es de 24 V.

Figura 2.18. Características de tensión del motor a emplear para el ejemplo.

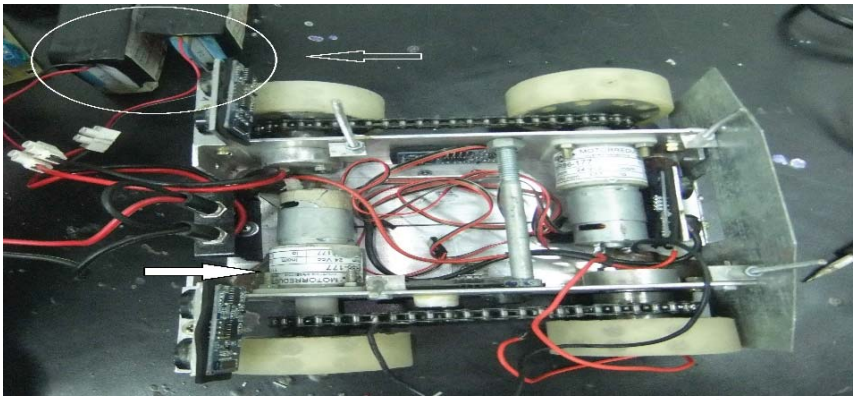


Figura 2.19. Montaje. Robot de sumo Wally IM. Motores de 24 VCC.

Para probar los motores en la mejor condición, se deben aplicar las características de los circuitos serie para agrupar las fuentes de tensión.

Figura 2.20. Batería recargable NI-MH 9 VCC.

Se aplica entonces:

En un circuito serie, sus componentes comparten solamente un terminal, es decir están seguidos uno de otro.

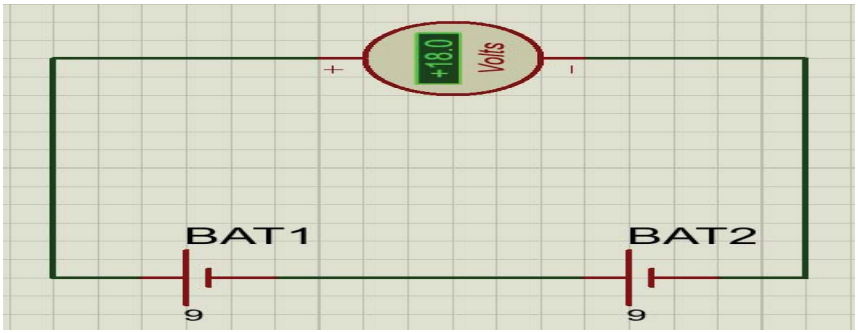


Figura 2.21. Símbolo eléctrico de la batería y conexión en serie.

Se observa en la imagen que, conectando dos fuentes de tensión en serie, los valores se suman, y se obtiene:

$$VT = V1 + V2 = 9\ V + 9\ V = 18\ V$$

Al conectar dos baterías en serie, de extremo a extremo, se obtiene el equivalente de una sola:

Figura 2.22. Las polaridades entre los extremos equivalen a una sola batería.

Para esta configuración, la IT (corriente total) suministrada por las fuentes es igual a 200mA/h. Es decir, duplica la tensión, pero mantiene el valor de corriente de una de ellas.

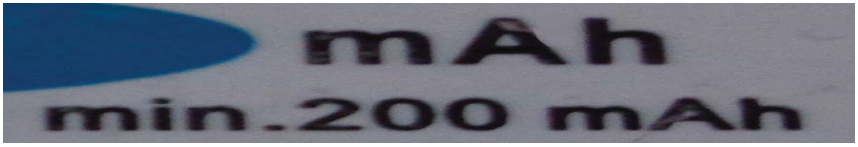


Figura 2.23. IT igual al valor de una de las baterías.

2.11.1 MALLA

En un circuito eléctrico, una malla es un camino cerrado para la corriente.

2.11.2 SEGUNDA LEY DE KIRCHHOFF (LVK)

La segunda ley de Kirchhoff, también llamada ley de las tensiones aplica a los circuitos serie y expresa lo siguiente:

En un circuito cerrado o malla, la sumatoria algebraica de las caídas de tensión es igual a la tensión aplicada.

Ejemplo:

Para el robot de la Figura 2.24, se emplean dos diodos LED rojos, para indicar el funcionamiento del canal izquierdo y derecho. Cada LED debe llevar una resistencia en serie, a fin de limitar la I que lo circula, y mantener su V de trabajo.

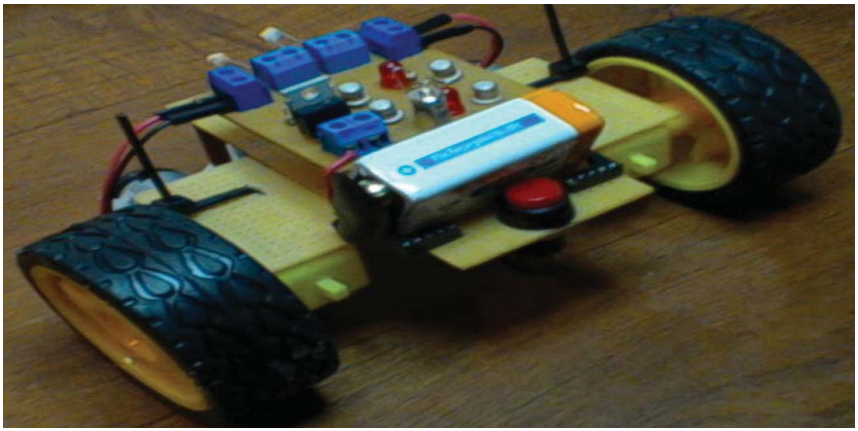


Figura 2.24. Robot seguidor de luz. Diodos LED en cada canal.

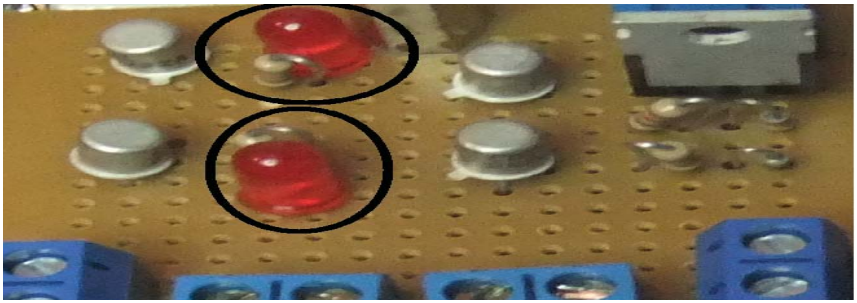


Figura 2.25. Resistencias en serie con los diodos LED.

Se empleará también este ejercicio para ejercitar con la Ley de Ohm (2.6 de este capítulo).

Como datos se tiene que para un diodo LED rojo, un valor seguro de corriente, es 25 mA, y su tensión de trabajo cercana a los 2 V. Otro dato para tener en cuenta es el valor de tensión de la fuente a emplear, 5 V para este proyecto.

La incógnita, es el valor que debe tener la resistencia en serie con el diodo **RD**.

$$RD = V_S - V_D / I_D$$

Donde:

RD	Resistencia serie del diodo LED
VS	Tensión de la fuente
VD	Tensión de trabajo del diodo LED
ID	Corriente de trabajo del diodo LED

Reemplazando, se obtiene:

$$RD = V_S - V_D / I_D = 5\ V - 2\ V / 25\ mA = 120$$

Circuito eléctrico

Figura 2.26. Resistencia en serie con el diodo LED.

Como se observa en la Figura 2.26, R1 y D1 están en serie, por lo que comparten un solo terminal.

Otra aplicación de agrupamiento de resistencias en serie, frecuente en robótica, ocurre cuando se necesita emplear un valor de resistencia, del que no se dispone.

Ejemplo:

Si para el circuito anterior, Figura 2.26, no se dispusiera al momento del montaje del robot, de una resistencia de 120 Ω, se debería recurrir a los valores comerciales más cercanos, de forma de agruparlos en serie, paralelo o mixto. Se aplicará ahora la agrupación serie.

Figura 2.27. Resistencia en serie para obtener valor cercano a los 120 Ω.

Obsérvese de la Figura 2.27 que se aplica la fórmula:

$$RT = R1 + R2 + ... + RN$$

Obteniendo:

$$100\,\Omega + 18\,\Omega = 118\,\Omega$$

Luego, como se verá en el apartado 3.3, las resistencias de carbón (las más empleadas), poseen una variación o tolerancia en su valor.

2.12 CIRCUITOS PARALELO

Características:

En un circuito paralelo, sus componentes tienen dos terminales o puntos en común.

En un circuito paralelo, la resistencia total es menor a la menor de todas las resistencias.

$$RT = 1 / 1 /R1 + 1/ R2 + ... + 1 / RN$$

En un circuito paralelo, la tensión es la misma para todos sus elementos.

En un circuito paralelo, la corriente en cada componente o rama depende de su valor resistivo.

2.12.1 PRIMERA LEY DE KIRCHHOFF (LIK)

La primera ley establece lo siguiente:

La suma algebraica de las corrientes entrantes en un nodo, es igual a la suma algebraica de las corrientes salientes del mismo.

La primera ley de Kirchhoff, también llamada de las corrientes, aplica a los circuitos paralelos.

Ejemplo:

Para un diseño propio de robot, se emplean en paralelo dos motores con las mismas características eléctricas, que demandan una corriente de entrada al nodo, de 420 mA, como se observa en el amperímetro de entrada. Luego, cada motor consume una corriente de 210 mA, cuya suma algebraica a la salida es igual al valor de entrada al nodo 420 mA.

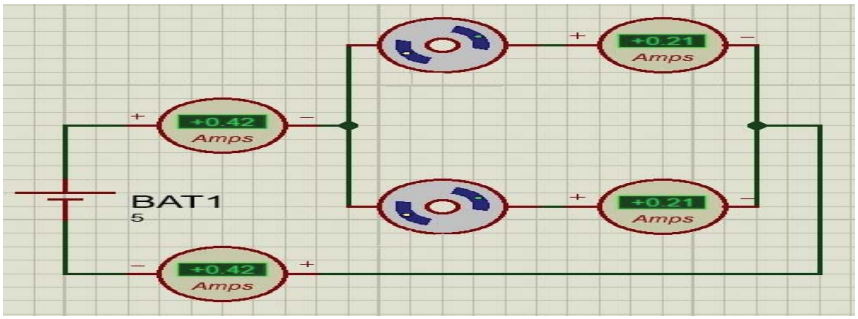


Figura 2.28. Ejemplo de aplicación de la 1ª Ley de Kirchhoff.

En formato de ecuación:

$$\sum I\,entrada = \sum I\,salida$$

$$0.42\,A = (0.21\,A + 0.21\,A)$$

2.12.2 NODO

En electrotecnia y circuitos de electrónica, un nodo es un punto de conexión entre dos o más componentes o ramas de un circuito.

Ejemplo:

Para el circuito anterior, de la Figura 2.28 se indican los nodos 1 y 2.

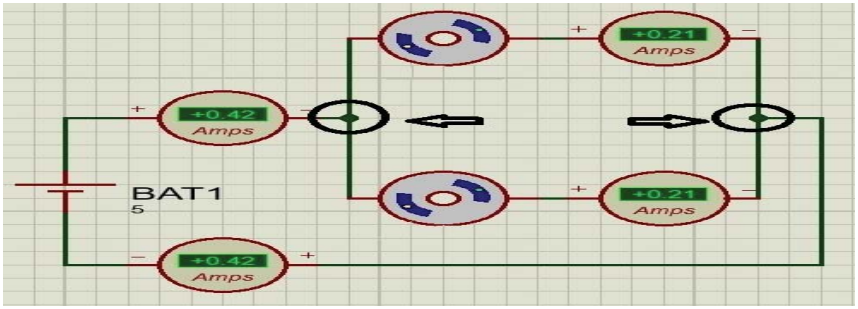


Figura 2.29. Demarcación de nodos. Ejemplo de la primera ley de Kirchhoff.

2.13 CIRCUITOS MIXTOS

Los circuitos mixtos son aquellos que combinan las características eléctricas de los circuitos serie y paralelo.

Los circuitos mixtos tienen su razón de ser, en la necesidad de obtener, por ejemplo, determinados valores de resistencias no comunes comercialmente, o disponibles al momento de necesitarlas para su empleo.

Ejemplo:

Se supone que se necesita emplear una resistencia de 5100 Ω, para polarizar un sensor de robot. Se cuenta con una resistencia de 100 Ω y dos de 10000 Ω o 10 KΩ. De esta forma, se puede emplear una configuración mixta para obtener dicho valor, aplicando:

Se agrupan las dos resistencias de 10 KΩ en paralelo, obteniendo un valor de 5000 Ω o 5 KΩ. Luego, se agrupa esta en serie con la resistencia de 100 Ω, obteniendo una RT = 5100 Ω.

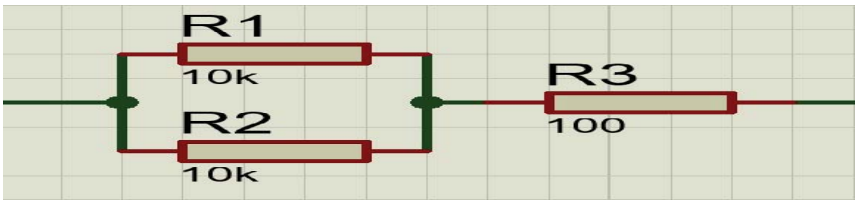


Figura 2.30. Agrupación de resistencias en configuración mixta.

Fórmulas:

$$RT\Omega = 1/1/10K + 1/10K = 5K\Omega$$

Esta resistencia equivalente se agrupa con la resistencia de 100 Ω, obteniendo una RT = 5100 Ω.

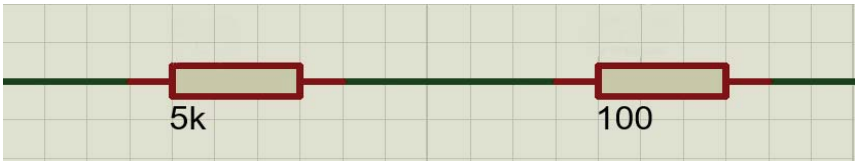


Figura 2.31. Circuito mixto. Obtención de una resistencia equivalente a 5100 Ω.

$$RT = R1_2 + R3 = 5000\,\Omega + 100\,\Omega = 5100\,\Omega$$

2.14 POTENCIA ELÉCTRICA

Definición:

Potencia es el trabajo realizado por unidad de tiempo.

Su unidad de medida es el watt.

MAGNITUD	NOMBRE	SÍMBOLO
Potencia	Watt	W

Figura 2.32. Potencia eléctrica.

EXPRESADA EN TÉRMINO DE OTRA UNIDAD SI
J/s

Figura 2.33. Potencia eléctrica en función de unidades del SI.

En corriente continua, la potencia eléctrica es igual al producto de la tensión por la intensidad de corriente.

$$P\ (potencia) = V\ (tensión).\ I\ (intensidad)$$

$$P = V\ .\ I$$

Si se reemplaza y se simplifica sobre las unidades anteriores de V e I, se obtienen también las siguientes fórmulas para el cálculo de potencia en corriente continua:

$$P = I^2\ .\ R$$

y

$$P = V^2 / R$$

Ejemplo:

Un motor cuya tensión de trabajo es 6 V, demanda una corriente de 125 mA. ¿Cuál es la potencia de dicho motor?

Resolución:

Se puede aplicar de forma directa, $P = V.\ I$ obteniendo:

$$P = V.I = 6\ V\ .\ 125\ mA = 750\ mW$$

2.15 TRIÁNGULO DE POTENCIA

El triángulo mnemotécnico de potencia permite también conocer de forma simple una de tres magnitudes desconocidas, partiendo de la primera fórmula de potencia vista.

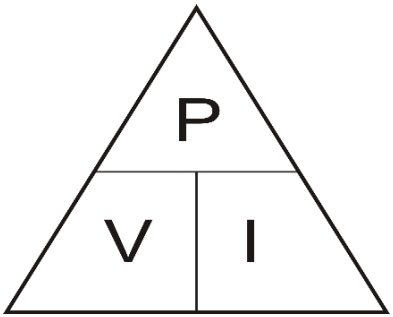


Figura 2.34. Triángulo de potencia.

2.16 GRÁFICO DE SECTORES

El gráfico de sectores integra las magnitudes con que se ha trabajado, sus diferentes fórmulas, y permite de forma directa y clara, hallar una incógnita en función de dos valores conocidos.

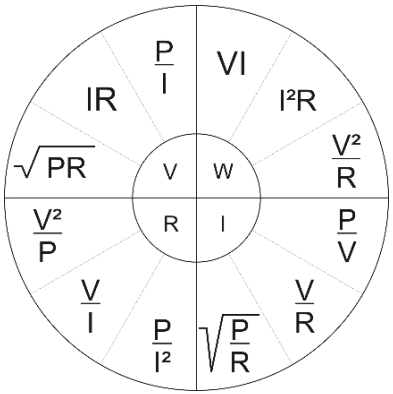


Figura 2.35. Gráfico de sectores.

2.17 SEÑALES

En robótica, los robots trabajan interactuando en todo momento con el medio en que se encuentran. En algunos casos desarrollan una actividad específica, tal como: seguir una línea negra o blanca; detectar la presencia de luz, y realizar un giro.

Con mayor integración de tecnología, encontrar la posición de otro robot en un tatami, bajar una pala mecánica e impactar.

También responder a un sensor de tacto; en el caso de los robots mascota; levantando una pata o emitiendo un sonido, más un sin fin de aplicaciones, donde lo que se hace en todo momento es procesar señales eléctricas. Estas pueden ser analógicas, digitales, o convertir de un tipo de señal a otro, empleando conversores DAC (*Digital to Analog Converter* o Conversor Digital a Analógico), ADC (*Analog to Digital Converter* o Conversor Analógico a Digital).

En este punto es importante destacar que, tal cual ocurre con las ramas de la Electrónica, hay dos tipos de señales: Analógicas y Digitales.

2.18 ELECTRÓNICA ANALÓGICA

La electrónica analógica es pionera en el campo de la electrónica. Siendo necesario señalar que no se debe confundir, dentro de la era digital, que los procesos físicos o naturales, representan señales continuas en el tiempo. Por el contrario, las señales digitales, emplean o representan valores discretos.

La electrónica analógica emplea magnitudes con valores continuos.

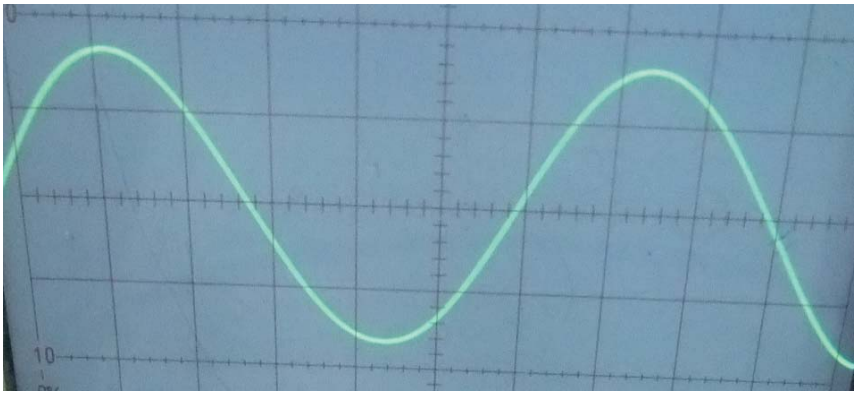


Figura 2.36. Señal senoidal. Función continua.

La señal senoidal es un ejemplo de función continua. En la figura se puede observar que no presenta ninguna discontinuidad. Sus valores son continuos e infinitos.

2.18.1 SEÑALES ANALÓGICAS

Las señales analógicas representan funciones matemáticas continuas, donde la amplitud es variable en función del tiempo, dependiendo al igual que la frecuencia de la información que contiene o su fuente de origen.

El prototipo de una señal analógica es la señal senoidal.

2.19 ELECTRÓNICA DIGITAL

En la actualidad, la electrónica digital lleva la vanguardia en la tecnología. Aplicaciones que nacieron siendo totalmente analógicas, como la televisión, han sufrido el conocido apagón analógico, con la consecuente migración de la TV analógica (emisiones analógicas) a la televisión digital.

La palabra digital deriva del latín y significa ***digitus***, es decir, **dedo**.

La denominación digital fue adaptada hoy en día. Digito es un número y aplica a los diez primeros elementos o base del sistema decimal: 0 al 9. Estos dígitos, justamente, se pueden contar con los dedos. De esta forma, el 0 y el 1, base del sistema de numeración binario, son también dígitos, de este modo, nace el sistema de numeración binario como base de los sistemas digitales.

2.19.1 SEÑALES DIGITALES

Una magnitud o señal digital es aquella que toma un conjunto de valores discretos.

El prototipo de una señal digital es la señal cuadrada.

En matemática, una función discreta es un conjunto numerable (o discreto). Esto contrapone con la función senoidal, prototipo de la señal analógica, que posee infinitos valores.

A la señal cuadrada, se la puede generar tanto en la transmisión de niveles lógicos con un microcontrolador, salida de una compuerta lógica, otros circuitos integrados digitales, o empleando al igual que con la señal senoidal, un generador de funciones.

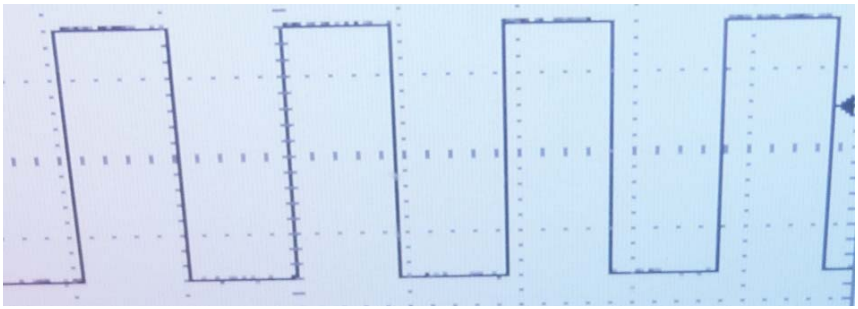


Figura 2.37. Señal cuadrada. Prototipo de Señal Digital.

2.20 SISTEMA DE NUMERACIÓN BINARIO

El sistema de numeración binario es un sistema de numeración cuya base es dos. Esto significa que posee dos elementos, 0 y 1, a los cuales se les suele llamar también dígitos o bits.

Etimológicamente, la palabra binar, del latín *binarius*, significa doble.

Por otra parte, y al igual que ocurre con el sistema de numeración decimal, el sistema binario es posicional. Se puede descomponer o trabajar con sus elementos o dígitos, en función de su posición o peso, empleando potencias de 2, dada su base. El primer peso, comenzando desde la derecha, corresponderá a la potencia dos elevado a la cero (2^0), cuyo valor es 1.

2^3	2^2	2^1	2^0
8	4	2	1

Figura 2.38. Primeras cuatro potencias positivas de 2.

Como se puede observar para los pesos o potencias tomados de ejemplo, las potencias positivas crecen de derecha a izquierda, duplicando su valor.

En el sistema binario, al igual que ocurre con cualquier sistema de numeración, se pueden obtener infinitos números.

Al dígito de menor peso (derecha), se lo denomina LSB o Bit Menos Significativo (*Least Significant Bit*). Al dígito de mayor peso (izquierda), se lo denomina MSB o Bit Más Significativo (*Most Significant Bit*).

Ejemplo: representar una tabla con el equivalente decimal - binario, de los números 0 al 4.

DECIMAL	BINARIO
0	0
1	1
2	10
3	11
4	100

Figura 2.39. Equivalente decimal – binario del 0 al 4.

2.20.1 CONVERSIÓN BINARIO A DECIMAL

Si para los valores binarios de la tabla anterior se debiera verificar la correspondencia con los valores decimales obtenidos, se aplicaría la siguiente definición:

El valor decimal de cualquier número binario puede hallarse sumando los pesos de todos los bits que son 1, y descartando los pesos de todos los bits que son 0.

De esta forma, y aplicando las tablas de las figuras 2.38 y 2.39, se toma como ejemplo el número decimal 4, correspondiente al binario 100.

BINARIO	DECIMAL
100	4

Figura 2.40. Ejemplo de equivalente binario - decimal del número 4.

Aplicando la definición dada, se reemplazan los bits que valen 1 por el valor correspondiente a su peso o potencia, y se suman. Se descartan los valores de los bits que valen 0. De esta forma, se obtiene el valor del número decimal equivalente.

Potencia de 2	2^2	2^1	2^0
Valor	4	2	1
Número binario	1	0	0
Valor Decimal	4 + 0 + 0 = 4		

Figura 2.41. Ejemplo de conversión binario a decimal.

2.20.2 CONVERSIÓN DECIMAL A BINARIO

El método empleado para convertir un número decimal a binario es el siguiente:

Se determina el conjunto de pesos binarios, cuya suma es igual al número decimal.

Ejemplo: Convertir a binario el decimal 4.

Resolución: se retoma nuevamente la tabla de la Figura 2.38

2^3	2^2	2^1	2^0
8	4	2	1

Donde, $(2^2) = 4$, por tanto, el equivalente binario es:

2^3	2^2	2^1	2^0
8	4	2	1
	1	0	0

Figura 2.42. Ejemplo de conversión decimal a binario del número 4.

Obsérvese también que, al estar a la izquierda no se representa el 0 de los pesos no empleados.

2.21. COMPLEMENTO A 1 DE UN NÚMERO BINARIO

El complemento a 1, tiene su implementación física en la compuerta inversora o NOT. Es un concepto altamente aplicado en robótica, dado el empleo de los inversores Schmitt Trigger (Capítulo 7), que permiten asegurar un nivel lógico 0 o 1.

El complemento a 1 de un número binario es su inverso.

La definición anterior significa que:

Dígito Binario	0	1
Complemento	1	0

Figura 2.43. Complemento a 1 de un número binario.

El complemento a 1 también se representa como C1.

Ejemplo: complementar a 1 la siguiente secuencia de números binarios.

01110011

Resolución:

Número Binario	0	1	1	1	0	0	1	1
Complemento a 1	1	0	0	0	1	1	0	0

Figura 2.44. Ejemplo de complemento a 1. Resolución.

2.22 NIVELES LÓGICOS Y NIVELES DE TENSIÓN

En la introducción a la electrónica digital, se presentó al sistema binario como base de los sistemas digitales, y se destacó su importancia en la transmisión de información (datos). Ahora se amplía este concepto, y se toma como ejemplo una

aplicación móvil, para que dispare vía bluetooth una señal asociada a la opción 3 de un menú, para producir un evento en un robot.

En esta aplicación, se debe tener en cuenta que el microcontrolador presente en el dispositivo, no sabrá jamás que existe el número 3 en dicho proceso, debido a que lo único que conoce o entiende es de 0s y 1s. Así todo dato debe ser convertido a este formato. Este proceso se logra debido a la correspondencia entre los niveles lógicos, y los niveles de tensión eléctrica, donde:

Nivel Lógico	0	1
Valor de tensión	0 V	5 V

Figura 2.45. Correspondencia entre niveles lógicos y de tensión.

Esto significa entonces que cuando se transmite o recibe un 0, el procesador o el sistema transmite o recibe un nivel bajo, *low state* o la ausencia de tensión. Y cuando se recibe o transmite un 1, se trabaja con un nivel alto, *high state* o 5 volt de corriente continua.

Nivel Lógico	0	1
Estado Lógico	Bajo o *Low State*	Alto o *High State*
Valor de tensión	0 V	5 V

Figura 2.46. Correspondencia entre niveles lógicos. Estados lógicos. Y valores de tensión.

2.23 RESISTENCIAS DE PULL DOWN

Conocidas las correspondencias entre los niveles lógicos y los niveles de tensión, como así también las características de tensión y corriente en los circuitos integrados, se deben trabajar dos circuitos o configuraciones circuitales, imprescindibles en la Robótica, *pull down* y *pull up*. Necesarias para asegurar dichos valores en las entradas de los CI, sensores, base de los transistores y nodos, entre otras aplicaciones.

La importancia de estas configuraciones o circuitos es que se trata simplemente de un supuesto afirmar que, a la entrada de un CI, o en un punto particular de un circuito, existe un nivel lógico bajo. Cuando no existe un nivel lógico alto, ya que, por inducción electromagnética o ruido, se podría tal cual se verá para los CI digitales (Capítulo 7), superar un valor de *thresold* o umbral, y poseer un nivel lógico diferente al pretendido.

Por otra parte, si se tuviera en un nodo el nivel bajo proveniente de una fuente de tensión, y se hiciera llegar a ese punto de intercesión eléctrica, un nivel lógico

alto, sin contar con una resistencia, se estaría poniendo en cortocircuito la fuente, pues sería igual a conectar directamente el + con el -.

Pull down significa tirar de abajo, y hace referencia a tirar o conectar el terminal opuesto de una resistencia conectada a un nodo, al nivel lógico bajo o su equivalente eléctrico de 0 V.

Ejemplos:

En la Figura 2.47 se presenta el caso de tener conectado en un punto de intersección o nodo del circuito, un nivel lógico bajo o 0 V, separado del estado alto (5 V) por una llave eléctrica.

En esta instancia, en la medida que no se produzca un cambio de estado (cierre de llave), el nivel de tensión en el nodo, visualizado en el tester, será de 0 V.

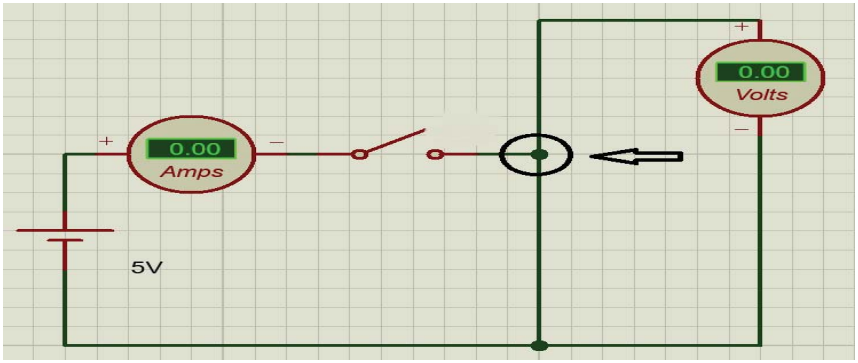


Figura 2.47. Ejemplo de circuito sin resistencia de *pull down*.

Pero en caso de ocurrir un cambio de estado (cierre de la llave), se puede analizar en el nodo que ahora el + y el – de la fuente, quedan unidos produciendo un corto circuito. Es decir que al no existir una resistencia que limite la circulación de electrones, la corriente tenderá a infinito, con el consiguiente daño de la fuente, y componentes.

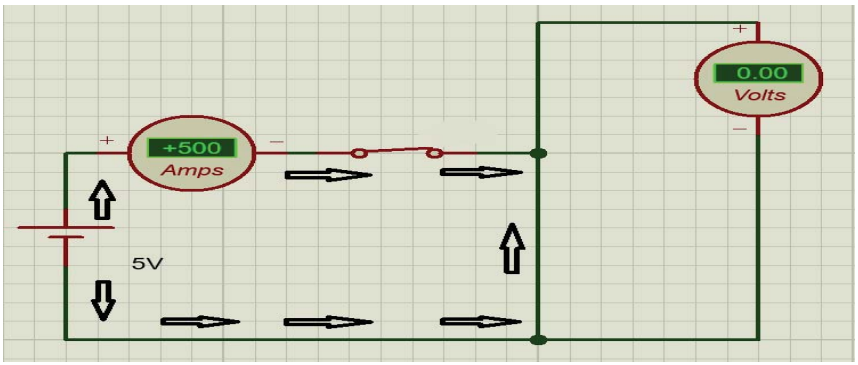


Figura 2.48. Al activar la llave (cambio de estado). Se produce un corto circuito.

A modo de asegurar los niveles lógicos ‘0’ y ‘1’, se deben emplear las resistencias de *pull down* o *pull up*, en función del tipo de dato, que se quiera leer.

En la Figura 2.48.se puede ver también que la llave está cerrada, sin embargo, el tester, sigue marcando 0 V. Pero la corriente demandada a la fuente es muy grande, debido a que se ha producido un cortocircuito. Obsérvese que ocurre ahora, cuando se conecta la resistencia de *pull down* (Figura 2.49).

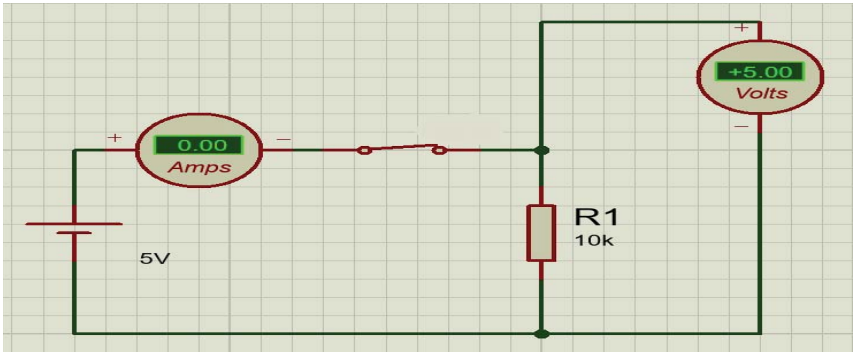


Figura 2.49. Resistencia de *pull down*.

Se puede apreciar que cuando la llave está cerrada, se tendrá un nivel lógico alto en el nodo, y si nuevamente se abre la llave, se obtendría un nivel lógico ‘0’. Se logra de esta forma asegurar los estados alto y bajo, y evitar cortocircuitos.

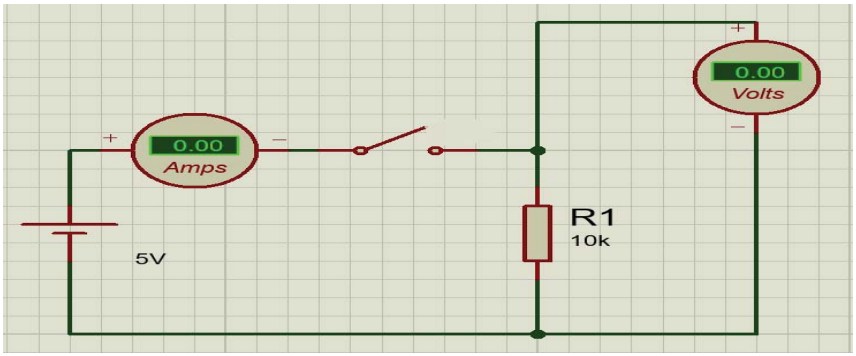


Figura 2.50. Cuando la llave está abierta se asegura un nivel lógico bajo.

La resistencia de *pull down* permite asegurar un nivel lógico bajo, en un punto determinado del circuito (entrada de un CI, nodo). Este valor se mantendrá hasta que ocurra un cambio o evento en el sistema (tecla pulsada, llave cerrada, conmutación de un sensor). Del mismo modo, se vio que su empleo es fundamental para evitar cortocircuitos y deterioros.

En la Figura 2.50. el valor de R1, la R de *pull down*, es de 10 KΩ. Este es un valor muy empleado en Robótica para ambas configuraciones. Puede darse el caso en algunos sensores, donde el fabricante dé la especificación del empleo de *pull down* muy fuertes, empleándose valores de 100 KΩ.

Recuérdese también y aplíquese lo trabajado en el presente capítulo, sobre Ley de Ohm, y configuraciones circuitales, donde para una tensión de 5 VCC, y una R de 10 KΩ, el valor de la I por esa rama será de:

$$I = V/R = 5\ V/10\ K\Omega = 0.5\ mA\ o\ 500\mu A$$

2.24 RESISTENCIAS DE PULL UP

Pull up significa tirar de arriba, y hace referencia a **tirar** o llevar el terminal opuesto de una resistencia conectada a un nodo, al nivel lógico alto, o su equivalente eléctrico de 5 V.

A diferencia de las R de *pull down*, los *pull up* son necesarios cuando se necesita de forma constante, tener o leer un nivel alto a la entrada de un CI o componente, y producir una acción o proceso, al ocurrir un cambio de estado, o lectura de un 0 lógico.

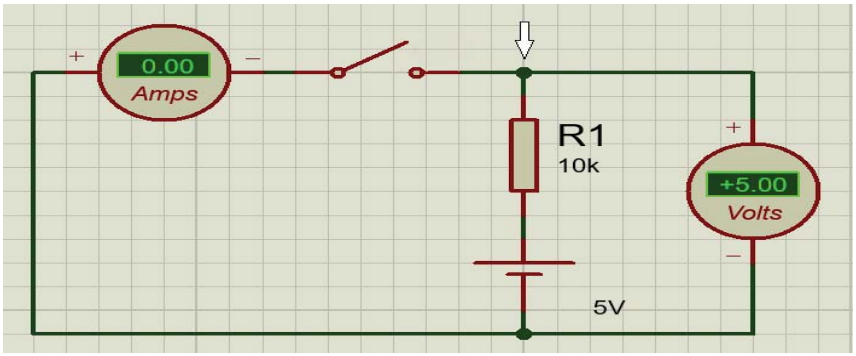


Figura 2.51. Resistencia de *pull up*. Lectura de nivel alto.

En la Figura 2.51 se observa que cuando la llave está abierta, se lee en el nodo un valor de tensión de 5 V, o nivel alto. Si se cierra la llave, (Figura 2.52), se produce un cambio de estado y lectura de 0 V.

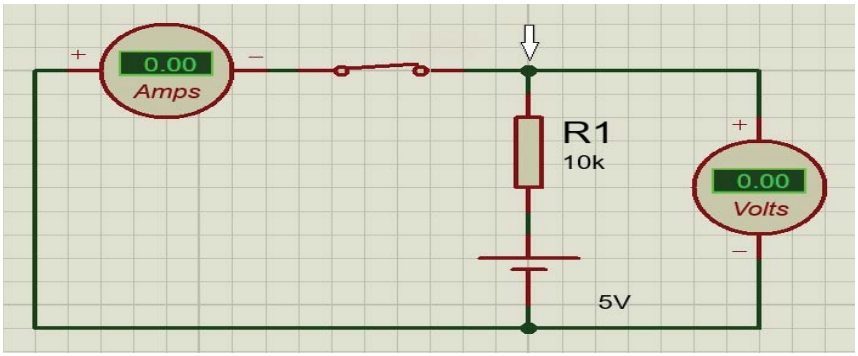


Figura 2.52. Cierre de llave. Conmutación y cambio de estado.

Los valores para las R de *pull up* son similares a los que se aplican en *pull down*.

Obsérvese en los ejemplos la indicación de 0 A en el amperímetro, salvo para el caso de cortocircuito. Esto es debido a que no se ha conectado una carga en el nodo correspondiente a la parte superior de la resistencia.

2.25 TIERRA O GROUND

Si bien puede no ser requerido por todos los simuladores de circuitos electrónicos, un símbolo fundamental para el desarrollo y teoría de circuitos es el de la tierra o *ground* (GND). Corresponde al terminal negativo de la batería o a un potencial eléctrico de 0 V.

Figura 2.53. Símbolo eléctrico. Tierra o *Ground*.

CAPÍTULO 3
COMPONENTES EMPLEADOS EN ROBÓTICA

3.1 INTRODUCCIÓN

Se expuso al inicio que uno de los grandes desafíos en la actualidad es hacer de la robótica una actividad de experimentación personal, a través de la implementación de robots. Para llevar a cabo esta propuesta, se trabajó en el Capítulo 2 en la introducción al conocimiento de fundamentos y configuraciones circuitales. Estos se necesitarán aplicar a los componentes electrónicos, a fin de desarrollar los robots o dispositivos.

3.2 TECNOLOGÍAS DE COMPONENTES THT Y SMD

Cuando en robótica y electrónica se hace referencia a tecnología de componentes THT (*Through Hole Technology*) Tecnología de Agujeros Pasantes, se habla del componente que posee patas o pines de conexión a la placa.

Una vez que las patas del componente han sido insertadas a través de la placa o PCB, deben ser soldadas siguiendo las pistas de cobre.

En el caso de las placas conocidas como: multipropósito, universales o experimentales, los componentes son soldados a las islas de cobre. Luego, cableados entre sí siguiendo el circuito eléctrico.

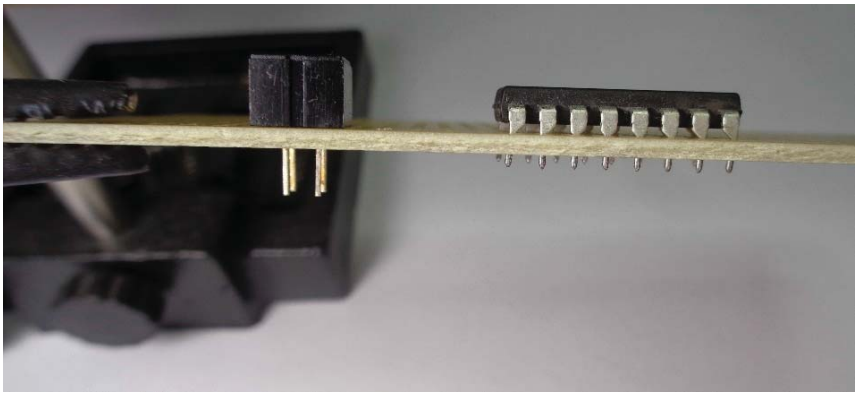


Figura 3.1. Sensor CNY70 y Driver L293B. Componentes THT de uso en robótica.

Si bien la tecnología THT sigue siendo muy empleada en electrónica, a nivel de diseño industrial se emplean dispositivos de tecnología SMT (*Surface Mount Technology*) Tecnología de Montaje Superficial. Es un sistema empleado para soldar estos componentes de tamaños reducidos, directamente a la superficie del circuito impreso o PCB. Se los denomina SMD (*Surface Mount Device*) – Dispositivo de Montaje Superficial.

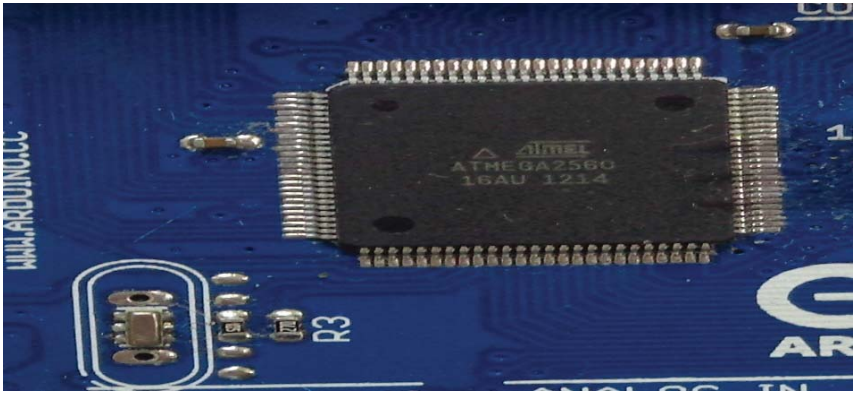


Figura 3.2. Microcontrolador y componentes SMD. Placa Arduino Mega 2560.

3.3 RESISTENCIAS

Se definió en el Capítulo 2 a la resistencia eléctrica como la capacidad de oponerse al paso de la corriente. Sin embargo, al conocer también y aplicar las leyes de Kirchhoff, tal el caso de la segunda ley o LVK, queda de manifiesto que las resistencias son fundamentales en el diseño y cálculo de circuitos, a fin de adecuar eléctricamente los mismos y proteger los componentes.

Como ejemplo de su aplicación en Robótica, basta simplemente citar que se puede armar el robot más simple, empleando dos resistencias de 10 KΩ, 1 LDR o resistencia variable con la luz, 1 transistor y un motor por canal.

Figura 3.3. Resistencia en SMD. Placa Arduino Mega 2560.

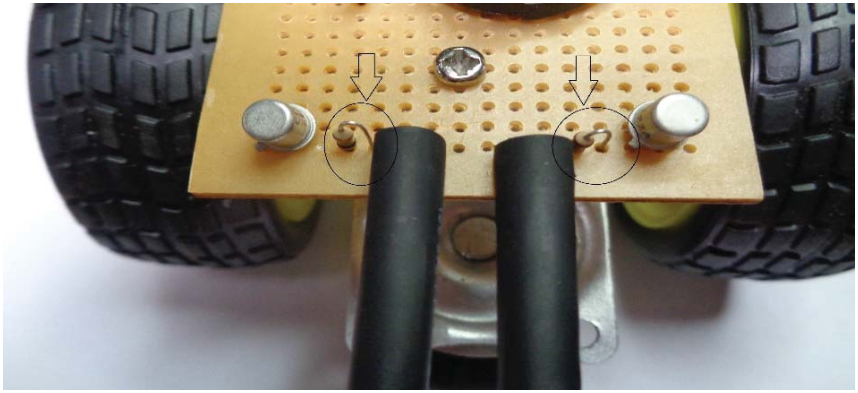


Figura 3.4 Resistencias de tecnología THT en Robot Seguidor de Luz.

3.3.1. TIPOS Y CARACTERÍSTICAS

La primera clasificación para las resistencias de aplicación en robótica, se la puede realizar dentro de:

TIPO	DENOMINACIÓN
FIJA	CARBÓN.
VARIABLES	POTENCIÓMETROS.
ESPECIALES	LDR.

Figura 3.5. Tipos de resistencias empleadas en Robótica.

Existen, además, diferentes tipos de tecnologías de materiales, e inclusive encapsulados para las resistencias. Las más empleadas en las aplicaciones de robótica son las de carbón.

Los LDR, si bien serán desarrolladas como sensores y empleados en el robot seguidor de luz, también son resistencias.

3.3.2. RESISTENCIAS FIJAS

Las resistencias fijas, como su denominación indica, poseen un valor fijo: se contraponen a las de tipo variable. Se las conoce también como lineales, dado su comportamiento.

3.3.2.1 RESISTENCIAS DE CARBÓN

Estas resistencias obtienen su denominación de la tecnología del material con el que están fabricadas.

Dentro de las características más importantes a tener en cuenta en el trabajo con resistencias, se encuentran:

Valor
Tolerancia
Potencia disipada

Figura 3.6. Resistencias fijas. Características a tener en cuenta.

VALOR

Las resistencias poseen sobre su cuerpo un código de colores, diseñado para identificar su valor (Figura 3.7)

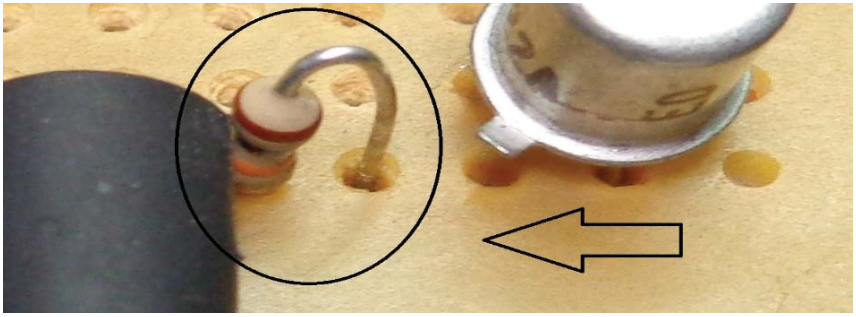


Figura 3.7. Código de colores para identificación del valor de la resistencia.

TOLERANCIA

Las resistencias de carbón no son precisas en sus valores. Entonces, la tolerancia indica el porcentaje de error existente en la lectura.

Una vez familiarizados con el código de colores, los valores se obtienen con un golpe de vista. La resistencia anterior, por ejemplo, posee un valor de 10 KΩ, con una tolerancia de ± 5% (puede valer entonces entre 9500 y 10500 Ω), compuesta por:

BANDA	COLOR	VALOR	VALOR FINAL
1ª	Marrón.	1	-
2ª	Negro.	0	-
3ª - ceros que agrega.	Naranja.	3	-
Tolerancia.	Dorado.	5%	10K Ω ± 5%

Figura 3.8. Cálculo del valor de la Resistencia de la Figura 3.7.

Los colores y valores tomados para el ejemplo surgen de la siguiente tabla.

Color	A 1ª cifra	B 2ª cifra	C 3ª cifra	TOLERANCIA
Negro	0	0	x1	
Marrón	1	1	x10	±1%
Rojo	2	2	x100	±2%
Naranja	3	3	x1000	
Amarillo	4	4	x10000	
Verde	5	5	x100000	
Azul	6	6	x1000000	
Violeta	7	7	x10000000	
Gris	8	8	x100000000	
Blanco	9	9	x1000000000	
Oro	-	-	x0,1	±5%
Plata	-	-	x0,01	±10%
Sin color	-	-		±20%

Figura 3.9. Resistencias. Código de colores.

POTENCIA DISIPADA

Se integra ahora la actividad trabajada sobre el cálculo de potencia en corriente continua -desarrollada en el Capítulo 2- para aplicarla al diseño de robots. Debe recordarse que, **la potencia está asociada a la capacidad de producción de trabajo**, es decir, del *power* que se desea que tengan sus dispositivos.

Se debe considerar en las implementaciones, la limitación física que la potencia presenta para la disipación de los componentes, como es el caso de la resistencia.

El mal cálculo del valor de la resistencia afecta también a otros componentes. Ejemplos pueden ser un diodo LED testigo, o la propia fuente de tensión del circuito.

Entre los valores de disipación de potencia más empleados para robótica y microelectrónica, se encuentran disponibles ½, ¼ de W. También hay valores comerciales superiores a 1 W, 2 W, y las líneas de resistencias de potencia, para aplicaciones de robótica industrial.

Debe tenerse presente que, asociado a cada componente, hay un profundo desarrollo en tecnología de materiales. Entonces, cualquier cambio en sus características, ya sea de potencia o corriente de trabajo, representa un cambio en su composición física. Las resistencias de potencia, por ejemplo, son fabricadas con aleaciones metálicas, dada su exigencia de empleo.

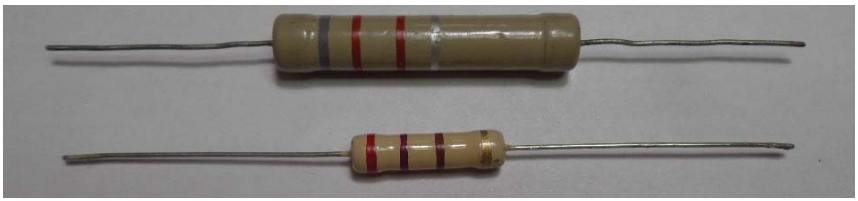


Figura 3.10. Resistencias de potencia. Valores superiores a 1 W de Metal Film.

Figura 3.11. Resistencia de potencia. 10 W de cerámica.

3.3.3. RESISTENCIAS VARIABLES

Las resistencias variables son aquellas que permiten variar el valor de su resistencia.

SÍMBOLO ELÉCTRICO

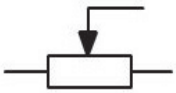


Figura 3.12. Símbolo eléctrico de un potenciómetro.

Los tipos de resistencias variables más empleados para las actividades de robótica son los potenciómetros y *presets*.

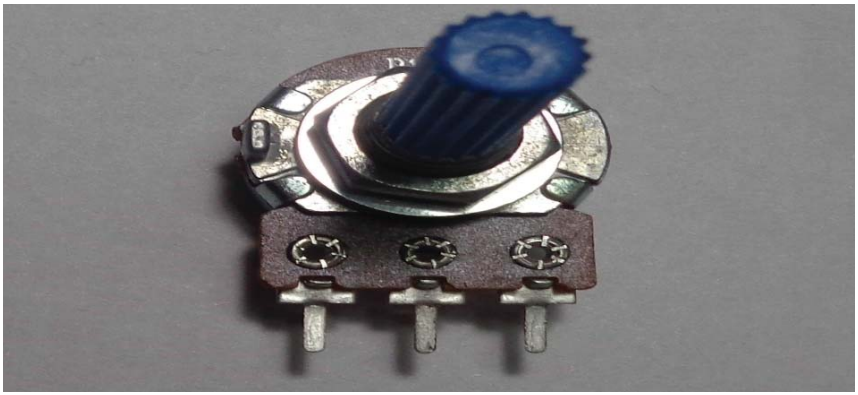


Figura 3.13. Potenciómetro de carbón. Lineal con eje plástico.

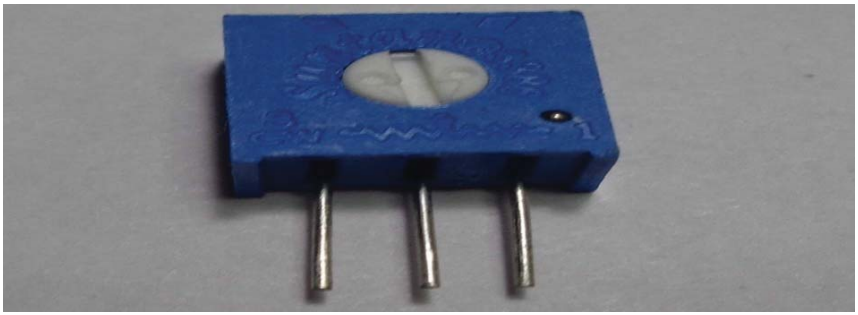


Figura 3.14. *Preset* Vertical de 1 vuelta. Tecnología Cermet.

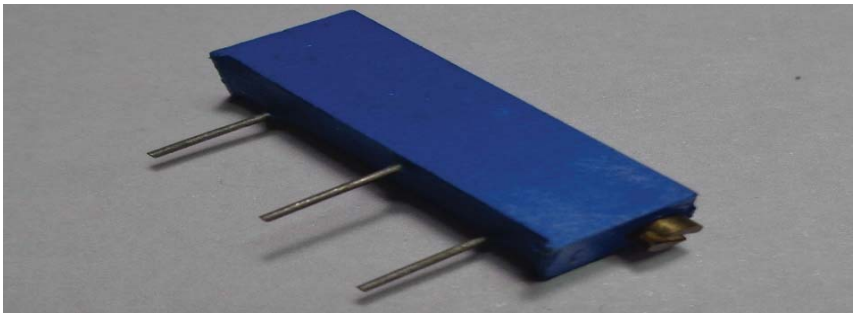


Figura 3.15. *Preset* horizontal multivuelta. Tecnología Cermet.

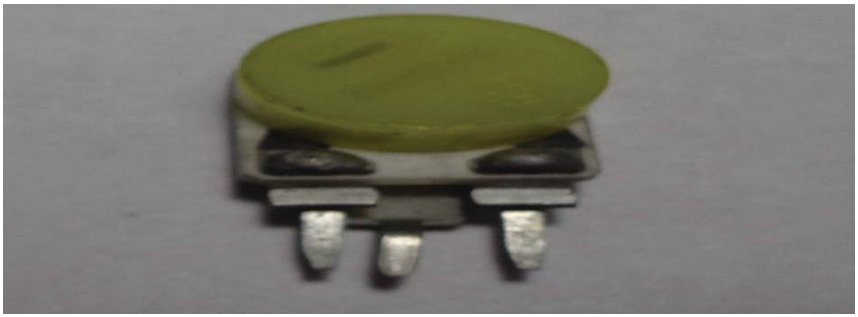


Figura 3.16. *Preset* vertical de carbón. Multivuelta.

3.3.4. RESISTENCIAS DE METAL FILM O PRECISIÓN

Las resistencias de precisión permiten reducir el porcentaje de tolerancia o aumentar la precisión del valor leído. Se puede obtener, dentro de valores estándar, una tolerancia del ± 1%.

Es interesante considerar que en estos diseños de robots habrá muchos factores que incidirán en la desviación de los valores ideales obtenidos en los cálculos. Por ejemplo: pérdidas por disipación y rendimiento en los motores, regulador de tensión, rozamiento y diseño de las ruedas. Entonces, es una buena práctica trabajar para mejor la precisión en cada etapa del circuito.

Figura 3.17. Resistencia de precisión de cinco bandas.

Para estas resistencias la tercera banda también representa un dígito.

La cuarta banda corresponde al multiplicador, con los siguientes valores: plata = 0.01, oro 0.1, negro = 1, marrón = 10 y los colores siguientes múltiplos de 10.

La quinta banda corresponde a la tolerancia, con los siguientes valores: plateado 10%, oro 5%, marrón 1%, rojo 2%, verde 0.5%, azul 0.25%, violeta 0.1%.

3.3.5. RESISTENCIAS DE ENCAPSULADOS ESPECIALES

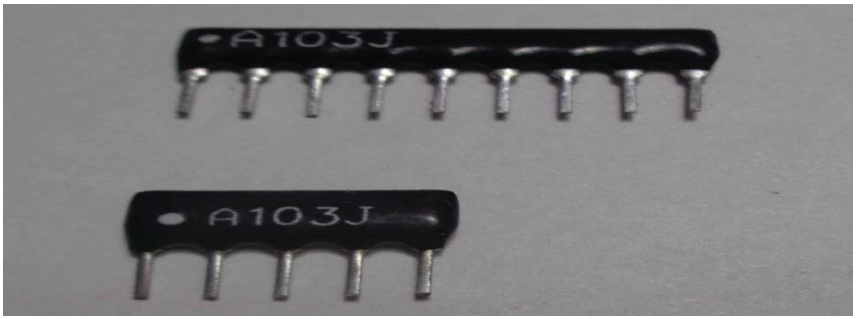


Figura 3.18. Resistencias en encapsulado Zip. Se emplean en *pull up* y *pull down*.

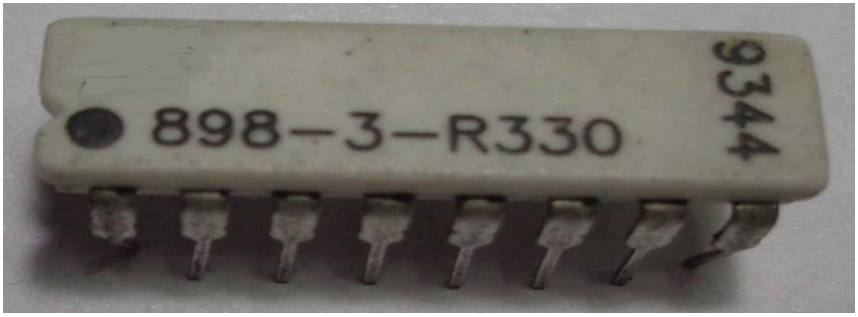


Figura 3.19. Resistencias en CI. Integra 8 resistencias de 330 Ω.

3.4 DIODOS

El diodo es un dispositivo semiconductor. Los semiconductores son elementos de valencia 4 (4 electrones en su órbita exterior). El semiconductor más empleado en la actualidad es el silicio, pero inicialmente se empleó también Germanio.

Por otra parte, un semiconductor intrínseco es un conductor puro, que posee mismo número de electrones libres que de huecos. A fin de aumentar la conductividad del semiconductor, se añaden átomos de impurezas. Este proceso se denomina dopado. Después de este proceso, el semiconductor se denomina extrínseco.

Es necesario dopar al semiconductor para que posea exceso de electrones y huecos. El objetivo es obtener semiconductores tipo p y tipo n. De la unión de un material tipo p con un material tipo n surge el diodo.

Figura 3.20. Material tipo P y material tipo N.

Figura 3.21. Conformación de un diodo.

Una de sus aplicaciones más comunes es la de rectificador. Este proceso consiste en rectificar(convertir) la corriente alterna en corriente continua.

Los diodos rectificadores se emplean de forma discreta en los paneles solares, que actúan como fuentes de tensión en aplicaciones de robótica. Esto se debe a que, en su principio de funcionamiento, permite la circulación de electrones en un solo sentido. De esta forma, bloquea el retorno de corriente desde el circuito hacia el panel a modo de protección.

Otra característica de los diodos es que en la unión de un material tipo P y el tipo N que lo conforman, se forma una zona llamada **zona de deplexión**, que actúa como barrera para el paso de electrones. Para romper esta barrera de potencial, se debe aplicar una tensión superior a los 0.7 V. Esta caída de potencial debe ser tenida en cuenta en los cálculos cada vez que se emplee un diodo rectificador en un circuito.

Figura 3.22. Zona de deplexión formada en la juntura de los materiales P y N.

El diodo es el principio de construcción de los transistores.

Figura 3.23. Diodo rectificador símbolo.

Figura 3.24. Diodo rectificador.

El diodo rectificador está compuesto por un terminal llamado Ánodo (+) y el Cátodo (-). Cátodo también se indica con la letra K y corresponde al terminal marcado con la línea plateada en la Imagen (raya en el símbolo eléctrico).

3.4.1 TIPOS Y CARACTERÍSTICAS

Además de diodos rectificadores, existen muchos otros tipos de diodos que poseen diferentes características. Han sido desarrollados para diferentes aplicaciones: comunicaciones, regulación, decoración.

A continuación, se presentan los diodos LED, RGB, y bicolores, empleados en aplicaciones de robótica.

3.4.2 DIODOS LED

LED es un acrónimo de Diodo Emisor de Luz. Los LED son empleados en los robots, como diodos testigo. Esto significa que indican cuándo un determinado punto del circuito está activo.

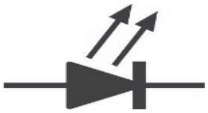


Figura 3.25. Símbolo diodo LED.

Cuando se trabaja con LED, se deben tener en cuenta sus características eléctricas, para no quemarlos.

Entre las especificaciones técnicas más importantes a tener en cuenta se encuentran su corriente de trabajo y la caída de tensión. Ella está asociada al color del diodo.

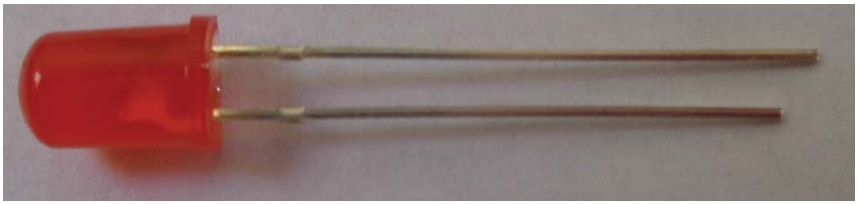


Figura 3.26. Diodo LED rojo difuso de 5 mm.

Existen, disponibles comercialmente, muchos modelos y tamaños de diodos LED. Dentro de los más empleados, destacan los diámetros de 3, 5 y 10 mm.

Para el diodo LED rojo de 5 mm, las especificaciones técnicas más importantes son:

TIPO	COLOR	IF (mA)	VF (V)	Intensidad (mCd)
Difuso.	Rojo.	15	2.1	2

Figura 3.27. Características eléctricas. Diodo LED difuso rojo de 5 mm.

IF = máxima corriente en directa.

VF = máximo valor de tensión en directa.

mCd = mili Candela.

También son muy comunes en la actualidad, los LED de alto brillo o luminosidad.

Figura 3.28. LED de alto brillo o luminosidad de 5 mm.

Para el diodo blue (azul) de la Figura 3.28, las especificaciones técnicas son:

IF = 20 mA.

VF = Min 3V – Typ 3,4 V.

Se encuentran también disponibles para este tipo de diodos, los colores: rojo, verde, amarillo, blanco,

DIODOS RGB

Otra opción interesante para trabajar en la terminación de proyectos de robótica son los diodos RGB, cuya denominación deriva de Red (rojo), Green (Verde), Blue (Azul).

Figura 3.29. Diodo RGB.

Para este modelo de diodo, la pata más larga corresponde al Cátodo (K) o negativo. De abajo hacia arriba: Red, Blue, Green.

DIODOS BICOLOR

Los diodos bicolor poseen un principio de funcionamiento similar al RGB. En robótica, son interesantes para visualizar un cambio de estado, nivel lógico o de tensión, en un robot.

Figura 3.30. Diodo bicolor RG.

En la Figura 3.30 se observa un diodo RG o Red (rojo) – Green (Verde).

3.5 TRANSISTORES

Para los modelos de robots propuestos a implementar en esta obra, se vio en el Capítulo 1 y será visto a partir del Capítulo 5, que el transistor cumple una función primordial.

Desde el punto de vista físico, estos semiconductores poseen tres terminales o patas. Pueden trabajar en modos de funcionamiento lineal y no lineal.

El modo no lineal se emplea en robótica para hacer funcionar al transistor como una llave eléctrica (conmutación). Cuando la VB (tensión de Base) supera un valor de 0.7 V, el transistor pasa del estado de corte al estado de saturación. Los equivalentes entre niveles lógicos y de tensión ya han sido trabajados y corresponden a 0 o 1 lógico, o 0 V y 5 V.

El transistor funciona en la zona activa cuando es utilizado como amplificador (circuito que amplifica señales débiles). A veces se lo denomina circuito lineal, porque los cambios en la señal de entrada producen cambios proporcionales en la señal de salida.

Los transistores son componentes activos, dada su capacidad de amplificar o ganar en corriente.

Existen dos grandes grupos de transistores: los bipolares o BJT (*Bipolar Junction Transistor*), y los FET o Transistores de Efecto de Campo (*Field Effect Transistor*).

3.5.1 BIPOLARES

Su construcción física básica fue descrita en la sección 3.4, al presentar los materiales semiconductores y diodos. Un transistor bipolar está compuesto por material tipo P y material tipo N.

La descripción de sus terminales, o patas, es la siguiente:

Terminal	Significado
B	Base.
E	Emisor.
C	Colector.

Figura 3.31. Descripción de terminales para un transistor bipolar.

Los transistores bipolares pueden ser de dos tipos: NPN o PNP

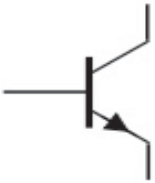


Figura 3.32. Símbolo de un transistor NPN.

Figura 3.33. Composición de un transistor NPN.

NPN corresponde a un material tipo Negativo-Positivo-Negativo. Físicamente, corresponde a una zona P (delgada) entre dos zonas N (más anchas).

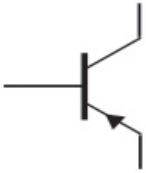


Figura 3.34. Símbolo de un transistor PNP.

Figura 3.35. Composición de un transistor PNP.

PNP corresponde a un material tipo Positivo-Negativo-Positivo.

3.5.2 MOSFET

Los transistores bipolares son la base de la electrónica lineal; su funcionamiento se basa en dos tipos de cargas: electrones y huecos. Sin embargo, debido a sus características, y para otro tipo de aplicaciones también, es más adecuado el uso del transistor unipolar, el cual depende de una sola carga. Un tipo de transistor de una sola carga es el FET (*Field Effect transistor*) que se traduce como: transistor de efecto de campo.

Dentro de los FET básicos, se encuentran:

- el FET de unión, o JFET.
- el MOSFET de empobrecimiento – FET de semiconductor óxido metal, también llamado IGFET (IG significa *insulated gate*, puerta aislada en inglés).
- El MOSFET de enriquecimiento o acumulación, empleado en la fabricación de circuitos integrados.

Complementary Metal-Oxide Semiconductor (Semiconductor de Metal Oxido Complementario) define que emplea dos tipos de transistores en el circuito de salida: uno de canal n, y otro de canal p tipo MOSFET (un transistor tipo p es complementario de un tipo n y viceversa). MOSFET significa *Metal Oxide Semiconductor Field Effect Transistor*, es decir, Transistor de Efecto de Campo Semiconductor de Metal Oxido.

Los CI CMOS pueden ser diferenciados mediante la denominación 4000, como será desarrollado en el ítem 7.4.3.

Como características principales de la tecnología CMOS, se pueden mencionar:

- Bajo consumo.
- Amplio margen de tensión de alimentación.
- Alta inmunidad al ruido.

3.5.3 FOTOTRANSISTORES

Los fototransistores serán empleados en la aplicación del CNY70 (Figura 5.22 Robot Seguidor de Línea Blanca). A diferencia de los transistores bipolares, o FET, poseen solamente dos terminales, equivalentes al emisor y colector, ya que su base es excitada por luz.

Conceptualmente, la radiación luminosa penetra en el interior de la unión base-colector; de esta forma, la corriente de colector es proporcional a la intensidad de la radiación.

Figura 3.36. Símbolo eléctrico de un fototransistor NPN.

Se pueden encontrar también fototransistores que poseen físicamente la pata de entrada a la base implementada. Un ejemplo es el CI 4N25, cuya imagen se muestra a continuación:

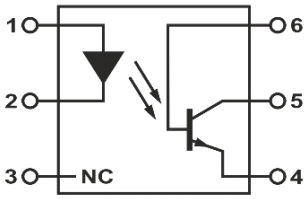


Figura 3.37. CI 4N25.

El CI 4N25 es un fototransistor optoacoplado de 6 pines.

3.5.4 ENCAPSULADOS MÁS EMPLEADOS EN ROBÓTICA

En la imagen siguiente se muestran los encapsulados más empleados en aplicaciones de Robótica.

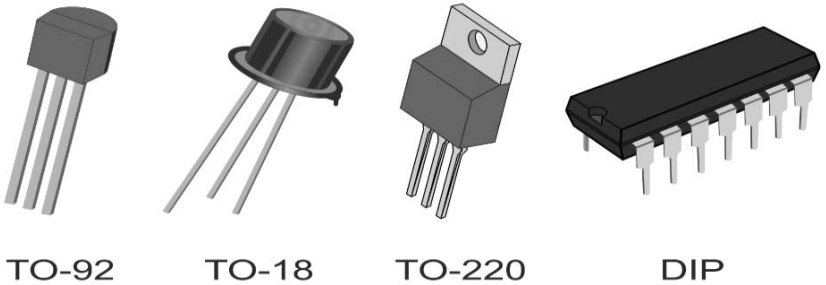


Figura 3.38. Encapsulados más empleados en robótica.

3.5.5 CONFIGURACIÓN DARLINGTON

La configuración Darlington posee como característica el empleo de dos transistores conectados en Cascada. Su función es la de multiplicar la corriente.

La ganancia final de corriente es igual al producto del hFE o βcc de ambos transistores.

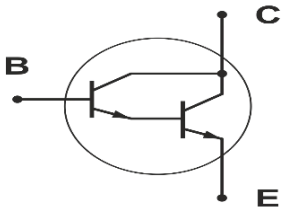


Figura 3.39. Transistor Darlington.

Las opciones para trabajar con la configuración Darlington son las siguientes:

- Implementar la configuración empleando dos transistores. Este es el método aplicado en el Capítulo 5 para la implementación del robot seguidor de línea blanca (Figura 5.22).
- Adquirir un transistor Darlington como el 2N6037. Ver el diagrama esquemático interno (Figura 3.40).
- Emplear el CI ULN2803. Este CI es empleado en el Capítulo 7 para la implementación del robot seguidor de línea negra con lógica de circuitos integrados digitales no programables (Figura 7.24).

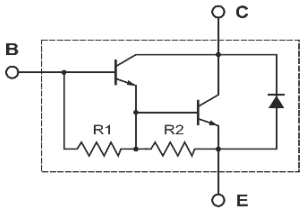


Figura 3.40. Transistor Darlington 2N6037. Diagrama Esquemático Interno.

Obsérvese en la imagen que el transistor posee integradas dos resistencias, formando un divisor de tensión.

Se debe tener en cuenta, para el diseño con estos transistores, que, al estar compuesto internamente por dos transistores bipolares, poseen doble caída de tensión. La razón radica en los 0,7 V necesarios para romper la barrera de potencial en el diodo base-emisor (Figura 3.22).

Puede ser analizado por la denominación de este transistor que los diodos poseen la nomenclatura 1N. Sin embargo, la tecnología de transistores bipolares posee la nomenclatura 2N, porque poseen dos diodos en su conformación básica.

3.5.6 MÉTODO DE TESTEO DE FUNCIONAMIENTO

En los robots a implementar en esta obra se emplean transistores bipolares, a excepción del empleo de Darlington en CI. Por tanto, se recomienda verificar el estado de los transistores antes de ser montados en el circuito de robot.

Un método, tanto simple como efectivo para los transistores bipolares, es pensarlos como si fueran dos diodos.

Para un transistor NPN se debe seleccionar el tester en la posición de diodo, y medir el diodo entre la base (+) y emisor (-). El mismo método se emplea para medir el diodo entre base (+) y colector (-).

Al medir continuidad entre emisor y colector, se debe indicar un valor de resistencia infinita (no continuidad) en ambas direcciones.

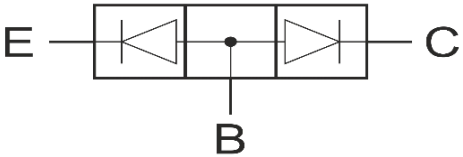


Figura 3.41. Transistor Bipolar. Diodos Base-Emisor y Base-Colector.

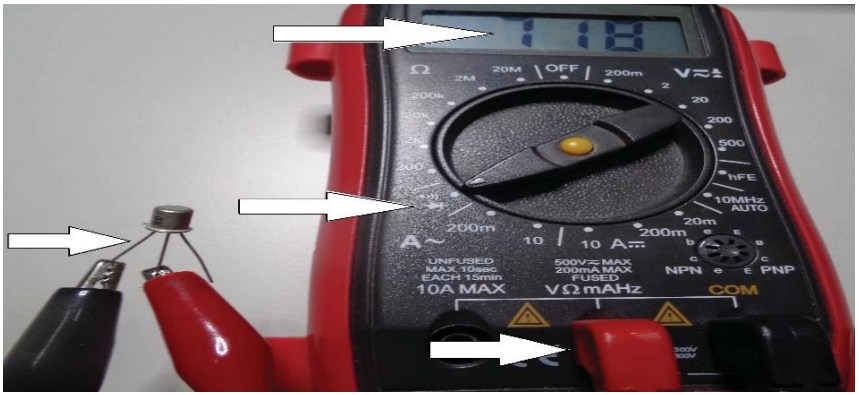


Figura 3.42. Transistor Bipolar. Prueba de funcionamiento.

Obsérvese en la imagen:

- Flecha superior en la imagen: el display muestra un valor de 718, equivalente a 718 mV o 0,7 V. Estos corresponden a la caída de tensión necesaria para romper la barrera de potencial entre el material tipo P (base) y tipo N (emisor para el ejemplo).

Si el transistor estuviera dañado, el display marcaría el valor '1', cuyo significado es el de resistencia tendiendo a infinito (por lo que no circula corriente). Lo mismo ocurriría si se invirtieran las puntas del tester + y común (-).

- Segunda flecha de arriba hacia abajo: se indica el seteo en el tester de la medición de diodos.

- Tercera flecha de arriba hacia abajo: se muestra la medición del diodo Base (+) Emisor (-). Luego se realizará la misma medición para el diodo Base (+) Colector (-).

- Cuarta flecha de arriba hacia abajo: se muestra la conexión de puntas o terminales. El cable negro corresponde al común (-). El cable rojo es conectado para la medida de magnitudes generales y representa al positivo (+).

3.5.7 PIN OUT DE LOS TRANSISTORES A EMPLEAR

Con el objeto de que sea una guía para la implementación de los robots de esta obra, se muestran a continuación, los diagramas de pines de los transistores 2N2222A y CTPN2222A.

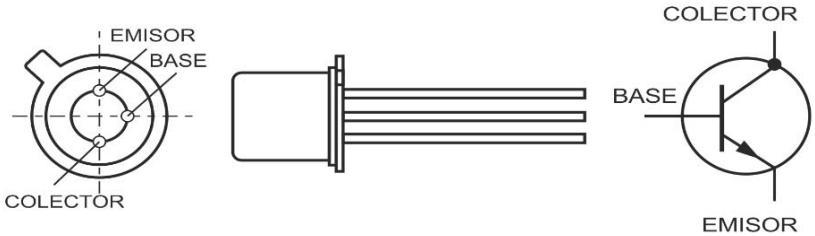


Figura 3.43. Diagrama de pines del transistor 2N2222A.

La forma más práctica y simple para la identificación de las patas del transistor 2N2222A consiste en cómo se visualiza en la imagen anterior: Identificar la chapa metálica que hace referencia al emisor, recordando que la pata central corresponde a la base y la tercera pata, al colector.

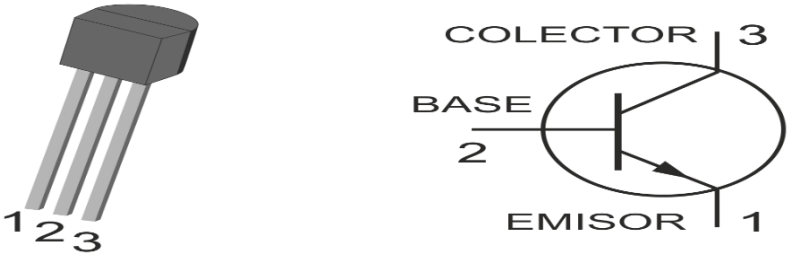


Figura 3.44. Diagrama de pines del transistor CTPN2222A.

En la imagen se observa la vista de frente o View Front del transistor CTPN2222A. Encapsulado TO-92.

3.5.8 OBTENCIÓN DEL VALOR DEL BCC O HFE

La ganancia de corriente de un transistor (las veces que gana en corriente), se denomina con la letra griega β. Sin embargo, en las hojas de datos o data sheets de los transistores, se trabaja con los parámetros conocidos como h (de híbrido), empleando la denominación de hFE. Los valores del βcc y del hFE son similares, por lo que se trabajará con este parámetro.

En el ítem 3.5.5 se vio que la ganancia de corriente para un transistor Darlington es igual al producto del hFE de sus transistores.

Las ecuaciones básicas para las corrientes en un transistor son las siguientes:

CORRIENTE	ECUACIÓN
Corriente de Colector (IC)	IC = βcc. IB
Corriente de Base (IB)	IB = IC/ βcc
Corriente de Emisor (IE)	IE = IB + IC

Figura 3.45. Corrientes de un transistor.

Al profundizar en esta actividad, se conoce que la IB, dependiendo del tipo de transistor, es muy pequeña, equivalente al 1% de la IC.

El emisor en cambio es la fuente de los electrones. Mayoritariamente, circulan hacia el colector. Entonces, en muchos casos, se aplica que:

$$IE = IC$$

Figura 3.46. Medición del hFE de un transistor con tester digital.

Figura 3.47. Medición del hFE para un transistor NPN.

Figura 3.48. Indicación del valor de hFE del transistor.

Obsérvese en la imagen anterior que la indicación del valor de hFE para un transistor 2N2222A expresa una ganancia de 238. Este valor concuerda con la ganancia típica para los transistores denominados de baja potencia, cuyo rango de hFE está determinado entre:

TIPO	SERIE	hFE típico
NPN	2N2222A	75-300

Figura 3.49. Valor de hFE típico para un transistor 2N2222A.

Para el robot seguidor de línea blanca a implementar en el Capítulo 5, (Figura 5.22) se emplea una variante del transistor 2N2222A. El transistor CTPN2222A presentado en la Figura 3.44, cuya ganancia de corriente o hFE típico es de:

TIPO	SERIE	hFE típico
NPN	CTP2N2222A	35-300

Figura 3.50. Valor de hFE típico para un transistor CTPN2222A.

Los valores típicos indicados para el hFE de los transistores será dependiente de la corriente de colector (IC) y de la tensión base emisor (VBE).

3.5.9 TRANSISTOR 2N2222A. CARACTERÍSTICAS

- Alta corriente – máx 800 mA.
- Transistor NPN switching.
- Encapsulado metálico TO-18.

3.5.10 SUBMÚLTIPLOS Y NOTACIÓN CIENTÍFICA

En el ítem anterior se representó para el transistor 2N2222A una corriente máxima de 800 mA. Este tipo de notación es muy empleada en el área Técnica y en Ingeniería. Es también conocida como submúltiplos.

Dicho valor podría ser también representado empleando notación científica de la siguiente forma: 800×10^{-3} A

El beneficio de esta notación reside en la posibilidad de representar valores y magnitudes, evitando el empleo de la coma. O también para evitar la representación de tantos ceros en los casos en que los valores del componente o la magnitud medida sean muy pequeños, ya que podrá originar errores al acarrear el número.

Las unidades de submúltiplos más empleadas son:

$$m = mili = 1 \times 10^{-3} \text{ (Unidad)}$$

$$u = micro = 1x10^{-6} \text{ (Unidad)}$$

$$n = nano = 1x10^{-9} \text{ (Unidad)}$$

$$p = pico = 1x10^{-12} \text{ (Unidad)}$$

Figura 3.51. Submúltiplos más empleados en Robótica y Electrónica.

Para la descripción de la corriente del transistor, 800 mA corresponden a 0,800 A (se elimina la coma y se corren tres lugares a la derecha).

Los submúltiplos son muy empleados también en componentes, tales como los capacitores, en los cuales el faradio resulta una unidad muy grande. Entonces se aplican los prefijos de micro, nano o pico, dependiendo de la tecnología del capacitor en cuestión.

En el Capítulo 8, se verá que la corriente máxima que entrega cada pin del microcontrolador es igual a 25 mA o 0,025 A.

3.5.11 CONFIGURACIONES DE UN TRANSISTOR

Dependiendo del tipo de aplicación a realizar, los circuitos con transistores pueden ser diseñados en tres configuraciones básicas:

- Emisor común.
- Base común.
- Colector común.

Para las aplicaciones a desarrollar en esta obra, se emplea la configuración de emisor común. Esto significa que el terminal de emisor está conectado al común o negativo de la fuente.

Esta configuración es la requerida para emplear el transistor como dispositivo de conmutación, que es el modo de funcionamiento aplicado para el diseño de robots en esta obra.

3.6 CAPACITORES

El capacitor es otro de los componentes versátiles con los que cuenta la electrónica. Su empleo es tanto necesario para aplicaciones de filtrado como condicionante en el manejo de señales en las que interviene la frecuencia.

Conceptualmente, un capacitor consiste en dos placas eléctricas, separadas entre sí por un dieléctrico (material aislante entre placas).

La unidad de capacidad es el faradio (F).

A diferencia de las resistencias, los capacitores y las bobinas (presentes en filtros, transformadores, relés) no disipan energía. Los capacitores almacenan la energía en forma de campo eléctrico; las bobinas, en forma de campo electromagnético. Entonces la función básica de los capacitores es la de almacenar carga eléctrica (energía) y reingresarla al circuito cuando sea requerida.

Los tipos de capacitores más empleados en las aplicaciones de Robótica son los cerámicos, multicapa, electrolíticos, o de tantalio.

Para el desarrollo de robots en esta obra, serán empleados los capacitores multicapa en los Capítulos 8 y 9, en serie con el oscilador o cristal de cuarzo. La función de estos capacitores, denominados C1 y C2, es estabilizar la oscilación. Serán necesarios, junto al oscilador de cristal, para la configuración básica de un microcontrolador.

Su empleo es también imprescindible como capacitores de desacople para los circuitos de robots desarrollados en el Capítulo 9. Allí son implementados en la entrada y salida de la batería, alimentaciones del microcontrolador, y en paralelo con los motores, para filtrar los armónicos producidos como efecto de las cargas inductivas como los motores, lo cual afecta el funcionamiento del microcontrolador.

Frecuentemente, son empleados también dos capacitores junto a un inductor para diseñar un filtro conocido como Pi, dada la distribución de los componentes, (esto lo asemeja a la forma de la letra griega).

Figura 3.52. Símbolo eléctrico de un capacitor no polarizado.

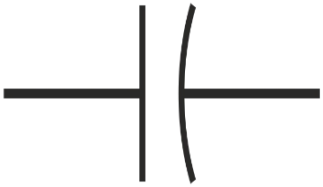


Figura 3.53. Símbolo eléctrico de un capacitor polarizado.

En la imagen anterior, la parte recta de la imagen corresponde al positivo del capacitor (lado izquierdo). La curvatura (lado derecho) corresponde al negativo.

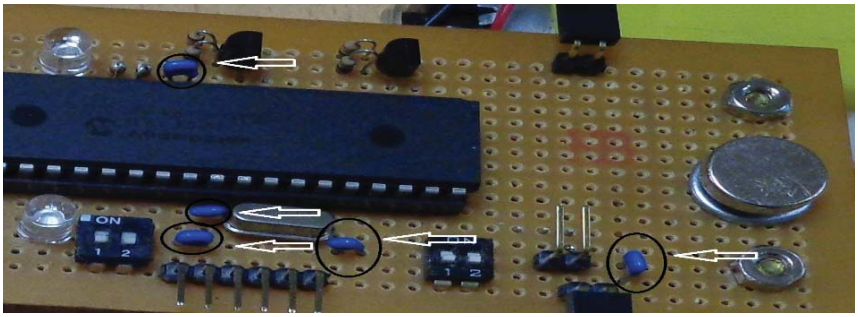


Figura 3.54. Empleo de capacitores en circuito de robot. Capítulo 9.

Al igual que ocurre con las resistencias, hay dos tipos básicos de capacitores: fijos o variables. Los capacitores variables son empleados mayormente en aplicaciones y equipos de telecomunicaciones.

Se debe tener en cuenta, para su empleo, que no todos los capacitores pueden ser utilizados para el trabajo en CA (corriente alterna) y CC (corriente continua).

Los capacitores cerámicos pueden ser empleados con CA y CC, ya que no son polarizados. Lo mismo ocurre con la tecnología multicapa.

Los capacitores electrolíticos y de tantalio son polarizados; esto significa que, de invertir su polaridad, explotan.

Las características más importantes a tener en cuenta en el trabajo con un capacitor son las siguientes:

Tensión de trabajo.	Es la tensión a la que puede trabajar un capacitor de forma permanente, sin sufrir deterioro. Este valor viene indicado en el cuerpo del capacitor.
Tensión de perforación.	Es la tensión máxima que puede soportar un capacitor, sin que se destruya su dieléctrico.
Tolerancia.	Al igual que en las resistencias, es una indicación de la variación que puede tener un componente en su valor.

Figura 3.55. Características eléctricas a tener en cuenta para un capacitor.

3.6.1 TIPOS Y CARACTERÍSTICAS

Serán presentadas a continuación las imágenes y características más importantes de los tipos de capacitores, que han sido descritos como los más empleados en Robótica.

3.6.2 CERÁMICOS

Figura 3.56. Capacitor Cerámico o Capacitor Cerámico de Disco.

Características:

- Capacidades, valores comprendidos entre:
  - Clase I: 1 pF a 100 pF.
  - Clase II: 150 pF a 47 nF.
- Tensión: 100 VCC.
- Bajo ruido.
- Alta estabilidad.

Los valores presentados dependerán del modelo y fabricante.

Las clases I y II, determinan para los capacitores cerámicos:

Clase I: ofrecen alta estabilidad y baja perdida para aplicaciones de circuitos resonantes.

Clase II: ofrecen alta eficiencia para aplicaciones de bypass, amortiguación y acoplamiento.

Identificación del valor: obsérvese en la Figura 3.56 que el capacitor posee la indicación de 103. Como estos capacitores comienzan en un rango de capacidad en picofaradios, el 3 es un indicador de la cantidad de ceros que se deben agregar. Por tanto, el valor sería 10000 pF, pero aplicando lo trabajado en el ítem 3.5.10, se debe correr tres lugares de derecha a izquierda para pasar a la unidad anterior (prefijo); es decir, nano. Entonces el valor resultante sería 10 nF.

3.6.3 CERÁMICOS MULTICAPA

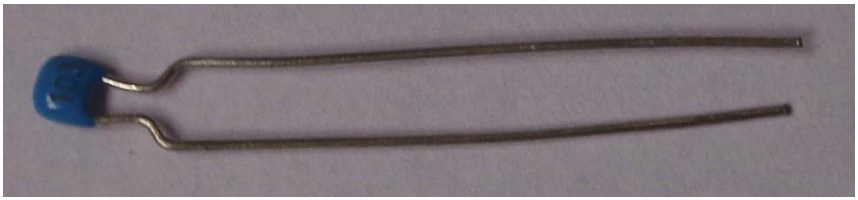


Figura 3.57. Capacitor Multicapa.

Características:

- Capacidades y tensiones de acuerdo con el tipo de dieléctrico:
  - X7R – 2.2 nF a 0.1 μF – 50 V.
  - Z5U – 1 pF a 1 μF – 50 V.
- Aplicaciones en circuitos sintonizados y filtros donde son requeridos baja pérdida y estabilidad.

Identificación del valor: obsérvese en la Figura 3.57 que el capacitor posee la misma indicación que el capacitor cerámico de disco presentado en la Figura 3.56. Por lo tanto, la lectura del valor será similar.

En otros casos, estos capacitores poseen directamente la indicación de 15, 27, 33, que hace referencia a valores en pF: 15 pF, 27 pF, 33 pF.

3.6.4 ELECTROLÍTICOS DE ALUMINIO

Figura 3.58. Capacitor Electrolítico de aluminio.

- Capacidades, valores comprendidos entre:
  - 0.47 µF – 10000 µF.
- Tensiones:
  - 6.3 VCC – 400 VCC.

3.6.5 ELECTROLÍTICOS DE TANTALIO

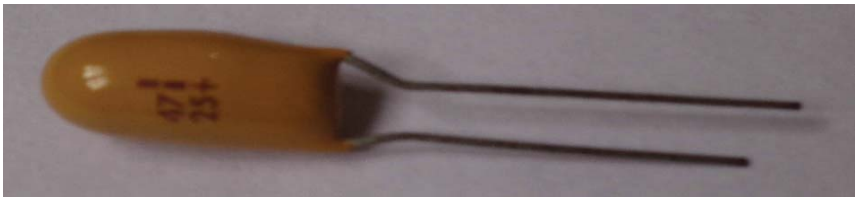


Figura 3.59. Capacitor de Tantalio.

- Capacidades, valores comprendidos entre:
  - 0.1 µF – 100 µF.
- Tensiones:
  - 16 VCC – 50 VCC.

3.6.6 MODOS DE FUNCIONAMIENTO

Se vio, en los ítems anteriores, la existencia de capacitores polarizados (electrolíticos) y no polarizados (cerámicos).

Una característica importante en el trabajo con capacitores es su comportamiento en función del tipo de corriente con el que se vaya a trabajar (recuérdese que para el trabajo en esta obra se emplea la corriente continua).

3.6.6.1 CAPACITOR EN CORRIENTE CONTINUA

En corriente continua, una vez que se carga; el capacitor actúa como una llave, es decir, se abre (y no permite la circulación de corriente).

Se debe considerar que el capacitor se carga al valor pico de la fuente. El valor pico es igual al valor de tensión de la fuente (batería) dividido por 0,707 o multiplicado por 1,41.

El tiempo de carga y descarga está determinado por una constante llamada Tau:

$$\tau = R \,.\, C$$

El tiempo de carga y descarga es de 5 tau. R es la resistencia a través de la cual el capacitor C se descarga.

3.6.6.2 CAPACITOR EN CORRIENTE ALTERNA

En corriente alterna, el capacitor se carga y descarga y presenta una resistencia, llamada reactancia capacitiva. Su valor dependerá de la frecuencia, tal cual se observa en la siguiente fórmula:

$$X_C = 1 / \omega \,.\, C$$

donde:

$$\omega = 2\pi f$$

Esto significa que la reactancia capacitiva aumenta con la corriente.

De los modos de trabajo del capacitor, se puede destacar que:

El capacitor se opone al paso de la corriente continua (CC) y permite el paso de la corriente alterna (CA).

3.7 OSCILADORES

Al trabajar en sistemas digitales, se destaca la existencia de dos tipos de sistemas o circuitos: combinacionales y secuenciales.

Los sistemas combinacionales son abordados en el Capítulo 7, con la presentación de los circuitos integrados de compuertas lógicas. Tienen como característica que la salida depende de la combinación de los valores o niveles lógicos de entrada.

En los sistemas secuenciales, la salida depende de una secuencia compuesta por los niveles lógicos o valores de entrada (estado presente), y un valor de realimentación (estado anterior). Por lo que, a diferencia de los sistemas combinacionales, los sistemas secuenciales poseen memoria.

Los Microcontroladores, (circuitos integrados digitales programables trabajados en el Capítulo 8) son sincrónicos. Requieren de una señal de onda cuadrada o pulsos de *clock* para su funcionamiento.

Para generar esta señal se empleará un cristal externo denominado oscilador. Es importante destacar que existen distintos tipos de osciladores. Los más empleados para aplicaciones con microcontroladores son los de XT (XTAL) cristal.

3.7.1 OSCILADOR DE CRISTAL

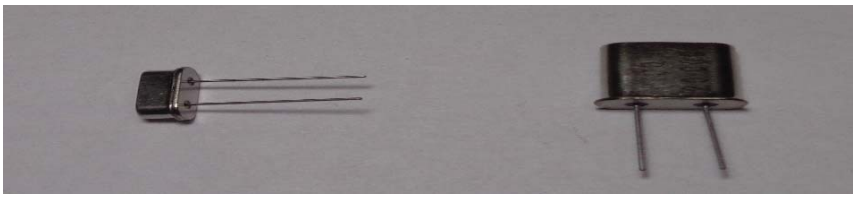


Figura 3.60. Cristal de perfil bajo (izquierda) – de perfil estándar (derecha).

El cristal es un material piezoeléctrico, con el cual se genera una señal eléctrica muy precisa. Su frecuencia determina la frecuencia de trabajo del microcontrolador.

Los modelos más empleados para aplicaciones de Robótica son los de perfil bajo y perfil estándar (Figura 3.60).

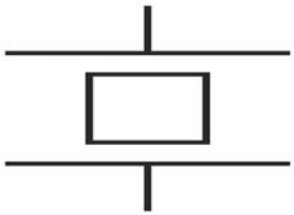


Figura 3.61. Símbolo eléctrico del Oscilador de Cristal.

La frecuencia máxima de cristal a emplear será indicada por el fabricante en la hoja de datos o *data sheet* del microcontrolador. Tómese como parámetro que los microcontroladores (Capítulo 8) pueden realizar millones de instrucciones por segundo.

3.8 SENSORES

Los sensores son transductores; esto significa que su empleo permite transformar una magnitud física en una señal eléctrica.

Existen, en la actualidad, una gran diversidad de tipos y modelos. Los más empleados para aplicaciones estándar de robótica son los siguientes:

LDR - Resistencia Dependiente de la Luz.
Sensor Óptico Infrarrojo.
Sensor Ultrasónico.

En muchos casos se procede al diseño de sensores, empleando los componentes de forma discreta (por separado).

Sensores de implementación con componentes discretos.

3.8.1 LDR – RESISTENCIA DEPENDIENTE DE LA LUZ

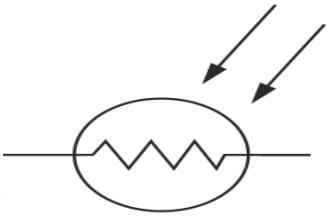


Figura 3.62. Símbolo eléctrico del LDR.

Los LDR (*Light Dependent Resistor*s) son un tipo especial de resistencia, que poseen la característica de que su valor resistivo disminuye con la incidencia de la luz.

Esta característica o propiedad es empleada en Robótica para implementar un divisor de tensión (Apartado 5.2.1), junto a otra resistencia en configuración *pull down* (Apartado 2.23).

De esta forma, cuando no incide luz en el LDR la tensión en el nodo de salida será de 0 V. Se obtendrá un valor de 5 VCC cada vez que incida luz sobre el LDR, ya que tiene su terminal superior conectado al positivo de la fuente. Al disminuir la resistencia, la corriente encuentra un camino de circulación.

Sin incidencia de luz, el LDR posee un valor resistivo muy alto, de varios MΩ. Luego, su valor puede reducirse a 50 Ω aproximadamente.

Figura 3.63. Prueba de LDR a emplear en Robot Seguidor de Luz.

En esta obra, se emplean LDR para la implementación del Robot Seguidor de Luz sin lógica adicional (Capítulo 5); Robots Seguidores de Luz con Lógica de Microcontrolador (Capítulo 9).

3.8.2 SENSOR ÓPTICO POR INFRARROJO

El sensor óptico por infrarrojo está compuesto por un LED infrarrojo (IR), y un fototransistor (presentados en el apartado 3.5.3).

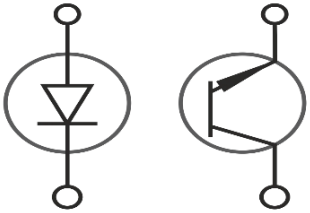


Figura 3.64. Composición interna del sensor CNY 70.

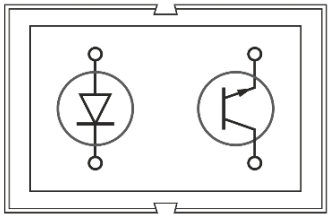


Figura 3.65. Símbolo eléctrico del Sensor Óptico Reflectivo CNY70.

El LED IR debe ser polarizado de la forma presentada en el Capítulo 2, es decir, empleando una resistencia serie RS a fin de limitarle la corriente.

Su principio de funcionamiento es muy simple: el LED IR emite un haz de luz IR (infrarrojo) reflectante. Si el haz encuentra una superficie clara, reflecta e ingresa

nuevamente al encapsulado, excitando la base del fototransistor. La intensidad del haz de luz será dependiente de los valores de polarización empleados.

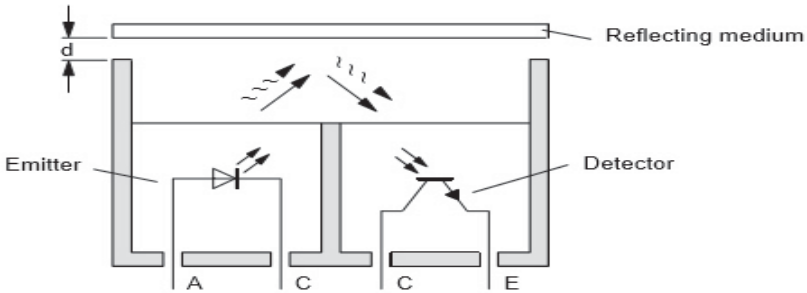


Figura 3.66. Sensor CNY70: representación de su principio de funcionamiento.

Prueba de funcionamiento:

Para el empleo de este sensor, es necesario tener identificados el LED IR y el fototransistor. Un método práctico de identificación es observar si el CNY70 posee una de sus dos ventanas más oscura, que corresponde al LED IR.

Una vez que se ha identificado el LED IR, se puede aplicar el método trabajado para el testeo de diodos con el tester. El objetivo es identificar su Ánodo (+) y Cátodo (-).

Para la implementación física, debe tenerse en cuenta la distribución de pines, ya que a diferencia de lo que ocurre en los transistores bipolares, el emisor del fototransistor está implementado (visto el sensor de frente) en el pin superior derecho (Figura 3.65).

Otra prueba rápida de funcionamiento consiste en verificar la emisión del haz de luz IR del LED, empleando la cámara del móvil (el ojo humano no percibe el espectro infrarrojo). Obsérvese, como ilustración, o ejemplo la siguiente imagen.

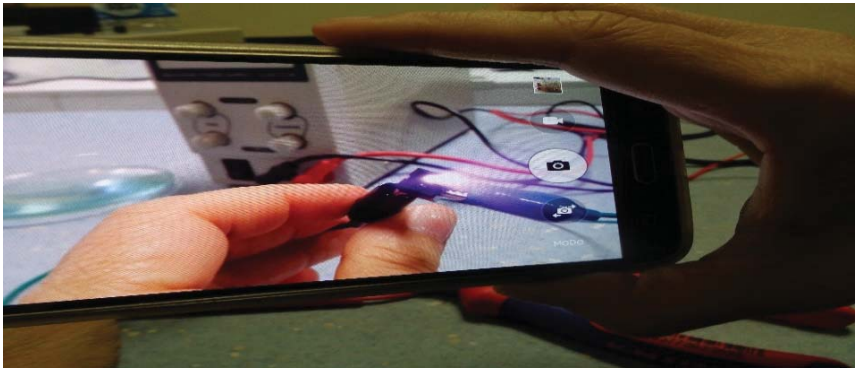


Figura 3.67. Prueba de sensor CNY70 con cámara del móvil.

Obsérvese en la Figura 3.67 otro método práctico para la verificación del sensor CNY70, en el cual se setea un valor seguro de tensión 2 V con una fuente de tensión regulada regulable. Luego se visualiza la emisión del haz de luz IR con la cámara del móvil.

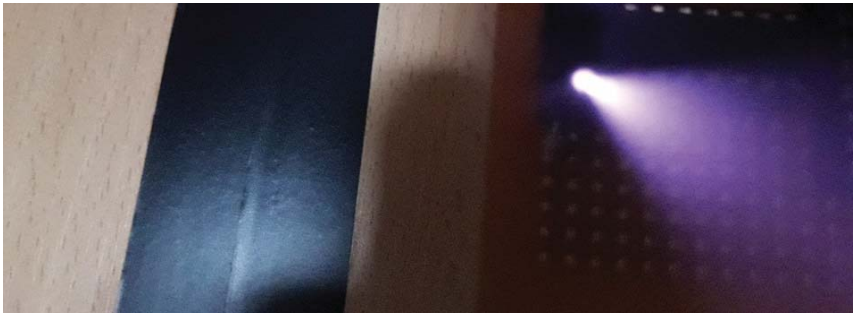


Figura 3.68. Sensor CNY 70: prueba de funcionamiento en la placa.

Comercialmente, se pueden adquirir otros modelos de sensores ópticos por infrarrojo con características similares al CNY70; entre ellos, podemos destacar:

- QRD1113/1114: descrito en su hoja de datos como Sensor de Objeto Reflectivo.
- TCRT5000: descrito en su hoja de datos como Sensor Óptico Reflectivo con Salida de Transistor (ídem al CNY70).

Otra indicación importante es la representada en las hojas de datos de la ***sensing distance*** o **distancia de sensado**. Se la establece en la hoja de datos del CNY70 como 3 mm, respecto a los 12 mm establecidos para el TCRT5000.

En la práctica, ocurre que, al implementar robots seguidores de línea, estos no responden al principio de funcionamiento establecido. Una de las causas más frecuente son las diferencias entre los modelos del sensor elegido.

De la experiencia en la fase de implementación práctica, desprenden las siguientes diferencias entre los modelos comerciales más empleados para el sensor CNY70.

SENSOR CNY70	
MODELO	CARACTERÍSTICAS
V69	Problema en la lectura al ras o a corta distancia.
TFK082	La lectura del sensor es de corto alcance. Debe estar al ras.
TFK102	Buena lectura de cerca y de lejos.

Figura 3.69. Sensor CNY70. Características de lectura de los diferentes modelos.

Es frecuente en la práctica no contar con el modelo de sensor requerido para nuestra aplicación. Así, resulta necesario implementar modificaciones al hardware, para compensar las diferencias en la lectura.

En la siguiente imagen se muestran las modificaciones introducidas al robot seguidor de línea blanca (implementado en el apartado 5.3). El objetivo es probar la distancia de sensado de los CNY70 empleados.

Se busca encontrar la distancia óptima de lectura.

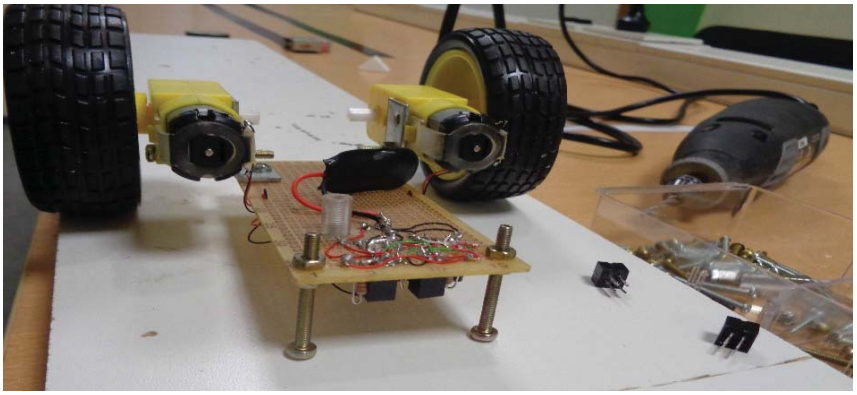


Figura 3.70. Prueba de distancia de sensado para los sensores CNY70.

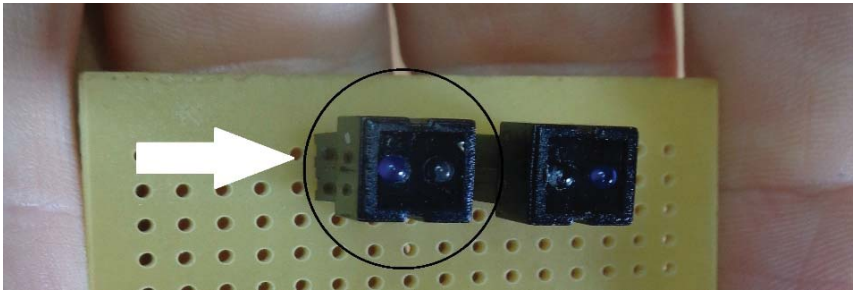


Figura 3.71. Forma visual de identificación del diodo IR y el fototransistor.

Obsérvese en la imagen que el lado más oscuro (azul en este modelo) corresponde al LED IR. De izquierda a derecha (visto de frente), la pata superior corresponderá al ánodo; la inferior, al cátodo.

Enfrentado, el lado derecho (visto de frente), la pata superior corresponderá al emisor del fototransistor, y la inferior, al colector.

Esta característica puede variar de acuerdo con el modelo del sensor. En este caso, se debe recurrir a la identificación del diodo con la lectura del tester.

Otro dato importante para comparar y analizar respecto de la hoja de datos para el sensor CNY70 y el TCRT5000, es el CTR *Current Transfer Ratio* o Relación de

Transferencia de Corriente. Esta indica un 5% para el CNY70 y un 10% para el TCRT5000.

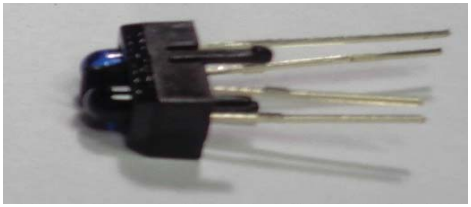


Figura 3.72. Sensor óptico reflectivo TCRT5000.

3.8.3 SENSOR ULTRASÓNICO

Los sensores por ultrasonido permiten detectar objetos y calcular la distancia a la que se encuentran. Al igual que ocurre con la señal de infrarrojo, que no es percibida por el ojo, las señales de ultrasonido operan por encima del ancho de banda del oído humano (20 Hz a 20 kHz). Por tanto, no resultan perceptibles para las personas.

Los rangos de distancia estándar con los que operan estos sensores oscilan entre los 2 cm - 400 cm; con una precisión de alcance de 3 mm.

Los módulos incluyen transmisores ultrasónicos, receptor y circuito de control.

Su principio básico de funcionamiento es el siguiente:

1) Se envía utilizando el disparador de entrada/salida, una señal de 10 µs.
2) El módulo envía automáticamente 8 pulsos de una frecuencia de 40 kHz, y detecta si alguna de las señales de pulso se ha retrasado.
3) Si la señal es devuelta a nivel alto, la duración del tiempo de salida alta IO es el tiempo desde el envío de ultrasonidos hasta el retorno.
   Test de distancia = (tiempo de nivel alto x velocidad del sonido (340 m/s)/2).

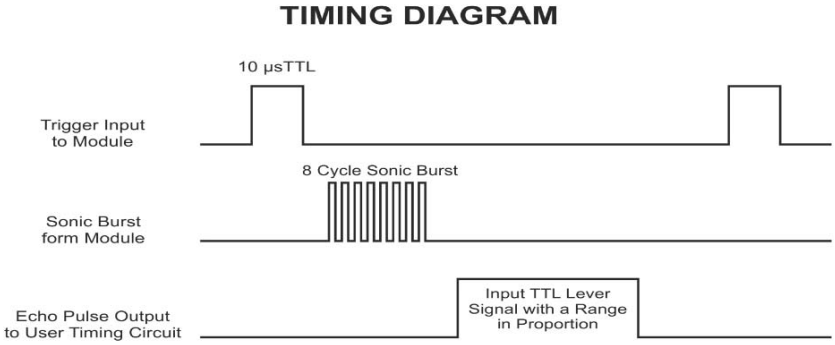


Figura 3.73. Diagrama de tiempo del módulo de ultrasonidos HC – SR04.

Otras características para este sensor son:

Tensión de trabajo.	**DC 5V.**
Corriente de trabajo.	15 mA.
Frecuencia de trabajo.	40 kHz
Rango máximo.	4 m.
Rango mínimo.	2 cm.
Ángulo de medición.	15 grados.
Señal de entrada de disparo.	Pulso TTL de 10 µs.
Señal de eco de salida.	Entrada de señal de palanca y el rango en proporción.
Dimensiones.	45 x 20 x 15mm.

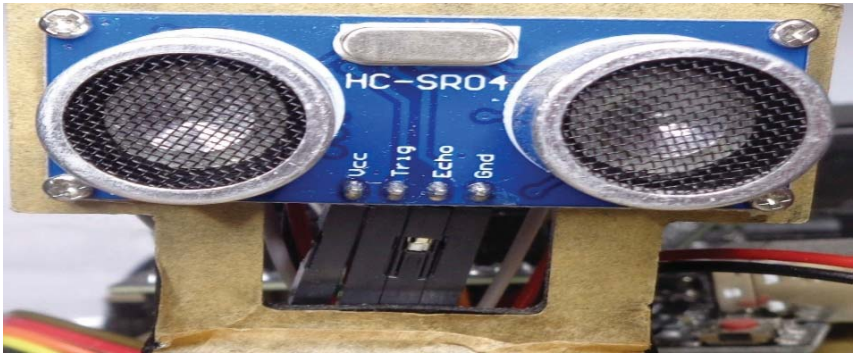


Figura 3.74. Sensor HC-SR04 empleado en robot compatible Arduino (Cap. 10).

Se observa en la imagen anterior la imagen del sensor HC-SR04 empleado en la Figura 10.63 robot compatible Arduino.

En el apartado 6.11.2, se presenta el empleo de bibliotecas (o librerías), que ofrecen al programador el beneficio de contar con aplicaciones de software ya resueltas (reutilización). Estos son conceptos muy empleados en la Ingeniería del Software.

Este es uno de los grandes beneficios propuestos por la tecnología Arduino, que cuenta con una gran comunidad de usuarios a nivel mundial. Ellos adhieren a compartir sus desarrollos bajo el paradigma GNU\GPL.

Dentro de la comunidad Arduino, se podrán encontrar librerías para sensores de ultrasonido, que serán de utilidad para iniciarse en el empleo de este sensor. Se encontrarán también las aplicaciones de software desarrolladas por los fabricantes de tecnologías compatibles con Arduino.

Una aplicación recomendable para iniciar los primeros pasos en el empleo de este sensor es visualizar su funcionamiento en un LCD **Display de Cristal Líquido**.

Una vez que se ha profundizado en el manejo de la programación, y en el caso que se desee desarrollar sus propias librerías, será preciso profundizar en la lectura de la hoja de datos del sensor. El propósito es analizar las características y tiempos de las señales a ser procesadas.

Comercialmente, se dispone de otros modelos de sensores de ultrasonido, como los modelos:

- SRF04. Este sensor posee 5 pines, uno de los cuales no tiene implementación. Su principio de funcionamiento es similar al HC-SR04.
- TR40-16. El T4016 y R4016 son una pareja de transmisor y receptor de ultrasonido, que operan a 40 kHz. Este transductor utiliza las propiedades piezoeléctricas de la ingeniería cerámica que proporciona alta presión de sonido y alta sensibilidad.
- TS601P01. Este sensor posee similar principio de funcionamiento que los sensores HC-SR04 y SRF04; sin embargo, su encapsulado tiene solo tres pines.

3.8.4 SENSORES DE IMPLEMENTACIÓN CON COMPONENTES DISCRETOS

Es necesario, en ocasiones, recurrir a la implementación de sensores empleando componentes discretos (por separado). Esta situación puede responder a:

- Reglamentos existentes en las ligas de Robótica.
- Necesidad de abaratar costes de implementación.
- Voluntad de integración de conocimientos.

Entre los componentes empleados para tal fin se pueden citar:

- LED infrarrojo.
- Fototransistores.
- Fotodiodos.

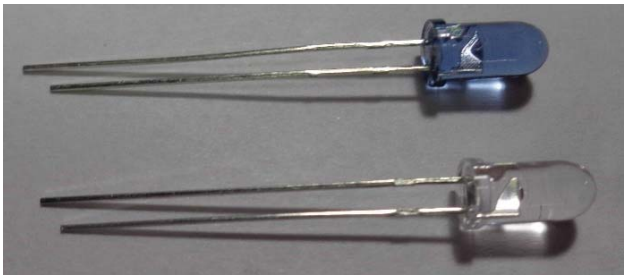


Figura 3.75. LED IR (vista superior) y fototransistor (vista inferior).

3.9 PLACAS MULTIPROPÓSITO O EXPERIMENTALES

Para el montaje de robots, en esta obra se emplean las placas conocidas como multipropósito o experimentales (son conocidas también como Universales). Los métodos y técnicas de montaje serán presentados en el Capítulo 5. Allí serán vistos los beneficios que posee el empleo del método de montaje cableado para el diseño de prototipos. Este método requiere de las placas multipropósito para el montaje y soldado de los componentes del circuito.

Estas placas fueron diseñadas para el trabajo y montaje con tecnología de componentes THT, presentada en el apartado 3.2.

Las tecnologías de materiales más empleadas para estas placas, y con mayor presencia en el mercado, son las siguientes:

- Pertinax.
- Epoxi.

Comercialmente, pueden ser adquiridas en las medidas de 5 x 10 cm y 10 x 20 cm. Las placas pueden ser adaptadas a las dimensiones y formas requeridas para la implementación de robots, empleando el torno de mano y los discos de corte que serán presentados en el Capítulo 4.

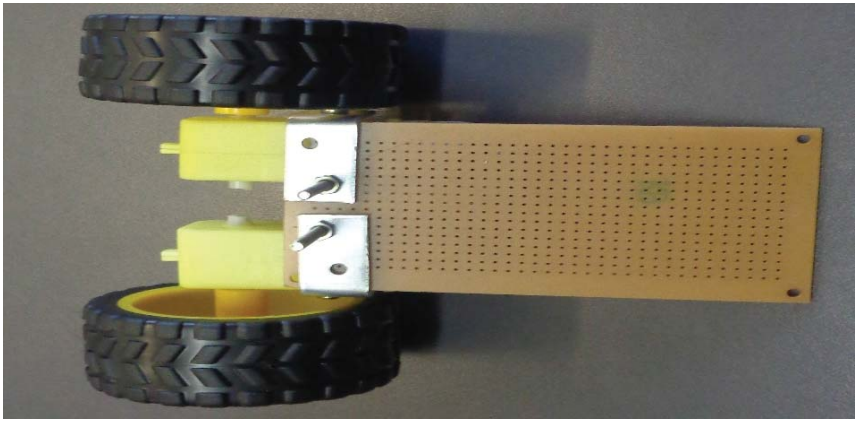


Figura 3.76. Placa de Pertinax empleada para el montaje de chasis de robot.

Obsérvese en la imagen que la placa posee una cara lisa, que será empleada para el montaje de los componentes.

La otra cara de la placa está dividida en islas de cobre diseñadas para el soldado de los componentes del circuito.

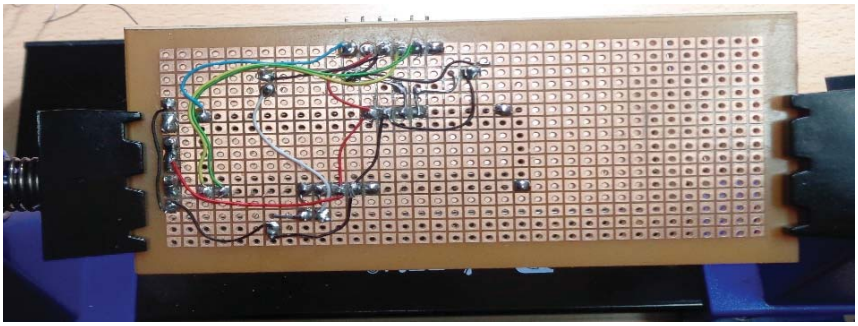


Figura 3.77. Islas de cobre empleadas para el soldado de componentes.

Comercialmente, pueden ser adquiridos modelos de placas provistas de perforaciones en las esquinas. Sin embargo, no es aconsejable su adquisición porque se perdería libertad en el diseño.

Las perforaciones pueden ser realizadas tomando con el calibre (Capítulo 4) las medidas de diámetro requeridas para seleccionar la mecha.

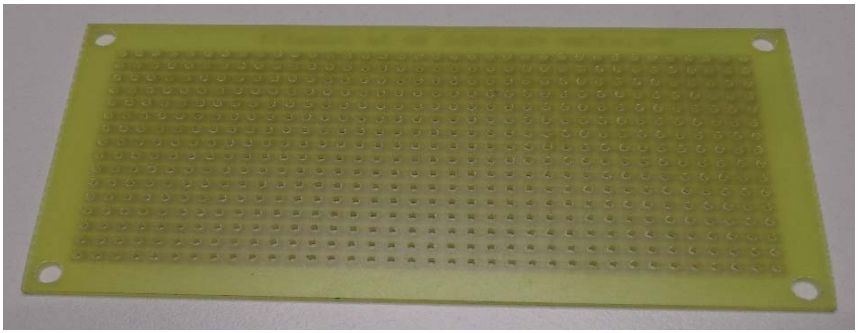


Figura 3.78. Placa Epoxi con perforaciones.

La placa epoxi puede diferenciarse fácilmente de la placa de Pertinax debido a su mayor rigidez, dada por su tecnología de materiales.

3.10 ZÓCALOS

Los zócalos son empleados en las aplicaciones de Robótica para la implementación de robots con lógica de circuitos integrados. Debido al coste que tienen, no deben ser soldados a la placa de forma directa para evitar daños por efecto de la temperatura, o bien permitir un recambio fácil ante una eventual falla en su funcionamiento.

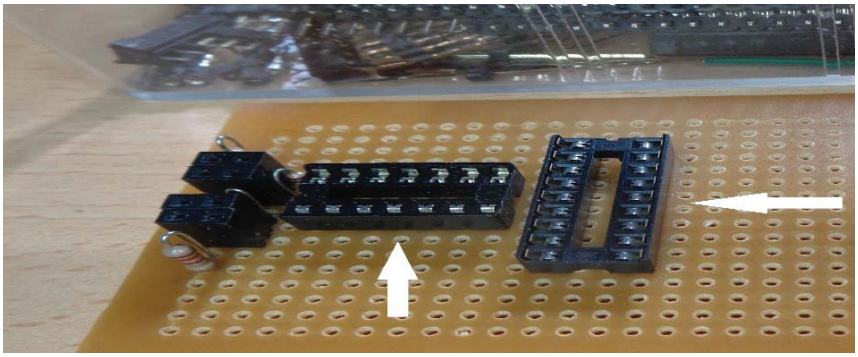


Figura 3.79. Zócalos DIP en el montaje de robot con CI (Capítulo 7).

En la figura se observa el empleo de zócalos DIP – Doble Hilera de Pines,

La denominación de los zócalos hace referencia a la cantidad de pines que posea el CI en cuestión. Por ejemplo, para el Microcontrolador 16F84A (Capítulo 8 – Microcontroladores) de 18 pines, la denominación del zócalo corresponde a **zócalo DIP 2 x 9**, es decir, dos hileras de 9 pines.

Como será tratado en el Capítulo 7, los CI poseen un punto y una muesca para hacer referencia a la pata o pin 1 (la cuenta se inicia desde la izquierda, pata superior). Del mismo modo, se debe hacer coincidir la pata 1 con la muesca que presentan los zócalos en su parte superior. Si así no fuera, se corre el riesgo de insertar el CI al revés, con su consecuente daño.

Figura 3.80. Punto y muesca para indicar la pata 1 del CI.

La tecnología *wire wrapping*, presentada a continuación, posee sus propios zócalos, diseñados con el diámetro de pin necesario para no dañar la pinza.

3.11 WIRE WRAPPING

Wire wrapping es una técnica de montaje de componentes electrónicos muy empleada para el diseño y desarrollo de prototipos. Permite unir dos o más puntos (nodo) eléctricos o de transmisión de datos.

Fue muy empleada en la producción de prototipos en los años '60 y '70, ya que su empleo podía ser más fiable que los circuitos impresos. La razón es que las conexiones de *wire wrapping*, son menos afectadas por la vibración o presión física en la placa. Se reduce, además, la resistencia eléctrica, debido a la ausencia de un material de soldadura.

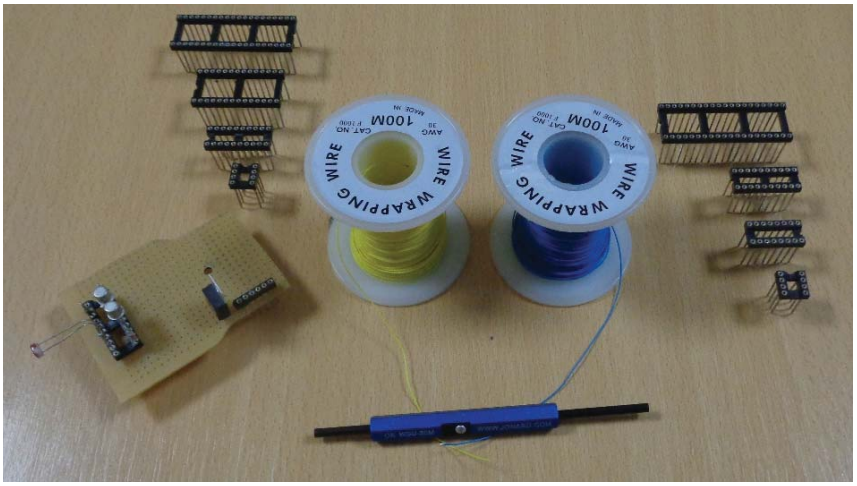


Figura 3.81. Sistema de *wire wrapping* aplicado al ensamble de Robots.

El empleo de *wire wrapping*, permite la producción de prototipos o circuitos finales.

Esta tecnología posee sus propias herramientas de trabajo: pinza, zócalos, cables.

3.11.1 PINZA DE WIRE WRAPPING

La pinza de *wire wrapping* posee tres herramientas clásicas integradas en una. Ellas son:

- Soldador (se enhebra el cable).
- Desoldador (el cable se puede desenhebrar).
- Alicate (la pinza posee una hoja de corte).

Figura 3.82. Pinza de *wire wrapping* en proceso de montaje.

3.11.2 ZÓCALOS DE WIRE WRAPPING

Los zócalos para tecnología de *wire wrapping* han sido desarrollados para que sus patas coincidan con el diámetro interno de la pinza. El fin es no dañarla.

Poseen un largo estándar, que permite asegurar la unión eléctrica e insertar diferentes puntos de nodo, de acuerdo con los requerimientos del circuito.

Los zócalos pueden ser adquiridos en formato DIP, a partir de 2 x 2.

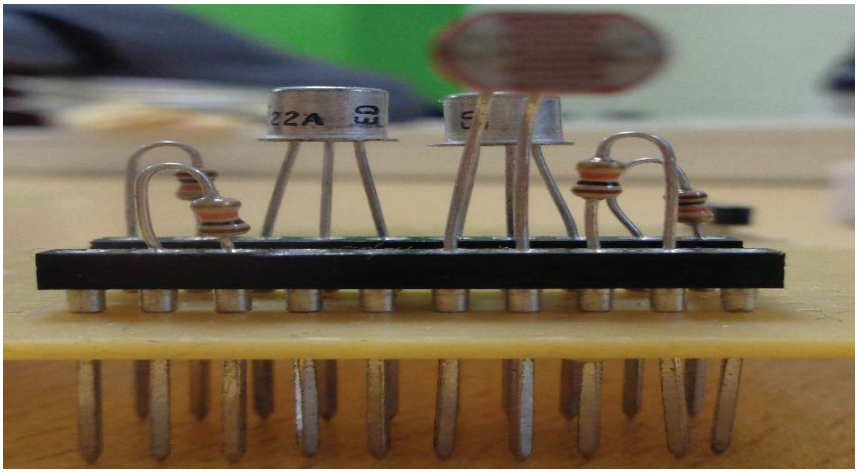


Figura 3.83. Zócalo de *wire wrapping*. Montaje de robot seguidor de luz.

3.11.3 CABLE DE WIRE WRAPPING

El cable de *wire wrapping* es empleado tanto para trabajar con sistemas 100% *wire wrapping* o para trazar las pistas en el método de Montaje Cableado.

Como beneficio de empleo, destacan su diámetro reducido y su flexibilidad.

Comercialmente, puede ser adquirido en diferentes colores. Esto permite identificar las distintas etapas de un circuito. En las actividades con microcontroladores, permite identificar de forma rápida las conexiones de un puerto u otro.

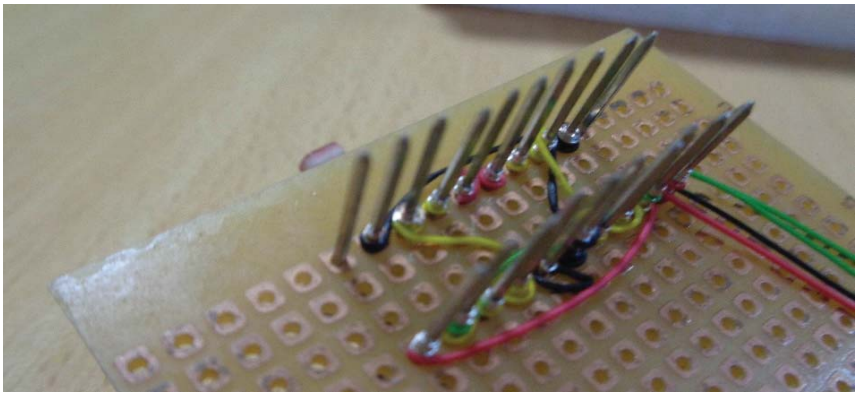


Figura 3.84. Cables de *wire wrapping* en zócalo para montaje de robot.

3.12 BORNERAS PARA PLACA DE 2 TERMINALES

Como se observará en la imagen siguiente, las borneras para placa de 2 terminales pueden ser empleadas para:

- El conector broche de la batería.
- Conexión de cables del motor.

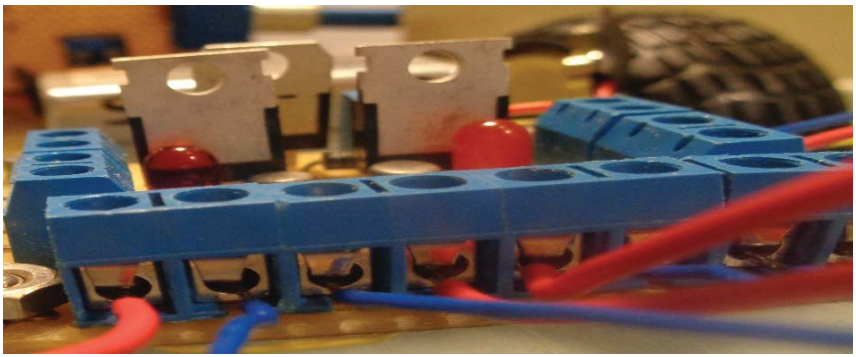


Figura 3.85. Borneras para placa de 2 terminales en montaje de robot.

Los diámetros de los terminales de la bornera coinciden con los diámetros para la conexión de componentes en las islas de la placa.

3.13 BATERÍAS

En el apartado 2.3 del Capítulo 2 se señaló que los componentes y la electrónica requieren de corriente continua para su trabajo. Se vio también que la corriente continua se encuentra disponible en pilas y baterías.

En este punto vale destacar que la batería es una agrupación (batería) de pilas en serie, (véase Figura 2.21).

Se afirmó también que los programas simuladores de electrónica presentan la batería con el símbolo:

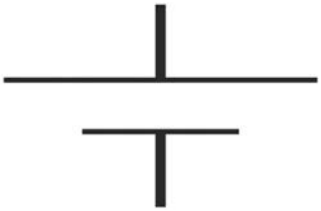


Figura 3.86. Símbolo eléctrico empleado para pilas y baterías.

Sin embargo, el símbolo eléctrico de las baterías puede ser representado en alusión a la integración de pilas, de la siguiente forma:

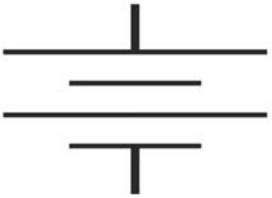


Figura 3.87. Símbolo eléctrico empleado para la batería.

En ambos símbolos el terminal más largo corresponde al positivo.

De actualización constante, son las opciones de tecnologías, corrientes y tensiones para las baterías empleadas en Robótica.

Es claro que puede ser tentador adaptar distintos tipos o tecnologías de pilas para el montaje de robots, incentivados por la reducción de tamaño, facilidad de montaje de su zócalo (porta pila), tensiones de trabajo cómodas para conectar en serie y consecuentemente evitar el empleo del CI regulador. Sin embargo, esta práctica debe ser descartada bajo la consideración del uso para el cual las baterías han sido diseñadas.

Figura 3.88. Pilas y zócalo. Deben ser consideradas las especificaciones técnicas.

Una opción recomendable para el trabajo inicial en Robótica es el empleo de baterías de 9 V recargables, cuya corriente de trabajo es, por lo general, de 250 mAh o 0,25 Ah. Esto significa que entregan un cuarto de Amper durante una hora. En algunos modelos, el fabricante garantiza un valor Amper hora mínimo (Figura 2.23). Este valor puede ser tomado en la etapa de diseño para calcular sobre la peor condición.

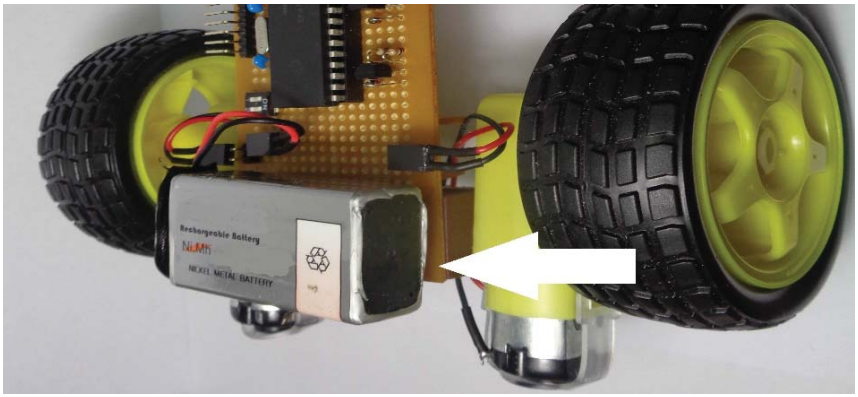


Figura 3.89. Batería recargable de 9 V - 250 mAh en circuito de Robot.

Tal vez al principio, conocer (la corriente de trabajo de la batería), no sea muy representativo. Pero se verá a continuación, en el desarrollo del ítem dedicado a motores, que los motores estándar actuales en el mercado tienen una corriente de trabajo de 125 mA ó 0,125 A, equivalente a un octavo de Amper por hora. Entonces bastará el empleo de una sola batería para abastecer (estimativamente) al robot durante ese tiempo de uso. Vale considerar que los circuitos han sido diseñados para emplear la menor cantidad de componentes y para el caso de los CI son de bajo consumo.

Figura 3.90. Opción de cargador para batería recargable.

No todas las baterías recargables poseen la misma tecnología. Esto significa que el cargador debe haber sido diseñado para recargar la tecnología de baterías que se haya adquirido.

Por lo general, puede ser adquirido el pack de batería y cargador.

No todas las baterías son recargables, por lo que no se debe intentar recargar una batería Alcalina.

Las tecnologías de baterías más empleadas en robótica son:

- Ni-MH Níquel-Metal.
- Li-ion Iones de Litio.

Las baterías de Li-ion poseen una carga más rápida y una mayor duración que el resto de las baterías.

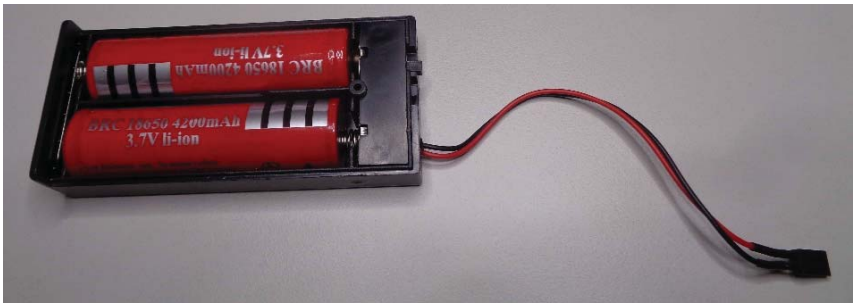


Figura 3.91. Pilas de Li-ion 3.7 V – 4200 mAh.

Obsérvese en la Imagen que las dos pilas en serie forman una batería de, aproximadamente, 7,4 V y 4200 mAh. Esta batería es empleada en el robot compatible Arduino (Figura 10.63).

3.14 CONECTOR BROCHE PARA BATERÍA

El conector broche para batería es un accesorio necesario para conectar la batería y energizar el circuito.

Tradicionalmente, se adquiría el conector y se conectaban los cables a una bornera de dos terminales. Pero, actualmente, vienen provistos de un conector hembra para poder conectar y desconectar. Esto simplifica el montaje, tal como se observa en la imagen anterior, (Figura 3.91).

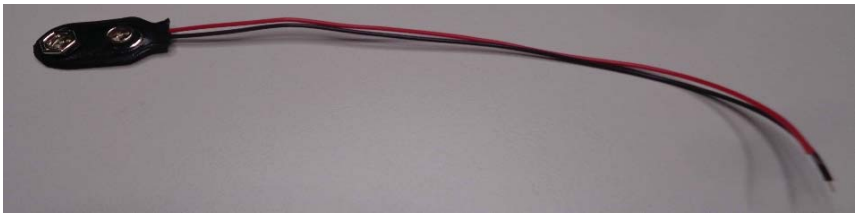


Figura 3.92. Conector broche para batería.

Se verá para la implementación de robots, se puede emplear un alojamiento hembra por dos pines para placa y termocontraíble, para armar el conector broche con terminal de conexión hembra.

3.15 IMÁN DE NEODIMIO

El imán de neodimio es empleado por el autor para la implementación de robots, como un método práctico de sujeción de las baterías al chasis. Esto evita emplear métodos poco prácticos, como la sujeción de las baterías con precintos.

En el Capítulo 11 (Impresoras 3D - Aplicaciones en Robótica), se modela un gabinete para batería, que es empleado para generar otras terminaciones de montaje para los robots.

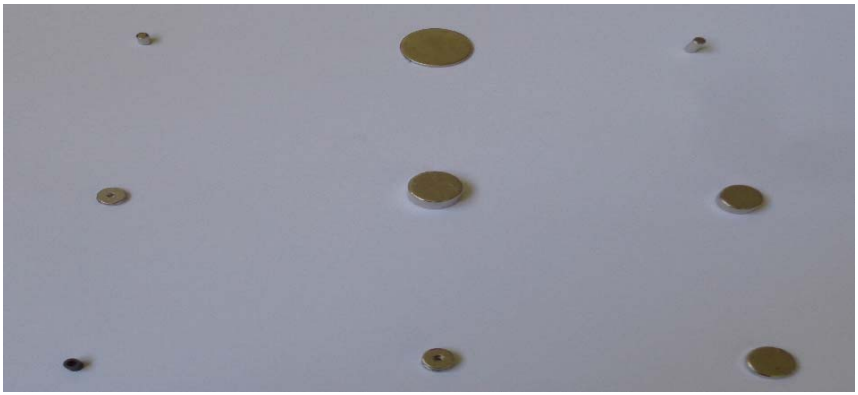


Figura 3.93. Diferentes modelos de imanes de Neodimio.

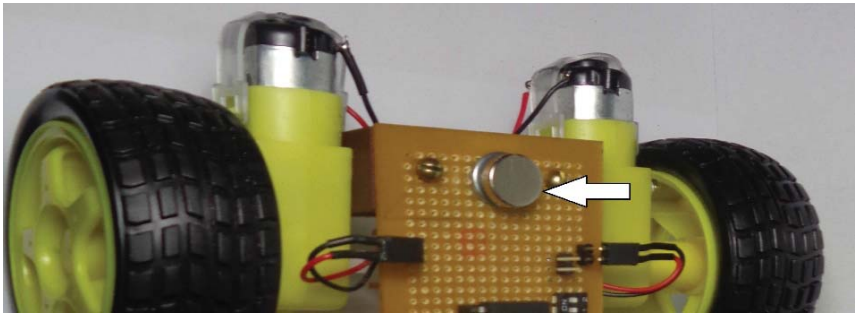


Figura 3.94. Imán de Neodimio empleado en montaje de robot.

Una opción para sujetar el imán a la placa es montar un tornillo de cabeza plana de diámetro similar al imán, centrado en la placa.

3.16 TORNILLOS, TUERCAS, ARANDELAS, BUJES

Para el montaje de robots es fundamental contar con todos los accesorios necesarios. A modo de ejemplo, obsérvese en la siguiente imagen: el empleo de tornillo de cabeza plana para la sujeción magnética del imán de neodimio a la placa.

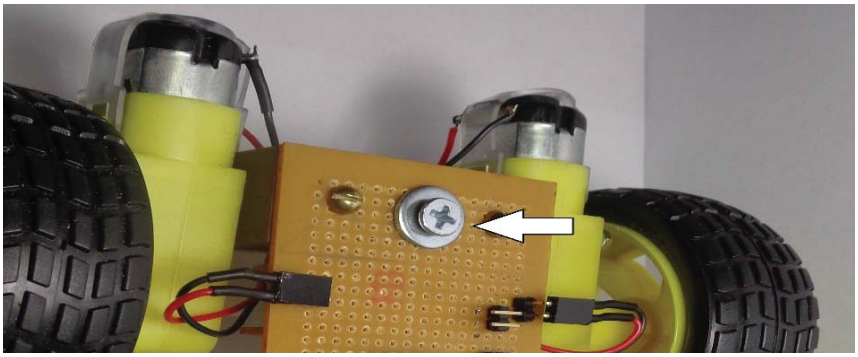


Figura 3.95. Tornillo de cabeza plana para sujeción magnética del imán.

Se observa el empleo de una arandela para distribuir la presión que ejerce el tornillo sobre la placa, y brindar mejor agarre. En la parte posterior, se emplea una tuerca de diámetro correspondiente al tornillo.

Los tornillos más empleados según el tipo de cabeza son:

- Plana.
- Phillips.

Las partes de un tornillo son:

- Cabeza.
- Cuello.
- Rosca.

Para la implementación de robots, en esta obra, serán listados en el BOM o Listado de materiales, los tornillos, tuercas y arandelas necesarios para el montaje de cada modelo.

Lo aconsejable es aprovisionarse con el tiempo de los tipos y medidas de empleo frecuente. Así también contar con diferentes tamaños y modelos para innovar en el diseño.

En las casas de electrónica son comercializados kits con los elementos necesarios para distintas aplicaciones, como, por ejemplo: sujetar el disipador de calor del CI regulador. El kit viene provisto de tornillo, mica, tuerca, buje.

Figura 3.96. Compartimento de gavetero con diferentes tipos de accesorios.

A diferencia del trabajo con ruedas, para el desarrollo de los ítems batería y motores no se profundiza en cuestiones de diseño, ya que la implementación de baterías y motores es un proceso industrial. Esto significa que su diseño excede el objetivo de esta obra, que es el diseño y aplicación de robots.

Así, la información desarrollada tiene como objetivo guiar al usuario en la adquisición correcta de los elementos de trabajo.

3.17 MOTORES

Para el trabajo en Robótica, se utilizan motores eléctricos para convertir energía eléctrica en energía mecánica.

Existen, en disponibilidad, gran cantidad de modelos de motores para Robótica. Sin embargo, se debe tener en cuenta, desde el principio, la necesidad de contar con ruedas, cuyos diámetros de encastre coincidan con el diámetro del eje del motor.

Los tipos de motores más empleados en aplicaciones de Robótica son:

- Corriente continua.
- Servomotor.

Compuestos por bobinados de alambre de cobre.

Principio de funcionamiento, movimiento por causa del electromagnetismo.

3.17.1 MOTOR DE CORRIENTE CONTINUA

- Es el más simple de todos.
- Los más simples y pequeños solo poseen bobinados en el rotor.

- A medida que pierde velocidad, aumenta su torque. Por eso, es muy estable en términos de velocidad.

Ventajas:

- Alta capacidad de torque.
- Control simple desde la electrónica.

Desventajas:

- Mantenimiento.
- Alto coste.
- Tamaño.
- Bajo rendimiento por rozamiento y generación de calor.

3.17.2 SERVOMOTOR

Se emplean para ubicar el eje en una determinada posición.

Compuesto por:

- Motor de corriente continua.
- Caja reductora.
- Placa controladora.

Por lo general, su recorrido está acotado a 180º (se puede modificar eliminando la chaveta que actúa como traba).

3.17.3 MOTORES PARA APLICACIÓN EN ROBÓTICA

De forma práctica, las características más importantes a tener en cuenta para la elección de un motor para aplicaciones en Robótica con dispositivos móviles son:

- Tensión de trabajo.
- Corriente.
- Torque.
- Velocidad.

Una alternativa es adquirir inicialmente las líneas de motores estándar, los cuales vienen en par con la rueda.

Figura 3.97. Motorreductor y rueda estándar empleados en montaje de robot.

Las especificaciones técnicas para este motor son las siguientes:

Voltaje	Parámetros	3 V DC	5 V DC	6 V DC
Parámetros del motor (sin caja reductora)	RPM	6000 RPM		
	Corriente	80-100 mA		
Parámetros de la caja reductora	Reducción	48:1		
	Velocidad sin carga	125 RPM	200 RPM	230 RPM
	Velocidad con carga	95 RPM	160 RPM	175 RPM
	Torque de salida	0.8 kg.cm	1.0 kg.cm	1.1 kg.cm
	Corriente	110 -130 mA	120-140 mA	130-150 mA
	Diámetro máximo de rueda	6.5 cm		
	Dimensiones	70 mm x 22 mm x 18 mm		
	Peso	50 g		
	Ruido	<65 Db		

Figura 3.98. Especificaciones para el motorreductor de la Figura 3.97.

Como puede observarse en la figura, estos motores son de baja corriente, pero permitirán implementar los primeros robots.

3.17.4 DIFERENCIA ENTRE VELOCIDAD Y TORQUE

Al iniciarse en las aplicaciones de Robótica, ocurre que, al reciclar los motores de algún coche o juguete disponible, se observa que la rueda provista originalmente, gira a gran velocidad. Esto supone un buen funcionamiento para nuestros robots; sin embargo, al realizar su montaje, se observa que el motor vibra, pero no produce movimiento.

Este efecto se produce debido a que el motor tiene velocidad, pero no posee torque (fuerza).

Por esta razón, los motores de corriente continua a emplear en estos robots poseen una caja reductora, con la que, en función a una serie de cálculos sobre el empleo de engranajes, se reduce su velocidad para obtener la fuerza requerida.

A esta unión (motor de corriente continua y caja reductora) se la denomina **motorreductor**.

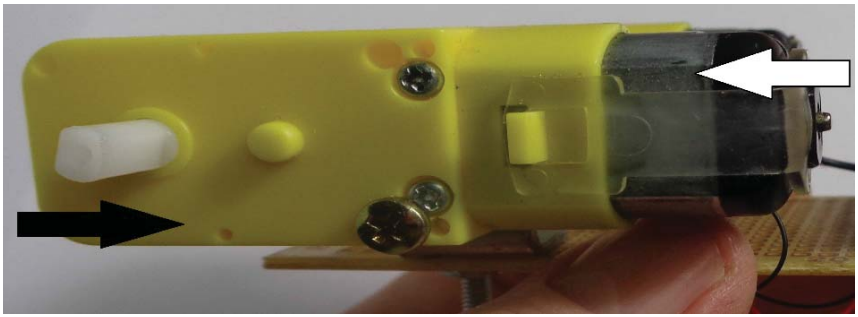


Figura 3.99. Caja reductora (izquierda) y motor de corriente continua (derecha).

En la imagen se observa la conformación de un motorreductor comercial estándar.

A continuación, se muestra la imagen de uno de los servomotores más empleados en la actualidad. Este se utilizará en el Capítulo 11 para controlar el ángulo de giro del sensor ultrasónico montado en el robot compatible Arduino.

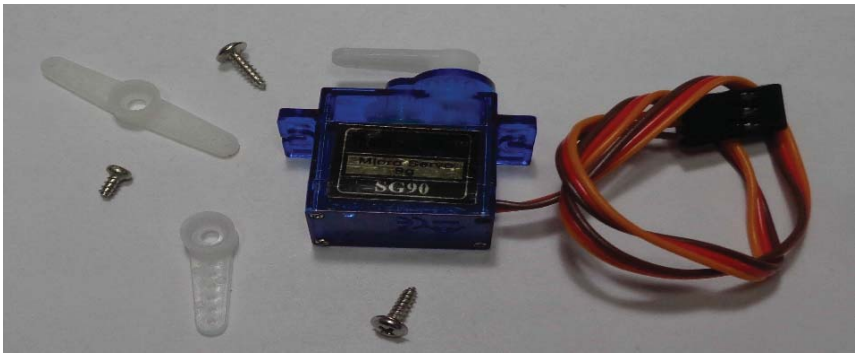


Figura 3.100. Servomotor de amplia aplicación en Robótica.

El servomotor de la imagen es empleado también en Robótica para implementar las articulaciones de actuadores (brazos).

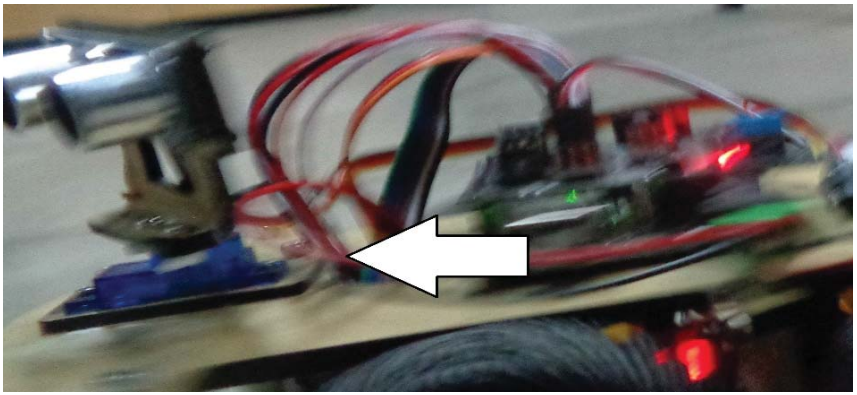


Figura 3.101. Servomotor empleado para controlar el ángulo de giro del sensor.

3.18 ESQUINERO ÁNGULO

Los esquineros ángulo son empleados para el montaje de los motores a la placa o chasis. Comercialmente, pueden ser adquiridos en diferentes formas y tamaños. Esto, claramente, significa un beneficio a la hora del diseño.

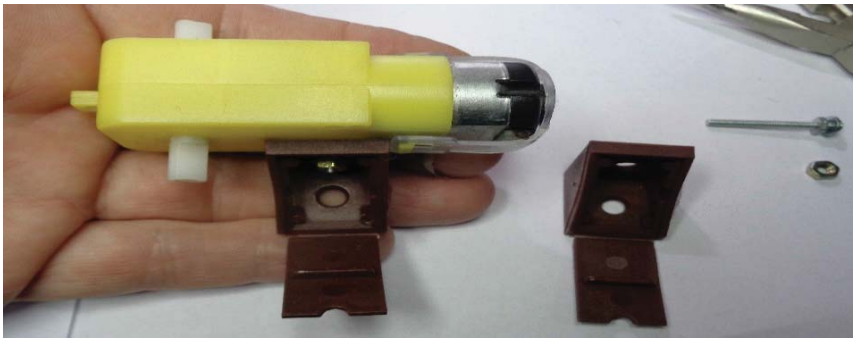


Figura 3.102. Esquineros ángulo empleados para el montaje de motores.

Para la implementación de algunos modelos de robots, se emplean esquineros ángulos metálicos. El beneficio del esquinero ángulo plástico reside en no atraer al imán de neodimio, y, al mismo tiempo, reducir el peso del robot.

3.19 RUEDAS

Las ruedas cumplen un rol fundamental en la implementación en Robótica. Resulta una condición necesaria que el diámetro de sus ejes coincida con el diámetro del eje de los motores, lo que dificulta el reciclado de piezas.

En aplicaciones de Robótica avanzada, entrarán en juego todos los factores físicos; por lo tanto, deberá ser considerado el tamaño de las ruedas en función de la velocidad requerida, el tipo de material y el rozamiento.

Será necesario recurrir también a la fabricación de moldes de madera para la fabricación de ruedas con cubiertas de caucho de silicona, requeridas para aplicaciones de sumo y mini-sumo.

Otra aplicación particular es desarrollada en el Capítulo 11, donde se presenta el modelado de una llanta de características similares a las más difundidas en el mercado, empleando impresión 3D.

3.19.1 MODELOS Y CARACTERÍSTICAS

La primera división en los tipos de ruedas empleadas para aplicaciones de robótica es la siguiente:

Ruedas para los motores.
Rueda loca.

3.19.2 RUEDAS PARA MOTORES

Para el montaje inicial de robots, lo más común es adquirir las ruedas disponibles en stock de acuerdo con los diferentes lugares de residencia.

Pueden ser adquiridas las ruedas que vienen en par con el motor.

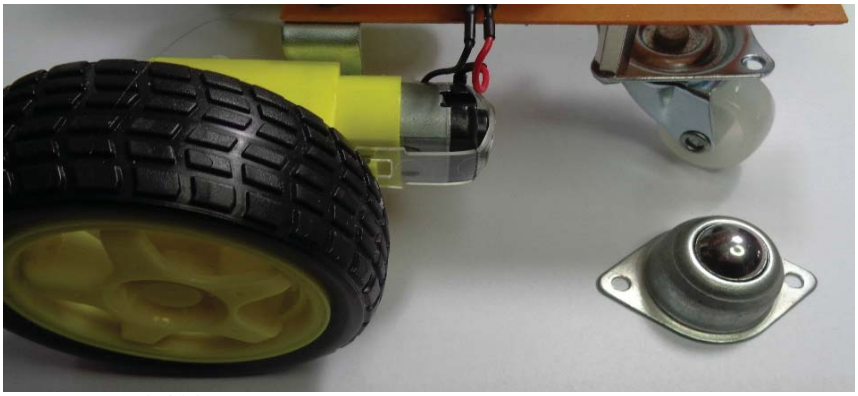


Figura 3.103. Rueda y motorreductor junto a opciones de rueda loca.

Las características más importantes para este modelo de rueda son las siguientes:

Altura.	65 mm
Ancho de rueda.	26.6 mm
Diámetro externo de eje.	11 mm
Encastre eje motor.	3.6 x 5 mm

3.19.3 OMNI WHEEL – RUEDA OMNIDIRECCIONAL

En la actualidad, pueden ser adquiridas ruedas como la *omni wheel* (rueda omnidireccional), que permiten desmontar el eje para adaptar diferentes diámetros de motor.

3.19.4 RUEDA LOCA

Entre los modelos más empleados, se encuentran las clásicas ruedas de nylon, blancas o negras. Obsérvese la siguiente imagen:

Figura 3.104. Ruedas locas. Material de fabricación nylon con soporte de acero.

Datos a tener en cuenta para estas ruedas:

Peso de carga.	10 kg
Altura total (estándar).	33 mm
Ancho de rueda (estándar).	13 mm
Diámetro orificio fijación.	4.5 mm

La opción más recomendable para ruedas locas es emplear la rueda loca de metal, que evita la dificultad de que se trabe entre un giro y otro, como suele ocurrir con las clásicas ruedas locas de nylon.

La desventaja que posee la rueda loca de metal es que por lo general resulta más difícil de conseguir.

Figura 3.105. Rueda loca de metal.

Figura 3.106. Rueda loca de metal vista de abajo.

Obsérvese en la Figura 3.107 que pueden ser empleados distintos tamaños de separadores metálicos o plásticos, con la finalidad de regular la altura requerida o necesaria de la rueda loca.

Figura 3.107. Separadores empleados para regular la altura de la rueda loca.

3.19.5 OPCIONES AVANZADAS DE DISEÑO DE RUEDAS

Cuando los requerimientos son más específicos, se recurre a otro tipo de métodos avanzados, que permiten obtener diferentes tipos de ruedas, llantas o cubiertas, fuera de los modelos estándar del mercado. Entre estos métodos, se puede destacar el modelado e impresión 3D, y el diseño de moldes.

3.19.5.1 MODELADO E IMPRESIÓN 3D

Una alternativa cada vez más requerida es el diseño de ruedas y partes empleando Impresión 3D. La actividad será trabajada en el Capítulo 11 Impresoras 3D - Aplicación en Robótica, donde se describe el proceso para modelar una llanta.

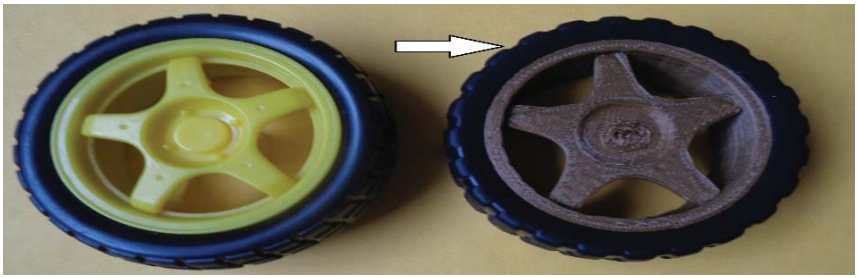


Figura 3.108. Llanta (derecha) modelada empleando impresión 3D.

3.19.5.2 DISEÑO DE MOLDES

Otro método avanzado empleado para el diseño de ruedas propias es el desarrollo de moldes para implementar cubiertas con caucho de silicona.

Figura 3.109. Molde de madera y llanta de aluminio implementados en el IM.

Figura 3.110. Llanta de aluminio.

Figura 3.111. Molde de madera para el diseño de cubiertas con silicona.

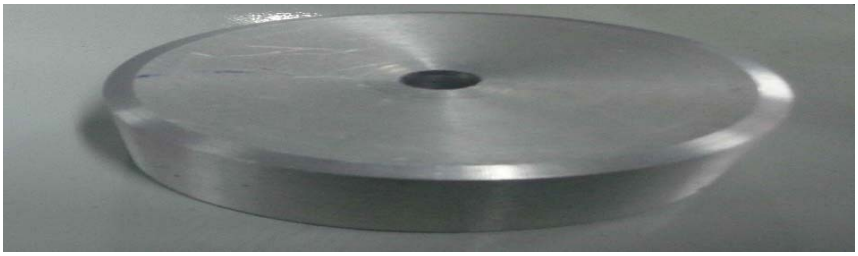


Figura 3.112. Pieza de aluminio para modelado de llanta.

Figura 3.113. Caucho de siliconas empleado para implementar cubiertas.

Figura 3.114. Cubierta implementada con caucho de siliconas.

3.20 PATAS METÁLICAS

Como puede observarse en la Figura 3.107, las patas metálicas son empleadas como separadores, o para regular la altura de la placa, rueda loca o motor en su montaje.

Comercialmente, pueden ser adquiridas en diferentes alturas.

Figura 3.115. Pata metálica y tuerca

3.21 TERMOCONTRAIBLES

El material termocontraible es empleado en la terminación y aislación de soldaduras. Comercialmente, se encuentran disponibles en diferentes diámetros.

Son empleados también en los robots seguidores de luz para aislar los LDR.

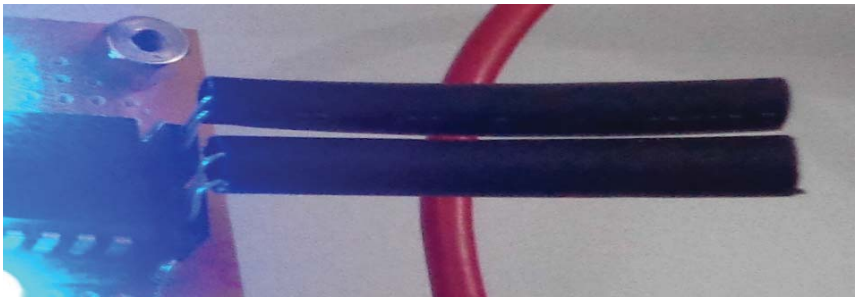


Figura 3.116. Termocontraibles empleados en robot seguidor de luz.

3.22 ESTAÑO

El estaño es un componente muy importante en la implementación de robots, ya que de la calidad de la soldadura depende el buen funcionamiento de los dispositivos.

Las características más importantes a la hora de elegir el rollo de estaño para trabajar son:

- Diámetros 0.7 mm, 0.35 mm.
- 60/40:Sn/Pb 60% de estaño y 40% de plomo.
- Almas 6.

Figura 3.117. Rollo de estaño de 0,35 mm.

CAPÍTULO 4
HERRAMIENTAS E INSTRUMENTAL PARA ROBÓTICA

4.1 SOLDADOR Y ESTACIÓN DE SOLDADO

El soldador es una herramienta empleada para derretir el estaño, y de esta forma, permite soldar o fijar los componentes a las placas.

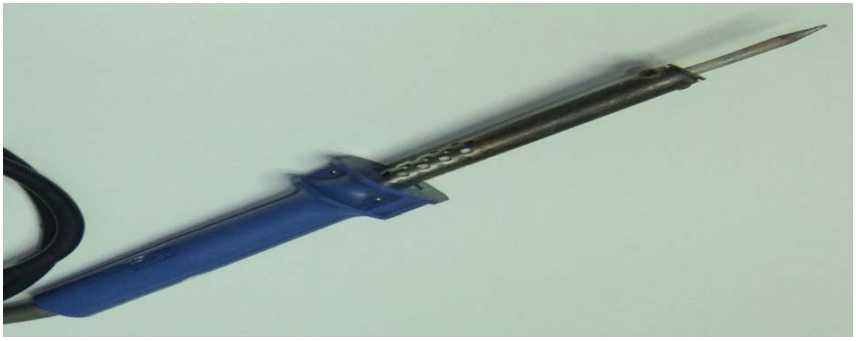


Figura 4.1. Soldador tipo lápiz con punta de cerámica.

La potencia del soldador es un dato para tener en cuenta. Los valores más comunes son los de 30 o 40 W (watt).

Para microelectrónica, se recomienda el empleo de un soldador tipo lápiz, de 30 W y punta de cerámica.

En el trabajo con componentes de tecnología SMT, se emplean las estaciones de soldado. Poseen una punta más fina. Permiten regular la temperatura y adoptar diferentes boquillas para extraer CI y componentes.

Figura 4.2. Estación de Soldado.

Figura 4.3. diferentes boquillas.

4.2 SOPORTE DE SOLDADOR

Contar con un soporte de soldador seguro es un factor primordial. De esta forma, se previenen accidentes y el deterioro de los componentes.

Como se observa en la figura, el soldador debe quedar con la punta hacia abajo y a resguardo de cualquier contacto ocasional. La esponja es empleada para limpiar la punta.

Figura 4.4. Soporte de soldador.

4.3 CUTTER

Esta herramienta de corte será útil durante el trabajo con PCB o placas multipropósito, para generar separación entre pistas, componentes, o remover gotas de estaño.

Figura 4.5. Cutter (cúter en español) de cuchilla segmentada.

4.4 LIMAS DE MATRICERO

Las limas de matricero son un tipo especial de limas. Se emplean en electrónica para limar los terminales de los componentes y, de esta forma, obtener mayor rozamiento al momento de la soldadura.

Son útiles también para reducir el diámetro de los componentes.

Figura 4.6. Lima de matricero.

4.5 ALICATE

El alicate es una herramienta de corte. Dependiendo del diámetro de corte, recibe la denominación de alicate de corte blando, semiduro o duro.

Figura 4.7. Alicate de corte superpuesto al ras.

Se debe tener en cuenta el diámetro de corte para no dañar la herramienta. El diámetro está expresado en mm.

Ejemplo para el alicate de la figura: max Cu 1,6 – Fe 1,0.

4.6 PINZA DE PUNTA PLANA

Esta pinza, cuya punta es plana, se emplea para sujetar el cable a cortar o enderezar los terminales de los componentes.

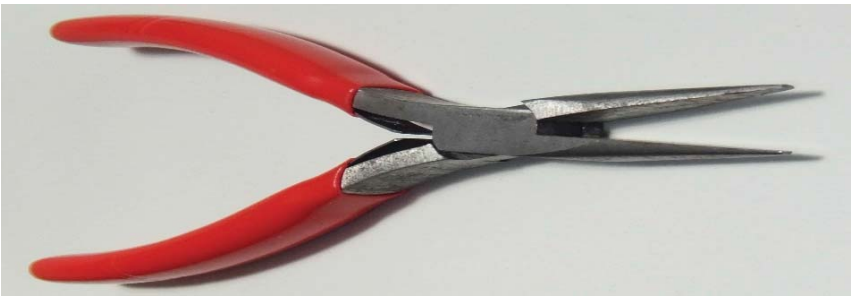


Figura 4.8. Pinza de punta plana.

4.7 SET DE DESTORNILLADORES PERILLEROS

El set de destornilladores constituye una herramienta útil y necesaria en las implementaciones de Robótica. Son necesarios al conectar los motores a las borneras y el ensamble de chasis.

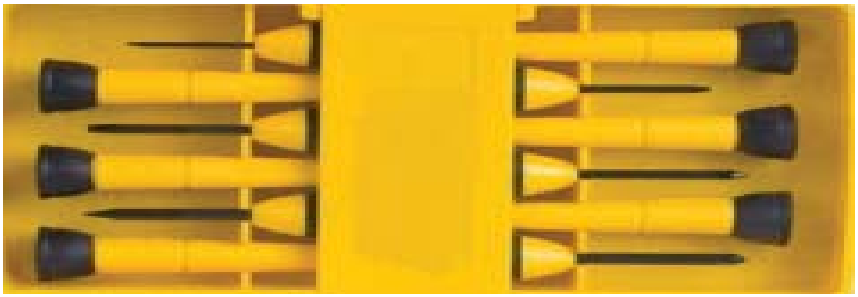


Figura 4.9. Set de Destornilladores perilleros.

Por lo general, este set de destornilladores está compuesto por cuatro destornilladores de punta plana estándar de 1.4 mm, 2 mm, 2.4 mm, 3.0 mm. Y también dos destornilladores punta Phillips PH0 y PH1.

4.8 BRUSELAS

La brusela es un tipo de pinza de tamaño pequeño. Pueden ser adquiridas en set o kits, al igual que los destornilladores o de forma discreta.

Son fabricadas con diferentes tipos de materiales y se emplean en robótica para manipular CI o en aplicaciones que requieren de alta precisión.

Figura 4.10. Pinza brusela de punta recta.

4.9 CALIBRE

Al momento de personalizar los robots, y en su fase de montaje, se necesitará tomar medidas de los diámetros internos y externos de los componentes. Para este fin, una herramienta imprescindible es el calibre.

El calibre de la Figura 4.11, posee escala Vernier, que se emplea para obtener mayor precisión en la medida.

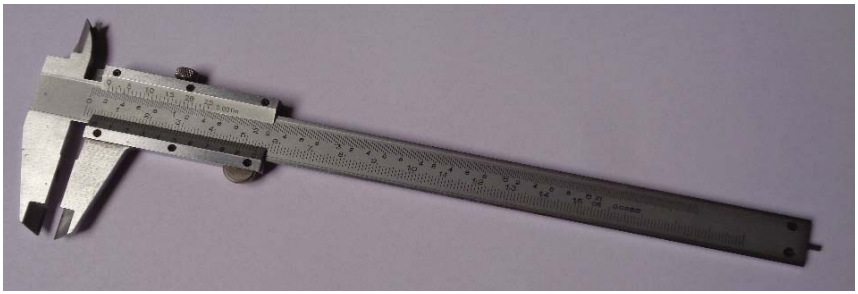


Figura 4.11. Calibre con escala Vernier.

4.10 TORNO DE MANO

El torno de mano es una herramienta imprescindible para el trabajo en diseño y montaje de robots, dado que permite emplear las mechas para realizar perforaciones, cortes con los discos o emplear una amplia gama de accesorios.

Figura 4.12. Torno de mano con mandril.

En el mercado existe una amplia gama de modelos y marcas. Se pueden destacar como características:

RPM.	10.000 a 33.000/min
Ø	32 mm
W	125

Figura 4.13. Torno de mano. Especificaciones estándar.

4.10.1 MECHAS

La elección del diámetro, así como el material de las mechas para el trabajo en electrónica, es fundamental dado que, si se perfora la placa con una mecha de mayor diámetro que el del componente, es preciso luego recurrir al empleo

masivo de estaño para su sujeción. Otro factor muy importante es el material de fabricación de la mecha, dados los diferentes usos. En este campo, se requiere el acero rápido.

Figura 4.14. Mechas de acero rápido.

4.10.2 DISCOS DE CORTE

Para la fase de diseño en los robots, resulta importante considerar su diseño. La razón es que a pesar de emplear en una instancia avanzada una impresora 3D para modelar los chasis (Capítulo 11), se requerirá algún corte o modificación sobre la placa, y será necesario emplear discos de corte.

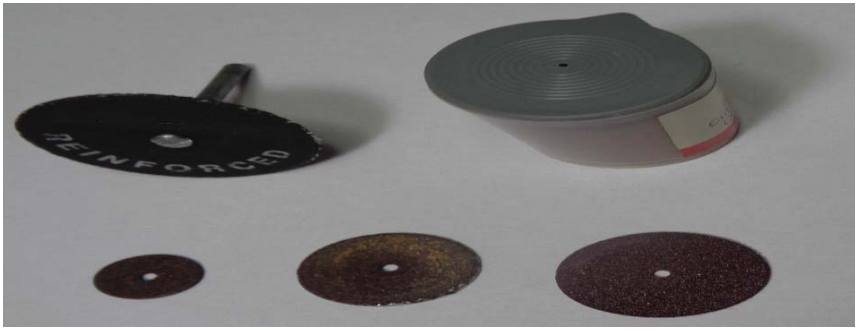


Figura 4.15. Discos de corte.

Es aconsejable, por cuestiones de seguridad y durabilidad, emplear los discos de corte reforzados, diseñados para tronzar y cortar sobre madera, metal, y plástico. Estos discos están fabricados en fibra de vidrio y requieren un mandril.

Figura 4.16. Disco de corte reforzado y mandril.

Dependiendo de su marca y modelo, se pueden tomar los siguientes parámetros:

Profundidad de trabajo máximo.	10 mm
Máximas rpm.	35.000 rpm
Diámetro de vástago.	3.2 mm
Diámetro de trabajo.	24 mm

Figura 4.17. Disco de corte reforzado – especificaciones.

Otra herramienta fundamental en toda instancia de diseño es la creatividad. Ella debe llevar a concebir adaptaciones en caso de ser requeridas, tales como la del disco de corte para trabajos duros, mostrado en la siguiente figura.

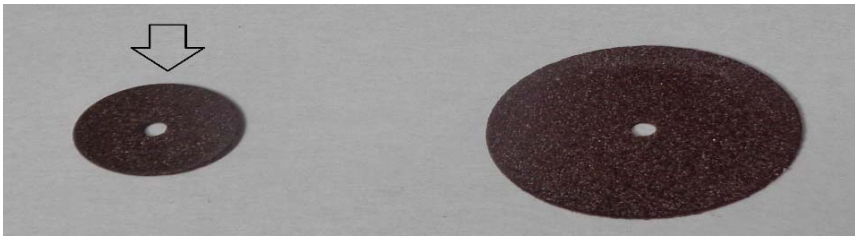


Figura 4.18. Disco de corte para trabajos duros. Adaptado a medida.

4.10.3 ACCESORIOS

Existe en el mercado una gran variedad de accesorios para los tornos de mano. Proveen versatilidad en el trabajo y permiten su adaptación a diferentes materiales y formas.

Figura 4.19. Accesorios para torno de mano.

4.11 INSTRUMENTAL DE MEDICIÓN

4.11.1 INTRODUCCIÓN

De la misma manera como ocurre con el osciloscopio, se podría profundizar en el estudio del tester, abarcando los contenidos de mediciones eléctricas y electrónicas, y tipos de errores. Sin embargo, para continuar con la línea de estudio de este libro, se va a abordar el trabajo con estos instrumentos de medición para facilitar y asegurar su implementación práctica.

4.11.2 TESTER

El tester es un instrumento de medición. La denominación multímetro está asociada a su capacidad de realizar múltiples mediciones. Dentro de las más habituales, destacan: tensión, corriente, resistencia, continuidad.

Al igual que ocurre en la industria de la electrónica en general, los tester poseen cada vez mayor cantidad de funciones a menor coste.

Existen dos tipos de tester empleados para el trabajo en electrónica: los de tipo analógico y los de tipo digital. En esta obra, se trabajará con el tester digital, actualmente el más difundido.

Figura 4.20. Tester digital.

Para comenzar a operar con el tester, simplemente se debe llevar a cabo dos pasos iniciales.

El primero es considerar la magnitud de trabajo, y en función de esta, salvo la medición de un valor de corriente hasta 10 A (no habitual en nuestras actividades de microelectrónica), conectar, como segundo paso, el cable negro del tester en el COM (común o negativo), y el cable rojo en la indicación de las unidades V, Ω, mA, Hz.

Como se indica en la siguiente figura:

Figura 4.21. Terminales para conexión de cables o puntas.

Figura 4.22. Conexión de puntas a terminales.

Este método debe ser aplicado ante cada instancia de medición:

1 - Conexión de puntas a terminales.
2 - Selección de magnitud a medir.
3 - Selección de escala.

Figura 4.23. Pasos para realizar una medición con tester.

La siguiente imagen muestra la aplicación de los pasos 2 y 3 para la medición de una resistencia de carbón de 120 Ω.

Figura 4.24. Medición – Resistencia de carbón de 120 Ω.

En la imagen se verifica también lo trabajado en el ítem 3.3.2 resistencias fijas, tolerancia para una resistencia de carbón, que, vimos de acuerdo con su cuarta banda, podía variar en +/- 10%.

Otro factor para tener en cuenta es que el tester empleado no es autorango. Los tester autorango como su nombre lo indica, no necesitan la selección de una escala para realizar una medición, que presenta, entre otros, el beneficio de simplificar su interface de hardware.

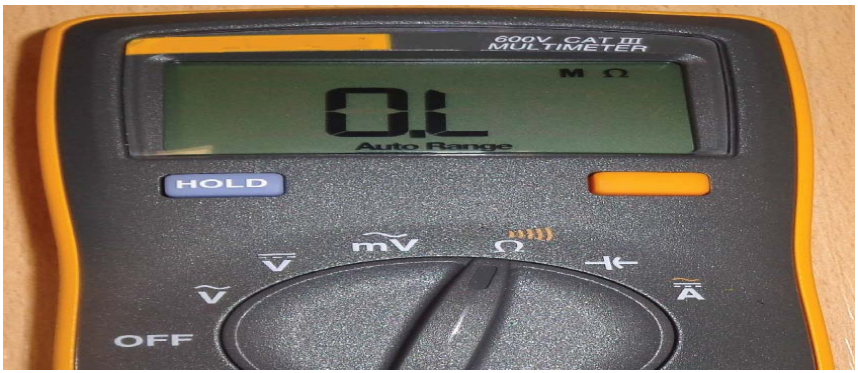


Figura 4.25. Tester autorango. Panel de selección.

4.11.3 OSCILOSCOPIO

El osciloscopio es un instrumento de medición empleado para analizar señales en función del tiempo.

Los primeros osciloscopios empleados fueron los analógicos. Actualmente están en uso, aunque se han expandido los osciloscopios digitales, dada su simpleza circuital, esto permitió también la disminución de costes.

El osciloscopio analógico es de gran tamaño dado que emplea un TRC o Tubo de Rayos Catódicos.

4.11.3.1 OSCILOSCOPIO ANALÓGICO

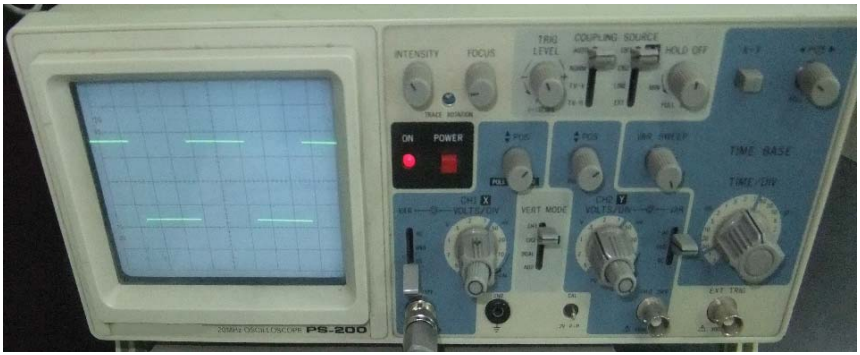


Figura 4.26. Osciloscopio analógico. Señal cuadrada.

Como se observa en la figura, este osciloscopio presenta en la parte central de la pantalla, un sistema de ejes cartesianos, donde x (horizontal - abscisa) representa el tempo, e Y (vertical – ordenada) representa el valor de la tensión.

Para el análisis de señales, recuérdese que, conociendo el tiempo se puede hallar la frecuencia, aplicando lo trabajado en el apartado 2.2.1 Frecuencia y 2.2.2 Período.

Entre las características de operación de este instrumento, se puede destacar que poseen atenuador de señal, dos canales, selector de rango o escala por canal, opción de *bypass* canal 1 – canal 2 – DUAL - ADD, control de intensidad, selector de acoplamiento de entrada CA – CC, GND para calibración de canal.

4.11.3.2 OSCILOSCOPIO DIGITAL

Figura 4.27. Osciloscopio digital. Señal cuadrada.

Mientras los osciloscopios analógicos trabajan de forma directa con la señal aplicada, los osciloscopios digitales emplean previamente un conversor analógico digital ADC. Almacenan digitalmente la señal de entrada y la reconstruyen posteriormente para su visualización.

Además de poseer un panel de control, el osciloscopio digital posee similares características de operación que el analógico. Cuenta con los beneficios de un sistema digital, como opciones de memorización, e interface a PC.

De los instrumentos presentados, el más empleado en las actividades de robótica es el tester. Pero, tanto el osciloscopio analógico como el digital, serán de gran utilidad para aplicaciones avanzadas, donde se deseen analizar las señales emitidas y recibidas por los sensores, pines de microcontroladores, funcionamiento del cristal o clk, entre otras.

4.12 ELEMENTOS DE SEGURIDAD PERSONAL

Es imprescindible para el trabajo en diseño y aplicación de robots, contar con los elementos de seguridad personal.

Entre los elementos básicos necesarios, pueden ser destacados:

- Anteojos o antiparras de protección ocular: en todo momento.
- Guantes: para manipulación de robots, perforación y corte empleando torno de mano.
- Barbijo: para el corte de placas universales o multipropósito.

Figura 4.28. Sergio Barreto. Empleo de protección ocular para el trabajo en robótica.

CAPÍTULO 5
DISEÑO E IMPLEMENTACIÓN DE ROBOTS SIN LÓGICA ADICIONAL

5.1 DISEÑO Y MONTAJE DE CIRCUITOS

5.1.1 INTRODUCCIÓN

Siguiendo el propósito de este libro, se abordará el diseño y aplicación de Robots. Con este objetivo, se van a integrar los contenidos expuestos en los Capítulos 2, 3, 4, a los métodos y técnicas de montaje aquí presentados.

5.1.2 MÉTODOS Y TÉCNICAS DE MONTAJE

Ya se ha expuesto que existen marcadas diferencias entre realizar un diseño, y realizar una producción a nivel comercial e industrial. Por tanto, esta actividad aplica al desarrollo de prototipos.

De acuerdo con el ámbito de aplicación, se puede establecer una clasificación inicial dentro de las siguientes categorías:

Competición en ligas de robótica.
Exposiciones en ferias tecnológicas o muestras de trabajo.
Entretenimiento personal.

Figura 5.1. Ámbitos de aplicación de los robots a desarrollar.

Definir una de las categorías o ámbitos de aplicación anteriores permitirá también evaluar el tipo de tecnología de montaje a emplear.

Si, por ejemplo, el objetivo es desarrollar un robot de sumo o mini sumo para participar en una liga, se debe tener en cuenta que en su impacto o colisión contra su adversario. Se buscará, cada vez que detecte su posición dentro del Tatami, desplazarlo a fuera e imponerse como el mejor de tres rounds.

Ante el marco de acción o escenario planteado, se debe realizar la elección del método o técnica de montaje de diseño, considerando factores como los esfuerzos mecánicos a los que estarán expuestos los componentes de la placa, por colisiones o impactos. Es así que se puede realizar la siguiente clasificación:

Método cableado.
Wire wrapping.
PCB.

Figura 5.2. Métodos y técnicas de montaje a emplear.

Es cierto que hay elecciones que se toman en función de la experiencia y el conocimiento de la actividad. Esto significa que si hace mucho tiempo se trabaja sobre el mismo diseño, conviene tener el circuito de base modelado en un Programa de Simulación de Circuitos Electrónicos y generar desde allí el PCB.

Ahora bien, en la práctica ocurre por ejemplo que, ante una avería o falla durante una competencia, se deben levantar componentes, puentear etapas, repasar pistas o modificar el circuito. De este modo, utilizar un impreso significaría, en este caso, una renovación constante de la placa. Mientras que, si emplea otros métodos como el cableado, las pistas son los mismos cables. Y en la medida en que posea espacio en la placa, no hace falta levantar un componente para su reemplazo, ya que puede insertar otro en el espacio disponible de la placa.

Por otra parte, salvo la limitación a nivel de costes, el método por excelencia a nivel de prototipado es *wire wrapping*.

Se puede realizar también una combinación de técnicas. A continuación, su desarrollo y presentación.

5.1.3. MÉTODO DE MONTAJE CABLEADO

Este método consiste en presentar (montar, distribuir) los componentes en una placa multipropósito o experimental. Se elige la posición, para cada uno de ellos, en función del análisis preliminar realizado, respecto al montaje de motores y batería. Posteriormente, se sigue el circuito eléctrico, soldando e interconectando los componentes con un cable, que actúa como una pista (camino para la circulación de corriente).

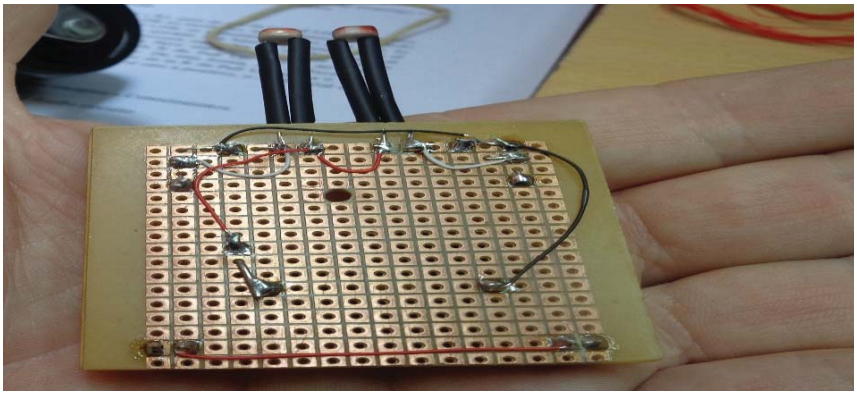


Figura 5.3. Método de montaje cableado para robot seguidor de luz.

5.1.4 MONTAJE CON WIRE WRAPPING

El sistema de *wire wrapping* fue presentado en el Capítulo 3, y como se vio, provee un sistema de montaje similar al método de montaje cableado, con la excepción de que no emplea soldadura, por el empleo de la pinza y los zócalos especiales para esta tecnología.

Se observa en la figura siguiente el empleo del sistema *wire wrapping*, para la implementación de un robot seguidor de luz.

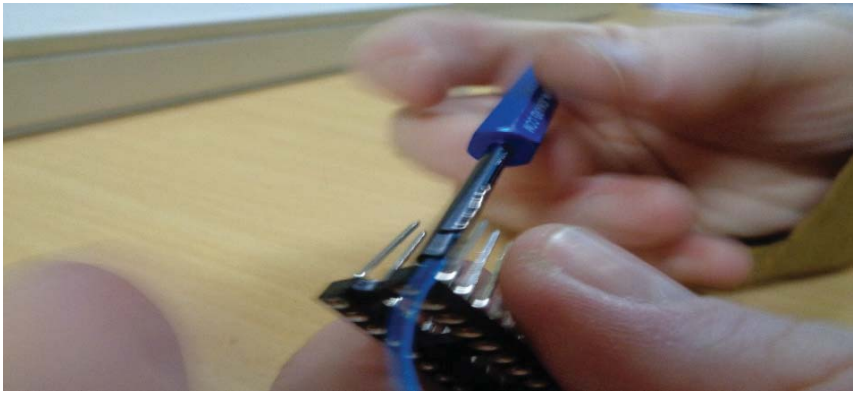


Figura 5.4. Sistema de wire wrapping en robot seguidor de luz.

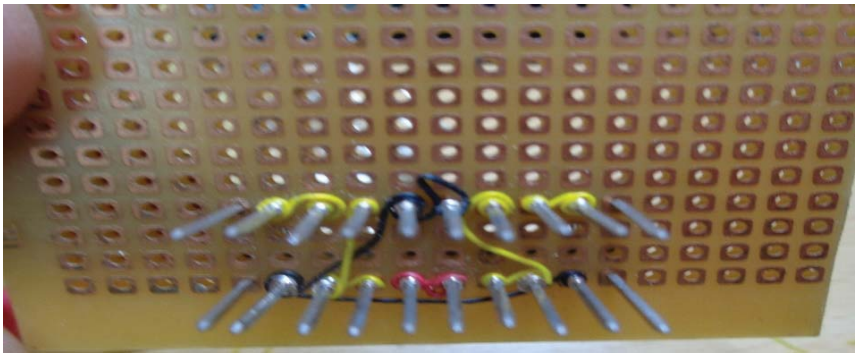


Figura 5.5. Circuito de robot seguidor de luz. Cableado con wire wrapping.

5.1.5 DISEÑO DE CIRCUITOS IMPRESOS (PCB)

Este método puede ser interesante para una etapa posterior, en la que se desarrolla el circuito en un programa de simulación de circuitos electrónicos. Luego, se genera la vista de PCB; se imprime, se lo transfiere a una placa y se lo somete a un proceso químico.

Se deberán realizar también las perforaciones para el montaje de los componentes.

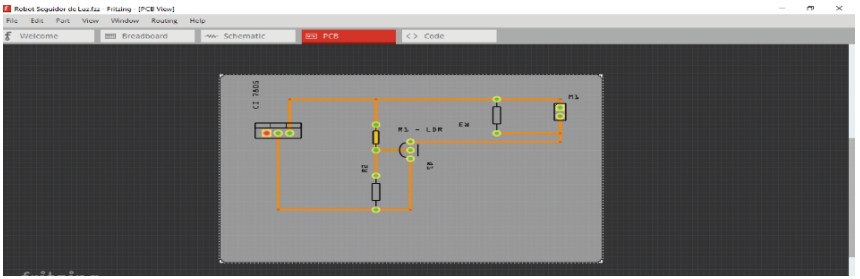


Figura 5.6. Vista de opción de conversión a PCB en Fritzing.

5.2 ROBOT SEGUIDOR DE LUZ

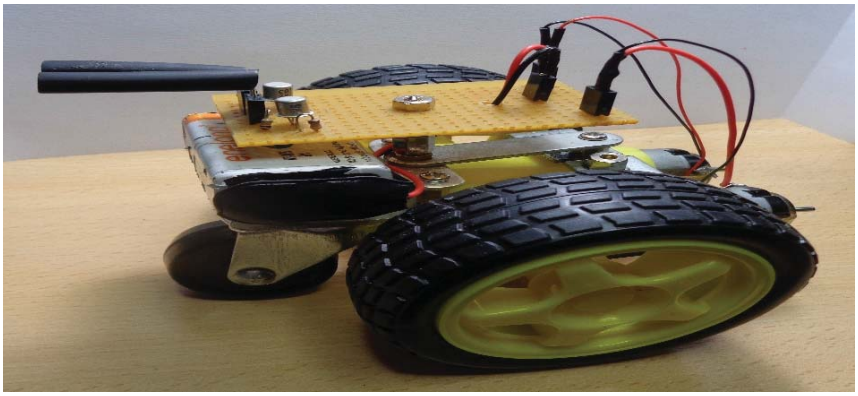


Figura 5.7. Robot seguidor de luz.

Figura 5.8. Robot seguidor de luz. Montaje.

5.2.1 INTRODUCCIÓN

REGLA DEL DIVISOR DE TENSIÓN

Aplicar la regla del divisor de tensión será fundamental para el desarrollo de robots.

La mejor forma de entender su funcionamiento se ha graficado en la imagen siguiente. Aquí se observa que, para dos resistencias iguales, la tensión se divide en el nodo en partes iguales.

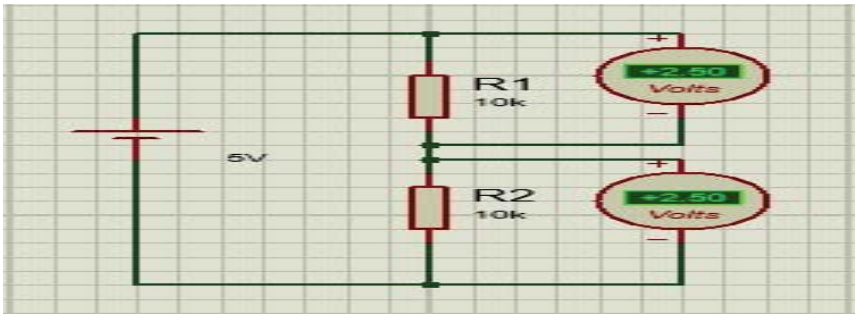


Figura 5.9. Divisor de tensión. Ejemplo.

REGLA DEL DIVISOR DE TENSION (RDV)

La regla del divisor de tensión permite determinar el valor de la tensión en cada resistencia (o en Rx), sin tener que hallar previamente la corriente.

$$V_X = R_X \,.\, V / R_T$$

Enunciado:

El voltaje en una resistencia en un circuito serie es igual al valor de esa resistencia, multiplicado por la tensión total en los elementos en serie, dividido la resistencia total de los elementos en serie.

5.2.2 CIRCUITO ELÉCTRICO

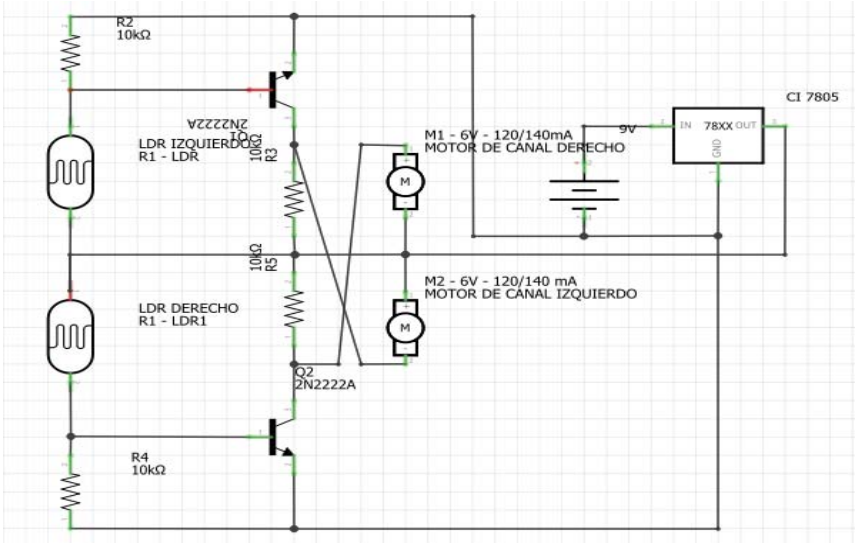


Figura 5.10. Circuito eléctrico (implementado en Fritzing). Robot seguidor de luz.

5.2.3 PRINCIPIO DE FUNCIONAMIENTO

Este robot ha sido presentado en el apartado 1.5.2 Robot seguidor de luz.

Los dos canales son simétricos. Se toma para el análisis el canal superior.

El primer dato fundamental para comprender el principio de funcionamiento del circuito es interpretar que se emplea el transistor como una llave eléctrica.

La primera etapa (entrada) está compuesta por un divisor de tensión, formado por el sensor o LDR (Resistencia variable con la luz) - denominada R1 -, y una resistencia de carbón de 10 KΩ, denominada R2.

Como se ha visto, cuando no incide luz en el LDR, este presenta una resistencia muy grande. Por lo tanto, no permite la circulación de electrones hacia la base del transistor. De este modo, la tensión a la entrada del transistor o base son los 0 V en el terminal superior de R2, que actúa como *pull down* (obsérvese que su terminal superior está conectado a 0 V).

Recuérdese que para que el transistor conmute o cambie de estado, debe tener una tensión superior a los 0.7 V en la base.

Cuando la luz incide en el LDR, su resistencia baja a un valor cercano a los 50 Ω. De esta forma, los 5 V de tensión a la salida del regulador están ahora presentes en el nodo que forman el terminal inferior de R2, el LDR y la base del transistor. Así, la tensión en la base es superior a los 0.7 V que son necesarios para conmutar; es así que el transistor cambia de estado.

Cuando el transistor conmuta, actúa como una llave entre el emisor y el colector. De esta forma, como el emisor está a un potencial de 0 V, el terminal inferior del motor queda referido a este potencial y el motor funciona, ya que posee 5 V en su terminal superior.

R3 en paralelo con el motor cumple la función de R de *pull up*, de modo de asegurar que cuando la llave entre el emisor y colector esté abierta, el motor no conduzca por ruido electromagnético.

Además, cuando el colector del transistor queda a un potencial de 0 V, R3 impide que la fuente se ponga en corto.

Cada vez que la fuente de luz es retirada del LDR, este vuelve a su estado inicial de resistencia muy grande, abriendo la llave entre emisor y colector; en consecuencia, el motor se apaga.

Observe en la Figura 5.10 que los motores están cruzados. Esto significa que el motor correspondiente al sensor del lado izquierdo, visto de frente (*view front*), está montado del lado derecho, y viceversa. Esto responde a que, en caso de incidir luz en el sensor izquierdo, el motor que está del lado izquierdo (correspondiente al sensor derecho) queda apagado, y el motor montado del lado derecho arranca produciendo un giro a la izquierda.

Si la luz incide sobre los dos LDR al mismo tiempo, arrancan los dos motores produciendo movimiento hacia adelante.

5.2.4 LISTADO DE MATERIALES O BOM

Al listado de materiales o BOM (*Bill Of Materials*) de la Figura 5.11 generado en Fritzing (Apartado 10.10), hay que adicionarle los siguientes componentes:

Ruedas para los motores y rueda loca	3
Placa experimental o multipropósito 5 x 10 cm	1
Esquineros Ángulo	2
Tornillos, arandelas, tuerca, separadores metálicos de placas	Varios

Bill of Materials: Robot Seguidor de Luz Circuito Completo.fzz

C:/Users/YIYE/Documents/Fritzing/bins/Robot Seguidor de Luz Circuito Completo.fzz
Wednesday,

Assembly List

Label	Part Type	Properties
9V	DC Power	type dc power supply; voltage 5
IC1	Voltage Regulator	chip 78005; voltage 5V; variant -to-220; package to-220-alt; part # CI 7805
M1 - 6V - 120/140mA	DC Motor	part # MOTOR DE CANAL DERECHO
M2 - 6V - 120/140 mA	DC Motor	part # MOTOR DE CANAL IZQUIERDO
Q1	NPN-Transistor	type NPN (EBC); package TO92 [THT]; part # 2N2222A
Q2	NPN-Transistor	type NPN (EBC); package TO92 [THT]; part # 2N2222A
R1 - LDR	Photocell (LDR)	resistance@ dark 300 kOhms@ 10 seconds; package THT; resistance@ luminance 16 kOhms@ 10 lux; part # LDR IZQUIERDO
R1 - LDR1	Photocell (LDR)	resistance@ dark 300 kOhms@ 10 seconds; package THT; resistance@ luminance 16 kOhms@ 10 lux; part # LDR DERECHO
R2	10kΩ Resistor	resistance 10kΩ; bands 4; pin spacing 400 mil; package THT; tolerance ±5%
R3	10kΩ Resistor	resistance 10kΩ; bands 4; pin spacing 400 mil; package THT; tolerance ±5%
R4	10kΩ Resistor	resistance 10kΩ; bands 4; pin spacing 400 mil; package THT; tolerance ±5%
R5	10kΩ Resistor	resistance 10kΩ; bands 4; pin spacing 400 mil; package THT; tolerance ±5%

Shopping List

Amount	Part Type	Properties
1	DC Power	type dc power supply; voltage 5
1	Voltage Regulator	chip 78005; voltage 5V; variant -to-220; package to-220-alt; part # CI 7805
1	DC Motor	part # MOTOR DE CANAL DERECHO
1	DC Motor	part # MOTOR DE CANAL IZQUIERDO
2	NPN-Transistor	type NPN (EBC); package TO92 [THT]; part # 2N2222A
1	Photocell (LDR)	resistance@ dark 300 kOhms@ 10 seconds; package THT; resistance@ luminance 16 kOhms@ 10 lux; part # LDR IZQUIERDO
1	Photocell (LDR)	resistance@ dark 300 kOhms@ 10 seconds; package THT; resistance@ luminance 16 kOhms@ 10 lux; part # LDR DERECHO
4	10kΩ Resistor	resistance 10kΩ; bands 4; pin spacing 400 mil; package THT; tolerance ±5%

Exported with Fritzing 0.9.3- http://fritzing.org

Figura 5.11. BOM o listado de materiales para cada lado o canal del robot.

Obsérvese en la Figura 5.43 el BOM o listado de materiales ampliado.

5.2.5 MONTAJE Y PASOS DE IMPLEMENTACIÓN

Se puede generar un chasis que le sirva como base para los circuitos de robots a implementar. De esta forma, no hay que disponer de una estructura diferente para cada dispositivo, salvo las adaptaciones para los correspondientes sensores.

Figura 5.12. Opción de chasis para robot seguidor de luz.

Figura 5.13. Inicio de montaje de chasis.

En la siguiente imagen se visualiza el inicio de armado de una estructura para unir la rueda loca y los motores a la placa del circuito.

Figura 5.14. Rueda loca con separadores metálicos.

Figura 5.15. Implementación de chasis.

Una vez implementado el chasis, se debe proceder a la fase de montaje de los componentes. El método propuesto es el montaje cableado.

Este método de montaje cableado consiste en unir los componentes mediante un trozo de cable. Si la distancia entre los componentes es muy pequeña, una soldadura de estaño. No se recomienda realizar pistas de estaño largas, por cuestiones de ruido electromagnético, o fallas y riesgos eléctricos.

Para el soldado y la unión eléctrica de los componentes (pistas), se puede emplear cable de *wire wrapping*, cable 0.125, 0.15 mm. Recuérdese no emplear cables de excesivo diámetro, ya que dificultan la soldadura en espacios que requieren de precisión.

Una vez calculado el largo de cable a emplear, se puede estañar la punta de este o, si fuera conveniente, limar las patas del componente a soldar con la lima de matricero, de forma de lograr mayor rozamiento o adherencia.

En la fase de montaje de componentes, se debe seguir el circuito eléctrico. Pero primero se debe diagramar la mejor disposición de los componentes en función del espacio disponible en la placa, una vez que se haya establecido el área que ocuparán en la misma, el chasis, los motores y la rueda loca. **El montaje del circuito debe implementarse de adelante hacia atrás**.

Para los sensores LDR, una buena práctica es diseñar un zócalo para cada uno, empleando alojamientos hembra de dos pines, tal cual se observa en la Figura 5.16.

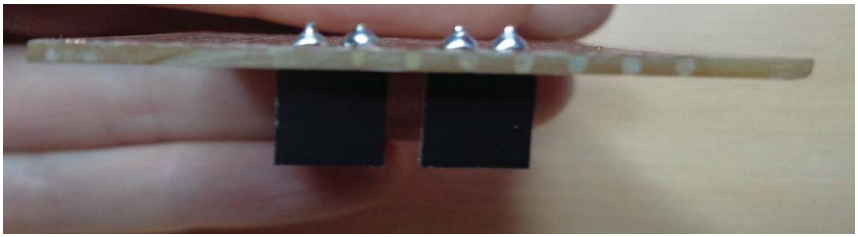


Figura 5.16. Fase de montaje. Zócalos para sensores LDR.

El empleo de los zócalos permitirá poner y retirar los sensores LDR, a fin de recambiarlos o emplearlos para otra actividad.

Obsérvese en la figura siguiente que cada zócalo para LDR y cada transistor corresponden a un canal o lado del robot.

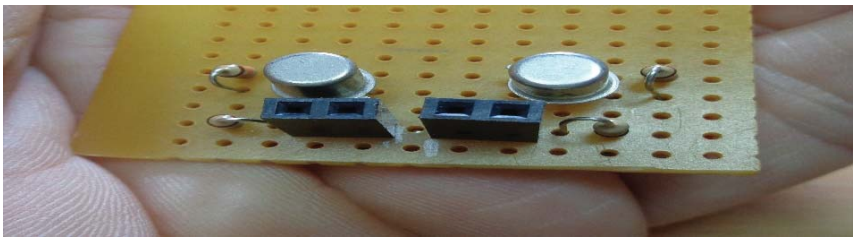


Figura 5.17. Montaje de etapa de entrada y transistores.

5.2.6 TESTEO Y PRUEBAS DE FUNCIONAMIENTO

Aunque pueda sorprender al principio, el circuito de la Figura 5.17 es el requerido para la implementación del robot seguidor de luz. Ha sido desarrollado por el autor, para simplificar el acceso a esta disciplina.

Una buena práctica para garantizar que el funcionamiento final del robot sea correcto es testear las diferentes etapas del circuito y tensiones, a medida que se avanza en su implementación. Asimismo, lo es analizar las conexiones y medir continuidad con el tester, para comprobar que no haya conexiones o pistas en cortocircuito.

Del mismo modo que ocurre en la programación, el diseño modular de las etapas del robot es una buena práctica para simplificar el montaje e intercambio de piezas en nuestros dispositivos.

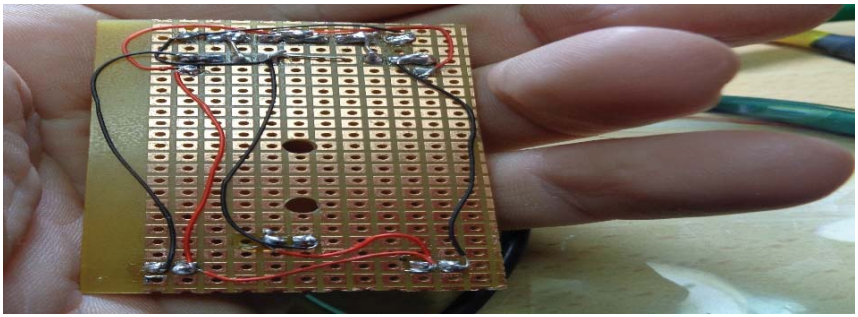


Figura 5.18. Fase final de prueba de la placa del robot seguidor de luz.

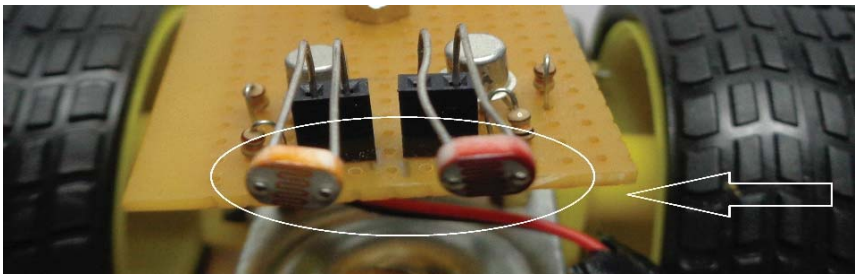


Figura 5.19. Montaje de LDR para testeo de circuito.

En la Figura 5.20 se presenta otra vista de la implementación final del robot seguidor de luz.

Al igual que para el proyecto siguiente (Robot seguidor de línea blanca), el regulador de tensión ha sido montado en la parte inferior de la placa.

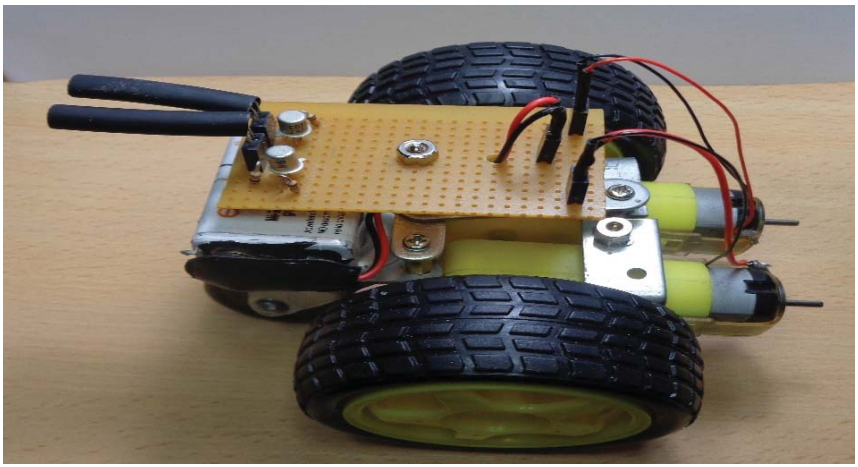


Figura 5.20. Robot seguidor de luz. Montaje final.

5.3 ROBOT SEGUIDOR DE LÍNEA BLANCA

En este ítem se desarrolla la implementación del robot seguidor de línea blanca, que emplea sensores CNY70, y transistores en configuración Darlington.

Este robot ha sido presentado en el apartado 1.5.3 Robot seguidor de línea blanca.

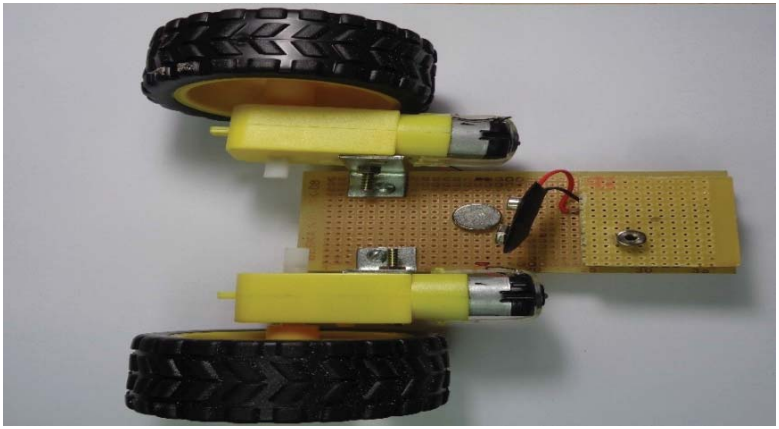


Figura 5.21. Robot seguidor de línea blanca.

5.3.1 CIRCUITO ELÉCTRICO

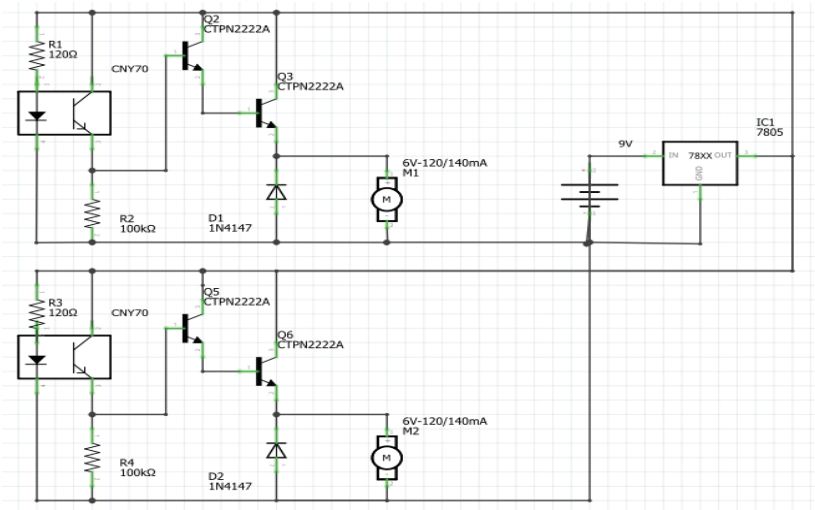


Figura 5.22. Circuito eléctrico. Robot seguidor de línea blanca.
El circuito eléctrico ha sido diseñado empleando Fritzing.

5.3.2 PRINCIPIO DE FUNCIONAMIENTO

En la etapa de entrada del sistema, se emplea el sensor CNY70. Recuérdese este es un sensor óptico reflectivo u optoacoplador, es decir, acopla por luz un diodo infrarrojo (IR), con la base de un fototransistor. Los fototransistores y el sensor han sido presentados en el Capítulo 3.

Obsérvese en el circuito que el ánodo (+) del diodo IR está conectado en la rama de la resistencia de 120 Ω, al polo positivo de la fuente de tensión (5 VCC desde la salida del regulador 7805). Luego, el diodo IR emite luz; y si encuentra una superficie clara, el haz de luz producido rebota y excita la base del fototransistor.

El fototransistor tiene el emisor conectado a una resistencia de *pull down* fuerte de 100 KΩ. El colector está conectado al positivo de la fuente o 5 VCC.

La entrada a la base de los transistores en configuración Darlington está conectada ahora al nodo (punto de intersección) entre el emisor y el colector.

Mientras se cumpla la condición de que la luz reflecta en una superficie clara, la base del fototransistor permitirá la conmutación del estado bajo de su emisor, al potencial de 5 VCC presente en el colector, valor suficiente para romper la barrera de potencial de 0,7 V en el diodo BE (base-emisor) del primer transistor de la configuración Darlington (Q2 o Q5 en el circuito). De esta forma, el segundo transistor (Q3 o Q6 en el circuito) conmutará también comportándose, nuevamente, como una llave eléctrica, entregando 5 VCC al extremo superior del motor. Como el motor tiene su otro extremo o terminal conectado al terminal negativo de la fuente, se enciende y conduce.

En paralelo con el motor, se conecta un diodo llamado *clamping*, empleado cuando se trabaja con carga inductiva, como la de motores, para proteger al transistor.

Cuando el diodo IR emite luz y esta encuentra una superficie de color negro, la luz no se propaga. Por lo tanto, la base del fototransistor queda a un potencial eléctrico bajo; no conmuta, y el primer transistor (Q2 o Q5 en el circuito) de la configuración Darlington tendrá un valor de 0 V en su base; no conmutará y dejará, de esta forma, al segundo transistor (Q3 o Q6 en el circuito) en corte o como llave abierta. Es así que el motor estará en posición de apagado.

5.3.3 LISTADO DE MATERIALES O BOM

El listado de materiales para la construcción del robot seguidor de línea blanca es el siguiente:

COMPONENTE	CANTIDAD	DESCRIPCIÓN
Placa	1	5 x 10 cm. Experimental o multipropósito. Pertinax o Epoxi
Batería de 9V	1	Recargable. 250 mAh. Ni.Mh. Estándar
Conector de Batería	1	Conector Broche
Imán de Neodimio	1	Opciones vistas en el Capítulo 3
Regulador de V	1	7805 - Regulador de tensión Positiva
Disipador de calor	1	Para TO220. Con tornillo y tuerca
Motores	2	6V - 120/140 mA – con reductor
Ruedas	2	Diámetro de 65 mm. con encastre para motor
Sensores	2	CNY70
Alojamientos Hembra	4	Por 2 pines. Se emplean para zócalos de sensores y conexión de motores
Transistores	4	NPN. CTPN2222A
Resistencias	2	120 Ω
Resistencias	2	100 KΩ.
Diodos	2	Rectificadores. 1N4147 o 1N4001/7
Esquineros Ángulo	2	25 x 25 mm
Tornillos (sujeción de motor a ángulo)	2	31 mm de largo x 3 mm diámetro. Se requieren también tuercas y arandelas
Tornillos (sujeción de ángulo a la placa)	2	15 mm de largo x 4 mm diámetro. Se requieren también tuercas y arandelas
Separador de placa	1	Metálico de 16 mm con tuerca
Separador de placa	1	Metálico de 11 mm con tuerca
Separador de placa	1	Acrílico de 10 mm hembra - hembra

5.3.4 MONTAJE Y PASOS DE IMPLEMENTACIÓN

Los pasos propuestos para la implementación del robot seguidor de línea blanca son similares a los descritos para el robot seguidor de luz.

Recuérdese:

- Analizar la disposición de motores, ángulos, batería y componentes.
- Marcar con lápiz espacios a ocupar por motores, ángulos y batería.
- Realizar el montaje de los componentes de adelante hacia atrás.
- Calcular el largo de cable necesario para la unión de los componentes.
- Salvo distancias muy cortas, no emplear pistas de cobre.
- Medir continuidad o analizar el montaje a medida que avanza.

Importante:

- No realizar soldaduras o manipulación de componentes con el circuito energizado.
- No someter los componentes a temperaturas de soldado excesivas o prolongadas.
- Emplear siempre protección ocular.

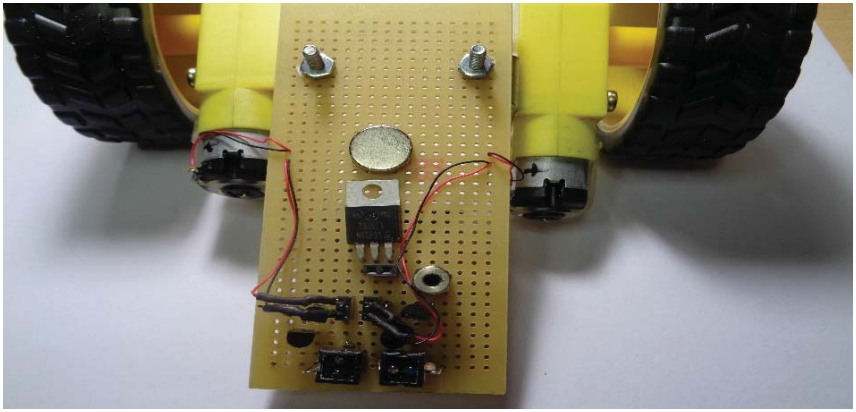


Figura 5.23. Montaje final del robot seguidor de línea blanca.

Obsérvese en la Figura 5.23 que el robot ha sido diseñado de forma tal de que los componentes quedan hacia abajo, con el fin de asegurar que los sensores estén al ras del piso.

5.3.5 TESTEO Y PRUEBAS DE FUNCIONAMIENTO

Se recomienda realizar el testeo de funcionamiento de cada etapa del circuito a medida que se realiza el montaje.

Recuérdese verificar las polaridades de la batería, motores, y componentes antes de energizar el circuito.

Es importante realizar las pruebas de los robots en condiciones o ámbitos similares a aquellos para los que han sido diseñados.

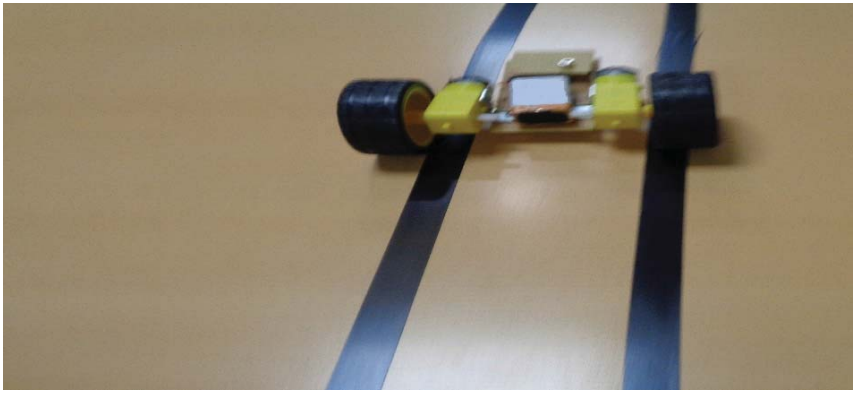


Figura 5.24. Prueba final de robot seguidor de línea blanca.

Obsérvese en la figura anterior que el robot circula sobre un carril de color claro. Esto permite que el haz de luz emitido por el diodo IR sea reflectado e inicie el proceso descrito para su funcionamiento. Las líneas negras laterales de la pista delimitan su camino.

5.4 OTRA FORMA DE MONTAJE DE ROBOTS

5.4.1 INTRODUCCIÓN

Se aplicará ahora como ejemplo otra forma de montaje al robot seguidor de luz descrito en el apartado 5.2. Esta vez se utiliza una sola placa multipropósito o experimental para el montaje de motores, rueda loca, batería y la electrónica del circuito. Posteriormente, se agregan también dos diodos LED azules para la indicación de encendido, y un gabinete para la batería.

5.4.2 MONTAJE Y PASOS DE IMPLEMENTACIÓN

Se aplican los pasos descritos en el apartado 5.3.4.

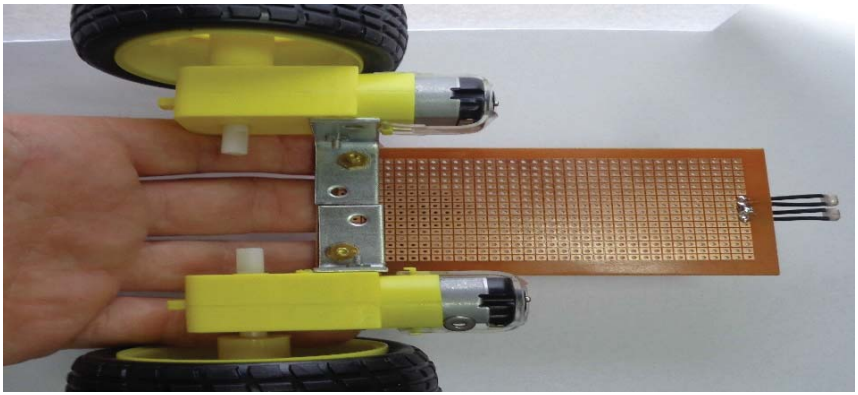


Figura 5.25. Otra forma de montaje para el robot seguidor de luz.

Obsérvese que en la Figura 5.25 se emplea un método de montaje similar al empleado en el apartado 5.3 Robot seguidor de línea blanca. Se montan los motores directamente a la placa, con el empleo de esquineros ángulos.

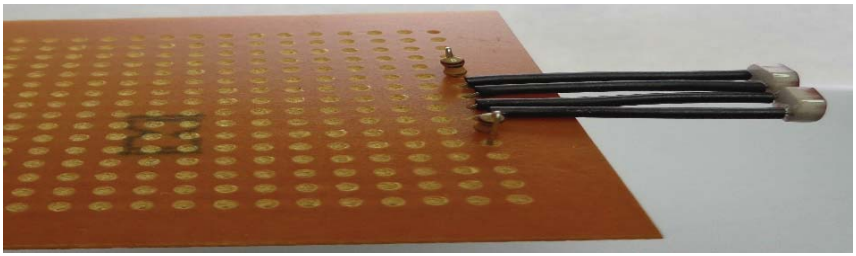


Figura 5.26. Inicio de montaje de los divisores de tensión.

Recuérdese del apartado 5.2.2 Circuito eléctrico del robot seguidor de luz, que la etapa de entrada para cada lado o canal está compuesta por un divisor de tensión entre un LDR y una resistencia de *pull down*.

Puede observarse también en la Figura 5.26 que en este caso no se emplean los alojamientos hembra para el montaje de los LDR.

Se emplea termocontraible en los LDR para evitar cortocircuitos.

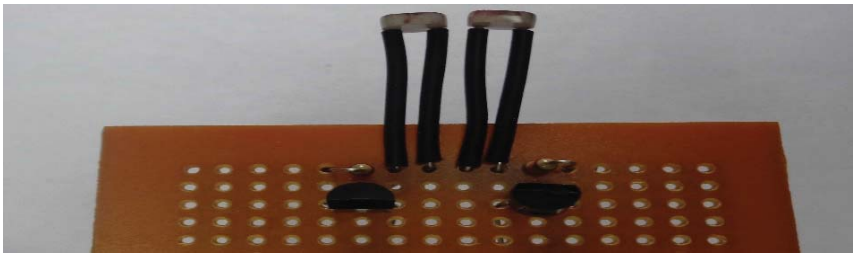


Figura 5.27. Montaje de los transistores CTPN2222A encapsulado TO92.

Recuérdese de la Figura 5.27 que los transistores han sido montados de forma tal que su emisor (pata izquierda visto de frente), coincida con el terminal de las resistencias de 10 KΩ (*pull down*). Estas van conectadas a un potencial de 0 V o GND.

Este montaje facilita también que el otro terminal de las resistencias, que está en el mismo nodo que una de las patas del LDR, quede de forma recta al terminal central del transistor o base. Al igual que en el robot seguidor de línea blanca, se ha empleado el transistor CTPN2222A, de menor coste que el transistor 2N2222A, y con características similares para este proyecto.

Los terminales de los LDR que se observan en posición central pueden ser también unidos entre sí, ya que van conectados a un potencial de 5 V o positivo de la fuente (salida del regulador de 5 V).

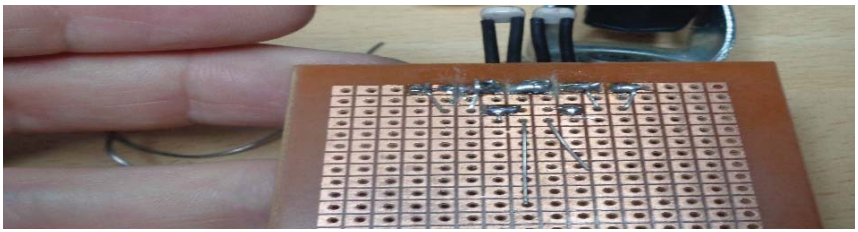


Figura 5.28. Conexión entre componentes.

Obsérvese en la Figura 5.28 que pueden ser empleados los terminales de los componentes para la conexión entre sí. El propósito es simplificar el montaje y la implementación del circuito.

Figura 5.29. Cableado correspondiente al negativo o GND del circuito.

Recuérdese, como detalle importante, para evitar errores, que cuando se trabaja en el montaje de los componentes, al dar vuelta la placa para el soldado o su fijación, el componente que veía del lado izquierdo queda ahora en la posición de lado derecho. Esto se observa en la Figura 5.29 para el cableado del negativo o GND.

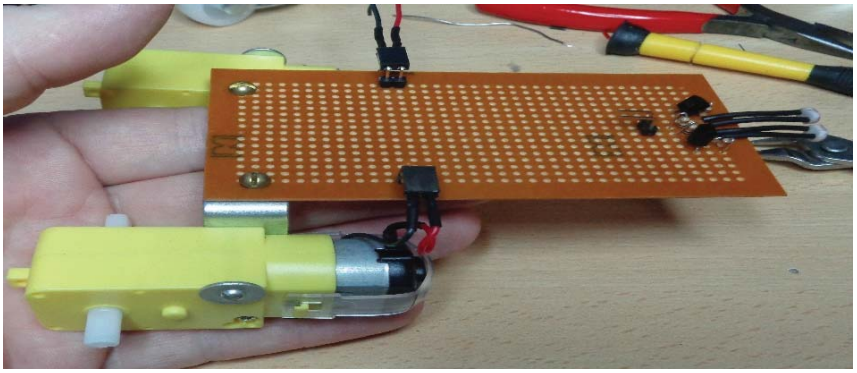


Figura 5.30. Montaje de pines rectos a 90º para conexión de motores.

Al igual que en los circuitos anteriores, se debe emplear un regulador de tensión para adaptar los 9 V de tensión de la batería a los 6 V requeridos por los motores.

Se emplea el regulador LM7805 para obtener un valor seguro en la VBE (tensión base emisor) máxima del transistor.

De otra forma, podría ser empleado el regulador LM7806 para obtener mayor rendimiento en los motores.

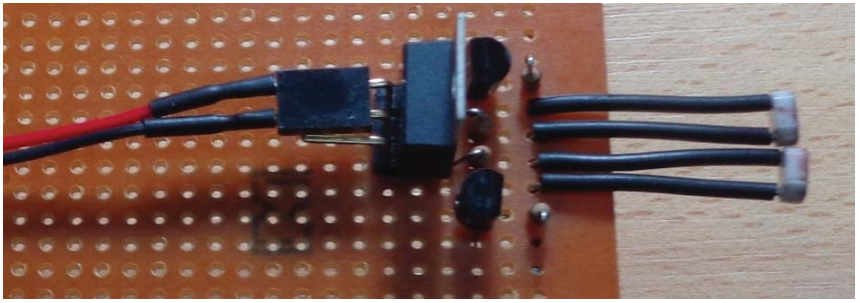


Figura 5.31. Montaje del regulador de tensión a la placa.

Obsérvese, en la Figura 5.31, el montaje del regulador de tensión 7805. El regulador ha sido dispuesto en un punto estratégico en la placa: entre las resistencias de 10 KΩ (*pull up*) que van en paralelo con el colector del transistor.

Recuérdese del *pin out* del CI regulador que, visto de frente (*view front*), el terminal o pata de la izquierda corresponde a la entrada de tensión. El terminal central corresponde al negativo. Y el terminal de la derecha corresponde a la salida o 5 V.

Se ha dejado libre un pin conectado al terminal de salida de 5 V, para facilitar cualquier instancia de testeo, futura ampliación o modificación del circuito.

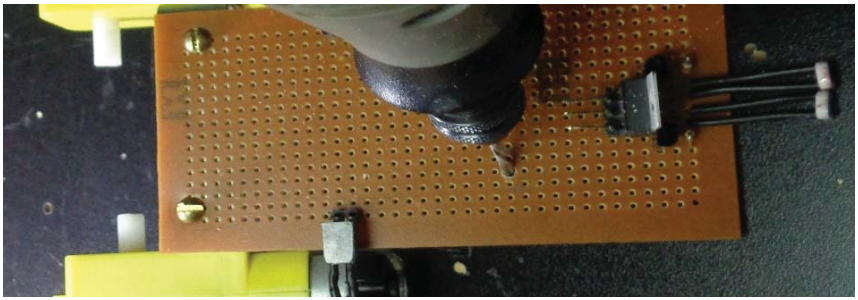


Figura 5.32. Montaje de la rueda loca.

Las opciones de rueda loca fueron tratadas en el Capítulo 3.

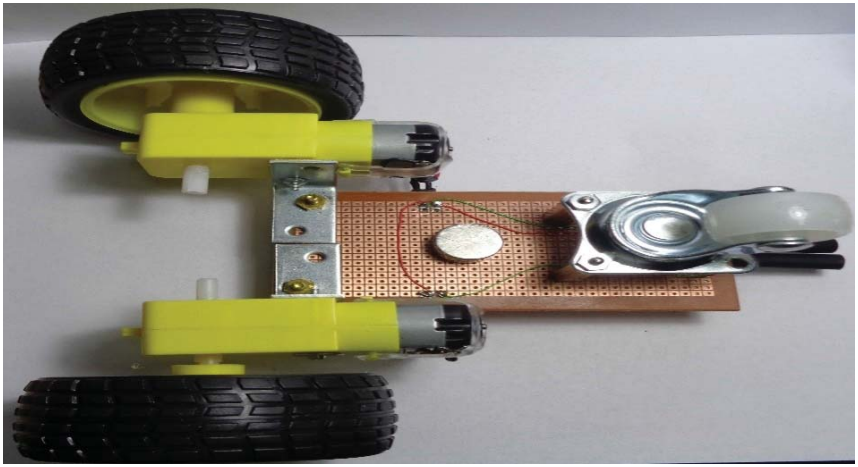


Figura 5.33. Montaje. Vista de la parte inferior.

Obsérvese en la Figura 5.33 la prolijidad del empleo de este método.

El imán de neodimio en la parte central de la placa fue presentado en Capítulo 3. Su función es la de sujetar la batería.

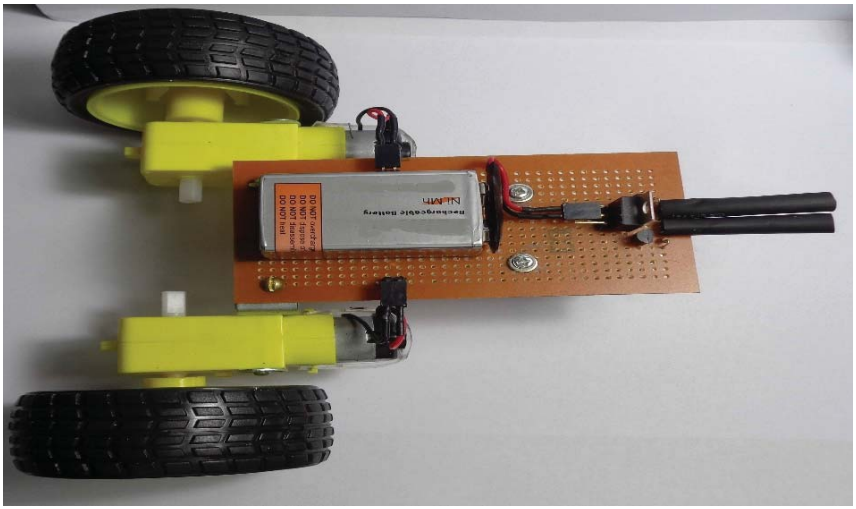


Figura 5.34. Vista final de implementación.

Figura 5.35. Vista lateral de la implementación final.

5.4.3 OTRAS MODIFICACIONES

Otra posibilidad de mejora o modificación del circuito de robot seguidor de luz consiste en montar un gabinete para la batería, desarrollada con el empleo de la impresión 3D, (que será presentado en el Capítulo 11, Figura 11.35).

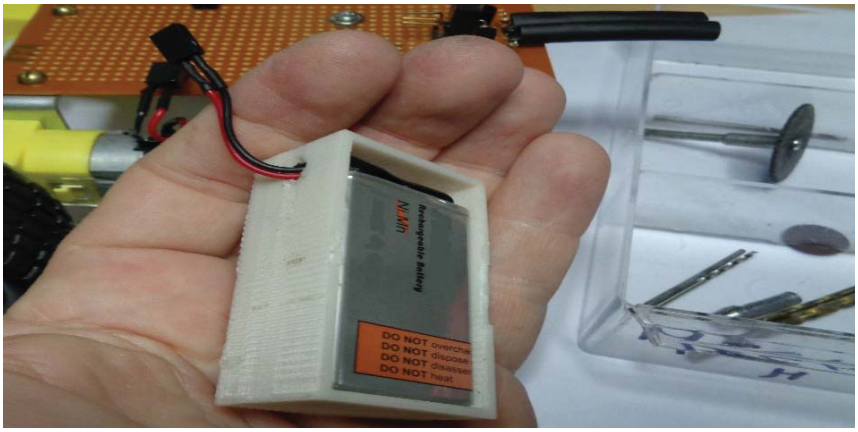


Figura 5.36. Gabinete para batería modelado con impresión 3D.

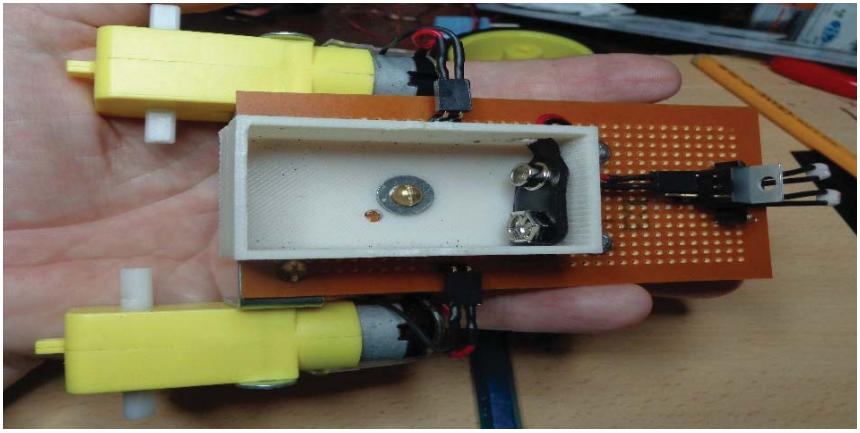


Figura 5.37. Montaje de gabinete para batería.

Una vez comprendido el funcionamiento del circuito y el montaje y soldadura de los componentes, es posible ampliar y mejorar el circuito.

Como cierre del capítulo se montará un diodo LED azul en cada lado o canal del robot, a fin de indicar que el robot está prendido.

Se empleará para la polarización de los diodos LED, el circuito presentado en el Capítulo 2, Figura 2.26. Requiere para una tensión de 5 V una resistencia de 120 Ω en serie con cada diodo LED testigo.

Recuérdese del Capítulo II que cada diodo LED en serie con la resistencia va conectado en paralelo con la salida del regulador de tensión de 5 V, ya que dos elementos o más conectados en paralelo tienen la misma tensión.

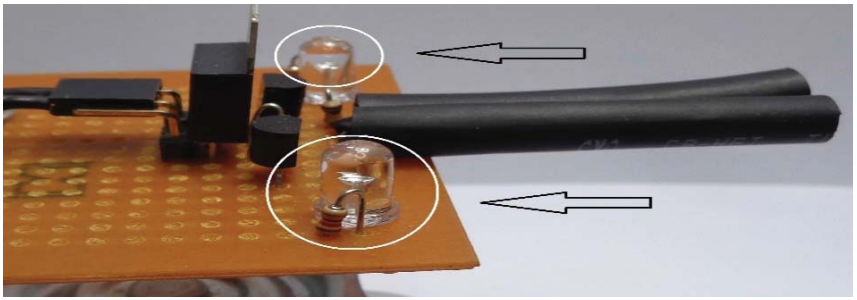


Figura 5.38. Diodos LED azules como diodos testigo de encendido.

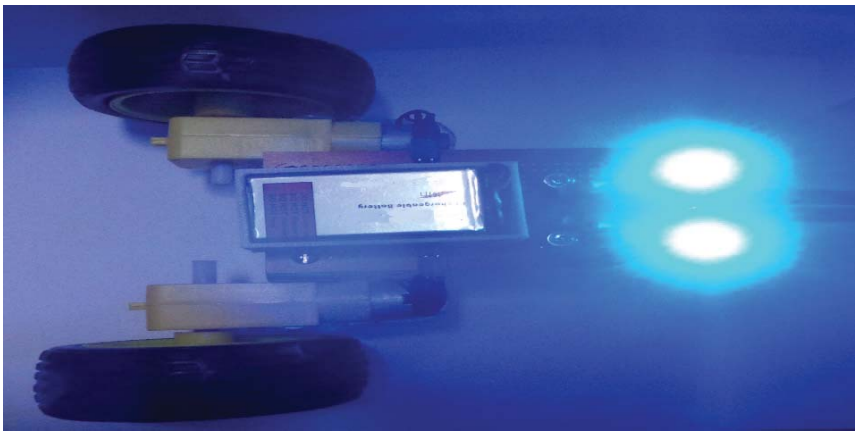


Figura 5.39. Montaje final del robot seguidor de luz con diodos LED testigos.

Figura 5.40. Robot seguidor de luz en funcionamiento.

Las modificaciones al circuito original de la Figura 5.10 son las siguientes:

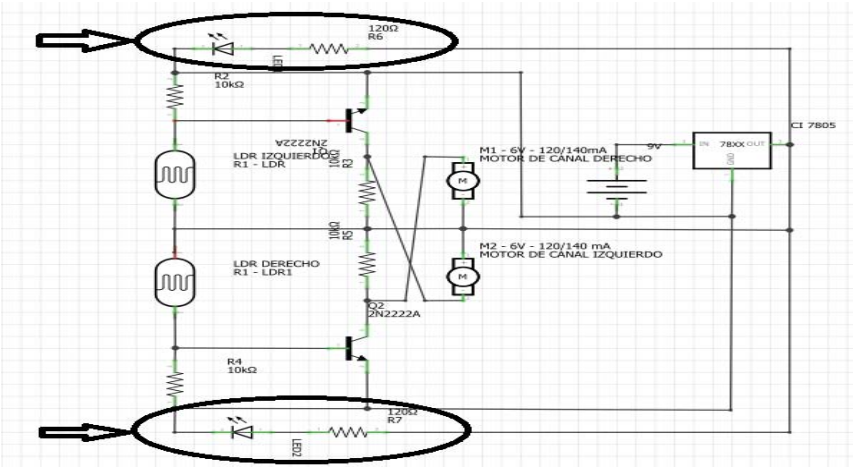


Figura 5.41. Circuito eléctrico final del robot seguidor de luz.

Figura 5.42. Robots seguidores de luz implementados en la Escuela de Maestros.

5.4.4 LISTADO DE MATERIALES O BOM. VISTA AMPLIADA

El BOM o listado de materiales de la Figura 5.11 (Robot seguidor de luz) en vista ampliada, es el siguiente:

COMPONENTE	CANTIDAD	DESCRIPCIÓN
Placa	1	5 x 10 cm. Experimental o multipropósito. Pertinax o Epoxi
Batería de 9V	1	Recargable. 250 mAh. Ni.Mh. Estándar
Conector de Batería	1	Conector Broche
Imán de Neodimio	1	Opciones vistas en el Capítulo 3
Regulador de V	1	7805 - Regulador de tensión Positiva
Disipador de calor	1	Para TO220. Con tornillo y tuerca
Motores	2	6 V - 120/140 mA – con reductor
Ruedas	2	Diámetro de 65 mm. con encastre para motor.
Sensores.	2	LDR
Termo contraíbles	2	Trozos de 4 cm para protección de LDR
Alojamientos Hembra	5	Por 2 pines. Se emplean para zócalos de sensores. Conexión de motores y batería
Transistores	2	NPN. 2N2222A
Resistencias	4	10 KΩ
Esquinero Ángulo	2	25 x 25 mm
Tornillos (sujeción de motor a ángulo)	2	31 mm de largo x 3 mm diámetro. Se requieren también tuercas y arandelas
Separadores de placa	4	Metálicos de 10 mm con tuercas y tornillos
Varillas de aluminio	2	Montaje de chasis (aplica al primer modelo)

Figura 5.43. Robot seguidor de luz. Listado de materiales ampliado.

Para realizar las modificaciones introducidas al circuito (Figura 5.41), deberán ser añadidos los siguientes componentes:

- 2 diodos LED azules de alto brillo.
- 2 resistencias de 120 Ω.

CAPÍTULO 6
INTRODUCCIÓN A LA PROGRAMACIÓN EN ROBÓTICA

6.1 INTRODUCCIÓN

"Quien no se traza una meta, jamás sabrá si llegó"

Séneca.

En el capítulo anterior, se presentó la línea de robots denominados sin lógica adicional. Estos no emplean lógica de CI o programable. A partir de aquí, la programación ocupa un espacio clave, basado en la necesidad de aplicación, para los capítulos siguientes.

La programación está cada vez más presente en la tecnología y, por ende, en lo cotidiano. Para lograr conectividad entre dispositivos, se requiere programación de base, por lo que entender sus principios, permite ser partícipes activos dentro de esta nueva gran revolución.

A continuación, se propone abordar el desarrollo y la formación de estos saberes, aplicando los mismos a situaciones prácticas en el ámbito de la robótica.

6.2 ALGORITMO

La potencialidad de los dispositivos electrónicos analógicos depende de la capacidad para ordenarlos en una secuencia, y forma lógica (circuito).

Este principio se profundiza en los sistemas digitales, donde si el dispositivo que se pretende dirigir o controlar (programar), no recibe las instrucciones correctas, puede que permanezca inclusive en estado OFF (apagado) todo el tiempo. Otras veces, como establecen los reglamentos para las competencias en las diferentes ligas, los robots deben poseer una demora (*delay*) inicial de cinco segundos, antes de poder realizar cualquier acción o movimiento. Esto refiere a la necesidad del desarrollo de un algoritmo confiable para cada aplicación.

6.2.1 DEFINICIONES

Un algoritmo es una serie de pasos, acciones o procedimientos, que permiten alcanzar un resultado o resolver un problema.

Aplicada la definición, al tiempo de espera requerido en las competencias de robótica. La primera instrucción, dentro de la secuencia de pasos (instrucciones), debe ser invocar a la función *delay*.

PROBLEMA

En general, cuando se piensa en la palabra problema, se la asocia a cosas complicadas o graves. Sin embargo, al abordar su etimología, vemos que esta representa una diferencia de estados, es decir:

- Necesidad de transformar un estado inicial de cosas, en otro final.
- Situación en la que las cosas que tenemos son diferentes de las que deseamos.

Ejemplo: el robot adversario está dispuesto en el tatami, y al robot de su equipo se le averió la pala. La situación constituye un problema, es decir, no funciona la pala, y necesitaría de ella para tomar distancia, empujar e impactar. **Qué problema, ... desearía que funcione.**

6.2.2 TRES ETAPAS DE DISEÑO

Imagínese también, para el ejemplo anterior, el entorno bajo el cual se debe resolver de forma rápida el problema. Esto lleva a analizar, que, a fin de simplificar la resolución de un problema, se puede dividir el desarrollo del algoritmo en las tres etapas siguientes:

ANÁLISIS Y COMPRENSIÓN DEL PROBLEMA
DESARROLLO DEL ALGORITMO
COMPROBACIÓN DEL ALGORITMO

Figura 6.1. Etapas para el diseño de un algoritmo.

6.2.3 TRES CARACTERÍSTICAS

Los algoritmos deben reunir también las siguientes tres características:

PRECISIÓN
DETERMINISMO
FINITUD

Figura 6.2. Características de un algoritmo.

PRECISIÓN: los pasos a seguir en el algoritmo deben ser definidos claramente.

DETERMINISMO: para los mismos datos de entrada, el algoritmo debe dar siempre los mismos resultados.

FINITUD: no puede existir un algoritmo de longitud infinita.

Ejemplos:

Precisión: se debe generar un *delay* de 5 segundos iniciales, y en vez de definirlo con precisión, se invoca a la función *delays*, que elige de forma aleatoria, una demora entre diez existentes en una tabla.

Resultado: existen 9 posibilidades contra 1 de que el algoritmo sea erróneo. Y aunque aleatoriamente coincida con el tiempo requerido, el desarrollo en función del requerimiento no es preciso.

Determinismo: el sensor ultrasónico derecho de un robot devuelve al pin del microcontrolador en que está conectado, la señal correspondiente a la detección de un obstáculo. De este modo el motor izquierdo es detenido, de forma tal de girar a la izquierda. A pocos decímetros, se repite el mismo proceso de detección, pero, esta vez, se aceleran los dos motores: el lado derecho colisiona y el sensor es dañado.

Resultado: para los mismos datos de entrada, la ejecución del algoritmo dio un resultado diferente.

Finitud: los microcontroladores empleados para el control de los robots, veremos en el Capítulo 8, tienen limitación en su memoria de programa. Por tanto, no es posible concebir la existencia de un algoritmo infinito.

6.2.4 ETAPAS DE UN ALGORITMO

Un algoritmo se divide en las siguientes tres etapas:

INTRODUCCIÓN DE DATOS
PROCESAMIENTO DE DATOS
SALIDA DE RESULTADOS

Figura 6.3. Etapas de un algoritmo.

En la Figura 8.1 se puede ver, por ejemplo, la similitud con estas etapas.

6.3 LENGUAJES DE PROGRAMACIÓN

Al igual que ocurre con los diferentes idiomas, los lenguajes de programación hacen referencia a un conjunto de reglas, que permiten la comunicación. Aplicado a los sistemas digitales, el objetivo es comunicarse con el procesador, sea de un ordenador, un móvil, un microcontrolador, o FPGA.

6.3.1 CLASIFICACIÓN

En función de su cercanía con el Procesador, o CPU, se pueden clasificar los lenguajes de programación, en las siguientes categorías:

MÁQUINA
BAJO NIVEL O ENSAMBLADOR
ALTO NIVEL

Figura 6.4. Clasificación de los lenguajes de programación.

Se desarrollarán ahora las características de cada una de estas tres categorías, tomando como referencia que el empleo del lenguaje máquina, por ejemplo, excede la actividad cotidiana de los programadores de aplicaciones Web y sistemas. El lenguaje máquina, es obtenido en instancia de compilación. El lenguaje Assembler es todavía empleado para la programación de Microcontroladores, y es la tendencia migrar al lenguaje de alto nivel C.

6.3.2 LENGUAJE DE MÁQUINA

El lenguaje o código máquina es aquel en que las instrucciones empleadas son directamente interpretadas por el procesador (microprocesador, microcontrolador), por lo que emplea secuencias de 0's y 1's.

Este lenguaje es específico para cada arquitectura o CPU.

6.3.3 LENGUAJE DE BAJO NIVEL

El lenguaje Assembler o ensamblador es un referente para este nivel.

Produjo como beneficio, ante una industria creciente y demandante de programadores, ofrecerles una interfaz más agradable, ya que emplea mnemónicos, y facilita de este modo la memorización de las instrucciones.

Los mnemónicos son abreviaciones de palabras del idioma inglés que referencian la acción a realizar.

Ejemplo:

Se necesita conmutar un transistor para arrancar el motor izquierdo del robot detector de línea. El transistor está conectado al pin RA0 del microcontrolador. La instrucción a emplear, para poner en estado alto o setear el mismo, es:

```
movlw b'00000001'
```

El mnemónico o instrucción `movlw` significa copiar (del inglés move = mover) literalmente en w (w es un registro), el valor binario 00000001, necesario para conmutar el transistor.

6.3.4 LENGUAJES DE ALTO NIVEL

Los lenguajes de programación de alto nivel emplean para sus instrucciones, palabras del idioma inglés; así su aplicación resulta más natural para las personas.

A su vez, son independientes de la CPU empleada, otorgando portabilidad, característica fundamental en la ingeniería del software.

El lenguaje de máquina es interpretado de forma directa por la CPU o procesador. El ensamblador es cercano a la misma; y el lenguaje de alto nivel, a las personas. Por lo que, una vez desarrollado el programa, se debe realizar la siguiente secuencia de pasos:

ACCIÓN	PROGRAMA REQUERIDO	ARCHIVO GENERADO
Compilar	Compilador	`.obj`

Figura 6.5. Etapas para la generación de código máquina desde alto nivel.

Ejemplos de lenguajes de alto nivel, relacionados a las aplicaciones de robótica:

LENGUAJE	APLICACIÓN
C	Microcontroladores, ARDUINO, Processing, Java.
C++	ARDUINO, Processing, Java.
Java	ARDUINO, Processing, NetBeans (en IDE MPLAB X).

Figura 6.6. Lenguajes de alto nivel y aplicaciones.

6.4 COMPILADOR, ENLAZADOR E INTÉRPRETES

6.4.1 COMPILADOR

Un compilador es un programa que forma parte de un IDE. Los compiladores son los encargados de **traducir** el código fuente, realizado en alto nivel o ensamblador, al lenguaje propio del procesador.

Al compilar, interviene también un programa llamado ENLAZADOR o LINKER, cuya función es unir el código objeto generado, con la información de las librerías incluidas en el programa fuente.

6.4.2 INTÉRPRETE

Un intérprete es también un programa, que, a diferencia del compilador, traduce y ejecuta las líneas una por una.

Los intérpretes no generan código objeto de salida. Entonces el mismo archivo fuente, puede generar resultados iguales en distintos sistemas o plataformas.

Como desventaja, son más lentos, por su estructura de trabajo.

6.5 PARADIGMAS DE PROGRAMACIÓN

Un paradigma es un modelo. En informática, define un conjunto de reglas y características para el diseño y desarrollo eficiente de programas.

6.5.1 CLASIFICACIÓN

Existen muchos paradigmas de programación, siendo el más empleado para nuestras categorías de robots a implementar, el paradigma de la Programación Estructurada o PE.

PARADIGMA
PROGRAMACIÓN ESTRUCTURADA

Figura 6.7. Paradigma de programación más empleado en robótica.

6.5.2 PROGRAMACIÓN ESTRUCTURADA (PE)

En la programación estructurada, se emplean las siguientes reglas para escribir un programa:

El programa tiene diseño modular
Los módulos son diseñados de modo descendente
Cada módulo, se codifica utilizando las tres estructuras de control básicas

Figura 6.8. PE – Reglas para escribir un programa.

ESTRUCTURAS DE CONTROL BÁSICAS
SECUENCIA
SELECCIÓN
REPETICIÓN

Figura 6.9. Tres estructuras de control básicas.

Entre sus características destacan también:

La programación estructurada evita el uso de la instrucción `goto`.

`Goto`, significa **ir a**, y es una instrucción de salto.

La programación estructurada es el conjunto de técnicas que incorporan:

PE – TÉCNICAS INCORPORADAS
RECURSOS ABSTRACTOS
DISEÑO DESCENDENTE (TOP-DOWN)
ESTRUCTURAS BÁSICAS

Figura 6.10. PE – Técnicas incorporadas.

Estas técnicas tienen como finalidad aumentar la productividad de los programas y reducir el tiempo de depuración y mantenimiento de estos.

RECURSOS ABSTRACTOS

Son los recursos con los que no se cuentan a la hora de programar, pero funcionan como apoyo a la hora de emplearlos. Estos recursos deben ir transformándose en recursos concretos.

DISEÑO DESCENDENTE (TOP-DOWN)

Es el proceso mediante el cual un problema, se descompone en una serie de niveles o pasos sucesivos de refinamiento (*stepwise*).

ESTRUCTURAS DE CONTROL

Las estructuras de control de un lenguaje de programación son métodos de especificación de la forma en que las instrucciones determinan el flujo de control.

LAS TRES ESTRUCTURAS DE CONTROL BASICO SON

Secuencia – Selección – Repetición.

DESARROLLO:

SECUENCIA

En la estructura secuencial, las instrucciones del programa se ejecutan una después de otra, siguiendo el orden en el que fueron escritas en el programa.

SELECCIÓN

La estructura de selección plantea la selección entre una, dos o múltiples alternativas, de acuerdo con el resultado de la evaluación de una condición o predicado.

REPETICIÓN (ITERACIÓN).

La estructura de repetición consiste en la ejecución repetida de una instrucción, mientras sea verdadera una determinada condición.

6.6 TIPOS DE DATOS

Cuando en robótica, se emplean sensores, estos están tomando información del medio. Esta información, a su vez, constituye datos, sean de luz, presión, temperatura, distancias.

Dependiendo de la sensibilidad requerida para el sistema, se necesitaría que esos datos sean enteros, con decimales. O si se ha desarrollado una interface, reemplazar esas señales o datos por una cadena de caracteres a mostrar en un display.

Los datos que se emplearan en el desarrollo de un programa pueden ser clasificados en:

SIMPLES
ESTRUCTURADOS

Figura 6.11. PE –Clasificación de datos.

DATOS SIMPLES

Dentro de los tipos de datos simples, comúnmente empleados, se encuentran:

int	char	float	short	long	double	long double

Figura 6.12. Tipos de datos más empleados.

A su vez, estos datos pueden ser subdivididos en:

int	signed
	unsigned

char	signed
	unsigned

short	signed
	unsigned

long	signed
	unsigned

SIGNED – UNSIGNED - SIGNIFICADO

signed y unsigned

Son modificadores, que hacen referencia al signo del dato (se aplican a los datos numéricos).

signed

Permite que un número sea positivo o negativo, es decir, que tiene en cuenta el signo.

unsigned

No tiene en cuenta el signo y lo considera siempre positivo.

short y long

También son modificadores. Hacen referencia al número de cifras o rango de los datos.

Dentro de los tipos de datos simples, se debe diferenciar entre aquellos que han de ser empleados para trabajar con tipos de datos enteros o reales:

Tipos de datos enteros: char, int, short, long
Tipos de datos reales: float, double, y long double

Figura 6.13. Clasificación. Tipos de datos. Empleo con enteros o reales.

Por defecto, si no se realiza ninguna declaración, se emplean los modificadores short y signed.

DESCRIPCIÓN DE LOS TIPOS DE DATOS SIMPLES

int (entero)

Se emplea para operar con números enteros con signo, en un rango numérico de (2 bytes).

unsigned int (entero sin singo)

Para este tipo, el rango de valores es de 0...65.535 (2 bytes) para 16 bits, y 0...4.294.967.296 (4 bytes) para 32 bits.

char (carácter)

Las variables de este tipo contienen un solo carácter, que se almacena en 1 byte (8 bits).

unsigned char

En la representación unsigned char, al abarcar el rango de posiciones de memoria del 0 al 255, podemos almacenar cualquier dato correspondiente al código ASCII.

short (corto)

Su aplicación es similar al empleo de int. Al igual que en los casos anteriores, el total de valores para los tipos con signo se divide entre positivos y negativos.

unsigned short (corto sin signo)

Para el tipo short con modificador unsigned, el rango de valores es de 2 bytes = 16 bits = 2^{16} = 0...65.535.

long (largo)

Este tipo de dato se emplea para el trabajo con enteros largos. Su rango numérico es de 4 bytes = 32 bits, entre los valores -2.147.483.648... 2.147.483.647.

unsigned long (largo sin signo)

Este tipo de dato se emplea para el trabajo con enteros largos sin signo. Su rango numérico es de 4 bytes = 32 bits, entre los valores 0...4.294.967.296.

float (flotante) – reales de simple precisión

El tipo de dato float se emplea para representar números reales, es decir, que se puede trabajar con números que poseen una parte entera, y una parte fraccionaria o decimal (coma flotante en informática y digitales). La representación en memoria de este tipo de datos es de 4 bytes - 32 bits, en los que se pueden representar valores positivos y negativos -3.402823E+38... 3.402823E+38.

double (doble) – reales en doble precisión

El tipo de dato double es también empleado para representar números reales en coma flotante, empleando 8 bytes – 64 bits. En el rango de -1.797693E+308… 1.797693E+308.

long double (doble largo) – reales doble precisión formato largo

El tipo de dato double es también empleado para representar números reales en coma flotante, pero empleando 10 bytes – 80 bits. En el rango de -1.189731E+4932… 1.189731E+4932.

TIPOS DE DATOS ESTRUCTURADOS

Los tipos de datos estructurados o derivados se construyen a partir de los tipos de datos simples. Estos tipos de datos, además, se dividen en:

ESTRUCTURAS ESTÁTICAS
ESTRUCTURAS DINÁMICAS

Figura 6.14. División de los datos estructurados.

Dentro de las estructuras estáticas, se encuentran las siguientes:

ESTRUCTURAS ESTÁTICAS
FUNCIONES
ARRAYS
PUNTEROS
ESTRUCTURAS
UNIONES

Figura 6.15. Estructuras Estáticas. Componentes.

ESTRUCTURAS DINÁMICAS
LISTAS
PILAS
COLAS
ÁRBOLES

Figura 6.16. Estructuras Dinámicas. Componentes.

6.7 DESARROLLO DE ALGORITMOS

En el trabajo en desarrollo de algoritmos, generalmente se emplean tres herramientas o recursos:

DESARROLLO DE DIAGRAMA
PRUEBA DE ESCRITORIO
CODIFICACIÓN

Figura 6.17. Desarrollo de Algoritmos. Herramientas o Recursos,

El empleo de estas herramientas o recursos propone aplicar las tres etapas de diseño, expuestas en 6.2.2.(Figura 6.1):

Analizar y comprender el problema; desarrollar el algoritmo; realizar la comprobación de este; codificar.

El desarrollo del diagrama permite graficar el algoritmo diseñado. La prueba de escritorio será útil para verificar el diseño correcto del diagrama. Una vez que éste ha sido verificado se procede a su codificación en un lenguaje de programación.

6.7.1 DIAGRAMAS

DIAGRAMAS

Los diagramas son la representación gráfica de un algoritmo.

Los más empleados son los de Flujo y Nassi Shneiderman,

El diagrama es el primer paso que desarrolla el programador. En función de éste, una vez validado, desarrollará la codificación del programa.

6.7.2 DIAGRAMAS DE FLUJO

Como se expuso, un diagrama de flujo (*flowchart*) es la representación gráfica de un algoritmo. Adquiere su denominación del flujo de circulación indicado por las flechas **conectoras** entre los diferentes diagramas.

En el diagrama de flujo, se muestra de forma gráfica los pasos y/o procesos a seguir para alcanzar la resolución de un problema.

6.7.3 DIAGRAMAS DE FLUJO SÍMBOLOS BÁSICOS

Representación del Símbolo.	Explicación del Símbolo.
	Representa inicio y fin del diagrama de flujo.
	Se emplea para entrada/salida de datos.
	Se emplea para representar un proceso. En su interior se expresan asignaciones, operaciones aritméticas, entre otras.
	Se emplea para representar una decisión. En su interior se almacena una condición; y, dependiendo del resultado de la evaluación de esta, se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza en la estructura selectiva.
	Se emplea para representar la impresión por pantalla.
	Se emplean para expresar la dirección del flujo del diagrama.

Figura 6.18. Diagramas de Flujo. Símbolos Básicos.

Sin lugar a dudas, la ingeniería de software no deja de dar grandes frutos. Ejemplo de esto es Visual Designer la aplicación desarrollada por PROTEUS para trabajar con Arduino (Capítulo 10), enfocada en la enseñanza e introducción a los Sistemas Embebidos.

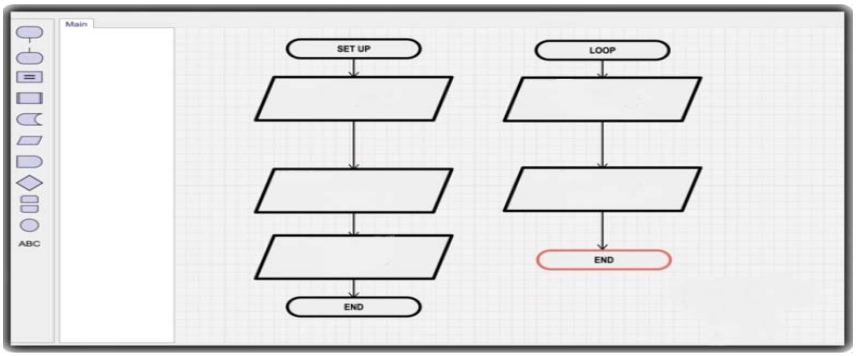


Figura 6.19. Visual Designer de PROTEUS.

La palabra **Embebido** proviene de *embedded* (incrustado o empotrado). Un sistema embebido está compuesto por un Microcontrolador (Capítulo 8), FPGA (12), Microprocesador, y ha sido diseñado para aplicaciones específicas.

SUITE es un conjunto de aplicaciones y herramientas de software, incluidas en un mismo programa, integrándose entre sí.

En Proteus Design Suite se selecciona la opción `NEW FLOWCHART`; se genera un proyecto asignando nombre. Se elige el entorno `ARDUINO`, la placa a emplear, compilador, periféricos (*hardware - shields*),

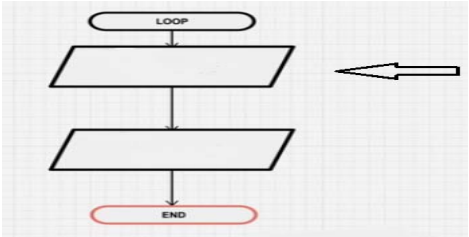


Figura 6.20. Opción de símbolo elegida en función de la aplicación.

En caso de seleccionar un *shield* de diodo LED (con resistencia serie), se configuran los periféricos como entrada (prendido) o salida (apagado), como se observa en la figura siguiente. El símbolo empleado para entrada y salida es el que se presentó en la Figura 6.18. (Diagramas de Flujo. Símbolos Básicos).

Figura 6.21. Símbolo. Operación de entrada/salida de un periférico.

6.7.4 DIAGRAMAS DE NASSI-SHNEIDERMAN

Los diagramas de Nassi-Shneiderman N-S (diagramas estructurados), también llamados diagramas de Chapin, son una técnica para la especificación de algoritmos, que combina la descripción textual del pseudocódigo con la representación gráfica del diagrama de flujo.

Para el desarrollo de algoritmos largos, los diagramas de Nassi-Shneiderman presentan como beneficio que los símbolos son contiguos. Por tanto, no se emplean flechas. Esto, entre otras ventajas, acorta el largo o tamaño de los diagramas.

Los diagramas Nassi-Shneiderman, fueron desarrollados para su aplicación en el desarrollo de algoritmos bajo el paradigma de la Programación Estructurada. Por tanto, existe un símbolo para representar cada una de sus 3 estructuras básicas (Figura 6.9. Tres estructuras de control básicas). Esto constituye otra ventaja; reducir la cantidad de símbolos existentes.

ESTRUCTURAS DE CONTROL BÁSICAS
SECUENCIA
SELECCIÓN
REPETICIÓN

ESTRUCTURA SECUENCIAL

SÍMBOLO N-S

Figura 6.22. Símbolo N-S. Estructura. Secuencial.

En Nassi-Shneiderman, se dispone de un símbolo único para la representación de instrucciones secuenciales, que es empleado para las operaciones de:

ASIGNACIÓN
DECLARACIÓN
ENTRADA
SALIDA

Figura 6.23. DNS. Secuencia. Operaciones representadas.

ESTRUCTURA SELECTIVA

SÍMBOLO N-S

Instrucciones –>simples (if) y compuestas (if-else).

Figura 6.24. Símbolo NS. Estructura. Selectiva Simple y Compuesta.

Instrucción –>Múltiple (switch).

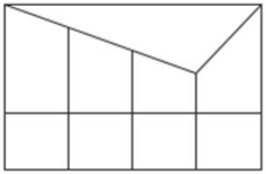


Figura 6.25. Símbolo NS. Estructura. Selectiva Múltiple.

ESTRUCTURA REPETITIVA

SÍMBOLO N-S

Instrucciones –> para (for) - mientras (while).

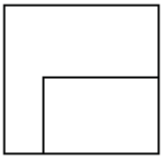


Figura 6.26. Símbolo NS. Estructuras Repetitivas - para y mientras.

Instrucción –> hacer mientras (do while).

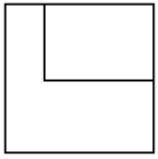


Figura 6.27. Símbolo NS. Estructura Repetitiva – hacer mientras.

ANIDAMIENTO Y COMBINACIÓN DE ESTRUCTURAS Y SÍMBOLOS.

En el trabajo con diagramas de Nassi-Shneiderman, para el desarrollo de estructuras de algoritmos más complejos, se deben anidar los símbolos existentes.

La siguiente figura integra los símbolos de los diagramas de Nassi-Shneiderman desarrollados, y su significado o función.

Representación del Símbolo.	Explicación del Símbolo.
	SECUENCIA: empleado para representar instrucciones de asignación, declaración, entrada y salida.
	SELECCIÓN: empleado para representar instrucciones de selección, simples o compuestas.
	SELECCIÓN: empleado para representar instrucciones de selección múltiples.
	REPETICIÓN: empleado para representar instrucciones repetitivas (iteración), tales como mientras, o para.
	REPETICIÓN: empleado para representar la instrucción repetir (la instrucción `repeat` no es empleada en lenguaje C).

Figura 6.28. Símbolos Nassi-Shneiderman y su representación.

6.7.5 STRUCTORIZER- SOFTWARE

Structorizer es un software concebido para el desarrollo de diagramas Nassi-Shneiderman, bajo el paradigma GNU/GPL.

Inicialmente, cuando se pensaba en un programa o software para el diseño de diagramas, se lo concebía con el fin de imprimir o documentar el algoritmo desarrollado. Pero en la actualidad, la posibilidad de aplicaciones de estos se ha

incrementado. Inclusive permiten, una vez realizados los diagramas, ser portados a un IDE de Programación, y generar la estructura de código.

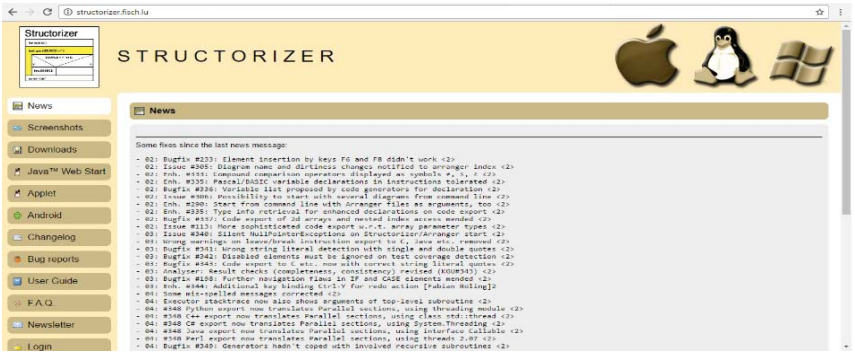


Figura 6.29. Structorizer. Website.

Desde el sitio Web, se puede descargar el software, eligiendo entre las opciones disponibles, aquella que aplique al sistema operativo usado.

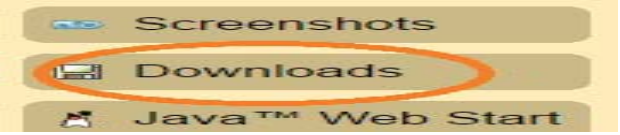


Figura 6.30. Opción de descarga.

Figura 6.31. Descarga para Windows y Linux.

Una vez descargado e instalado el programa, se observa la siguiente apariencia de pantalla.

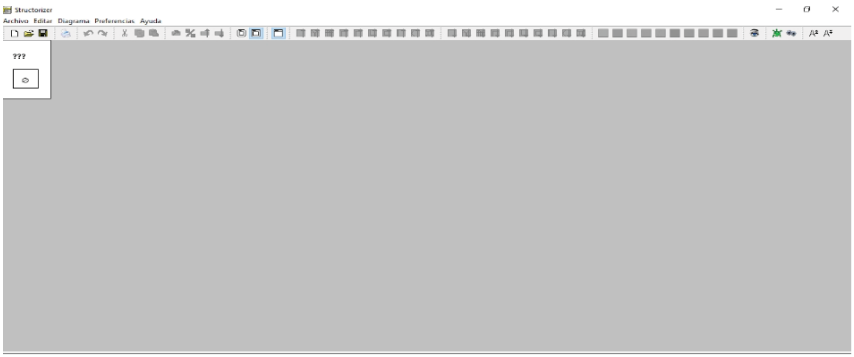


Figura 6.32. Structorizer IDE.

Ejemplo de aplicación:

Consigna: implementar diagrama de Nassi-Shneiderman. Se debe emplear Structorizer, para representar el algoritmo de trabajo de un sensor óptico reflectivo CNY70, en la implementación de un robot seguidor de línea.

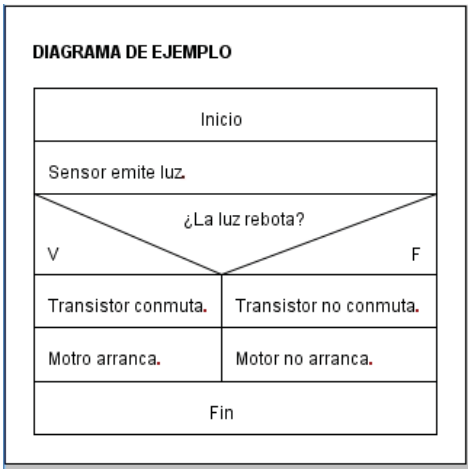


Figura 6.33. Diagrama de Nassi-Shneiderman implementado con Structorizer.

6.8 VALIDACIÓN O VERIFICACIÓN DE ALGORITMOS

Una vez desarrollado el algoritmo, se debe realizar la verificación o comprobación del correcto funcionamiento de este. Uno de los métodos que permite realizar este paso es la Prueba de Escritorio. Si la verificación es correcta, se procede a realizar el código de programa.

6.8.1 PRUEBAS DE ESCRITORIO (PE)

La prueba de escritorio es una técnica empleada para la comprobación del funcionamiento correcto de un algoritmo desarrollado.

Desde el punto de vista de su implementación, consiste en generar una tabla con tantas columnas como variables o acciones clave tenga el algoritmo. El objetivo es asignarles valores, analizar su comportamiento, y en función de éste, validar o no, su funcionamiento.

No existe un número definido de casos de prueba que se deben generar, si no que depende la cantidad de casos de prueba de la complejidad del algoritmo. En general, se puede considerar suficiente un total de 3 a 5 casos de prueba, donde se debe considerar tanto escenarios normales, como así también casos extremos.

Conviene el empleo de diagramas en el desarrollo de algoritmos, y su posterior verificación, a fin de adquirir una estructuración lógica, frente a la resolución de problemas.

6.9 LENGUAJE ENSAMBLADOR O ASSEMBLER

El Ensamblador o Assembler fue el lenguaje principal, empleado durante décadas, para la programación de Microcontroladores, a fin de aplicar los mismos al diseño de robots con lógica de Microcontrolador.

Su potencial reside, por ejemplo, en la eficiencia del código y velocidad, dada su cercanía al hardware. Es por esta razón que se emplea también para el desarrollo de drivers.

Cada Microcontrolador, tiene asociado un Set de Instrucciones, en función su CPU.

Las instrucciones en Assembler, a diferencia de los lenguajes de alto nivel, en los cuales se emplean palabras del idioma inglés, son llamadas mnemónicos. La razón es que fueron pensadas a fin de facilitar su memorización.

INSTRUCCIÓN	SINTAXIS	SIGNIFICADO
ASSEMBLER	`MOVLW`	Copiar literalmente en W
ALTO NIVEL	`printf`	Imprimir o mostrar por pantalla

Figura 6.34. Ejemplo de Instrucción en Assembler y Alto Nivel.

6.9.1 SET DE INSTRUCCIONES

El set de instrucciones (INSTRUCTION SET) es el conjunto de instrucciones que se puede emplear para programar un microcontrolador.

Estas instrucciones serán dependientes de la arquitectura y CPU a emplear.

6.10 IDE – ENTORNO DE DESARROLLO INTEGRADO

En la actualidad, para quienes se desarrollan en las áreas de diseño electrónico, o programación, el empleo de un IDE - Entorno de Desarrollo Integrado, resulta algo cotidiano. Pero, en realidad su masificación se produjo a partir del segundo milenio.

Un IDE es un programa o software, que integra las opciones de diseño de programa, compilación, generación de archivos `.obj` y `.hex`. E inclusive en las aplicaciones con microcontroladores, permite conectar el programador vía USB, realizar la carga del programa en memoria, verificar la correcta recepción del paquete de datos, y debuggear en circuito.

6.10.1 INTRODUCCIÓN

En la actualidad, Microchip, fabricante y líder mundial en microcontroladores de 8 bits, tiene disponibles dos IDE, que pueden ser descargados de su sitio Web. La versión más reciente es MPLAB X, pero se mantiene aún el clásico MPLAB.

6.10.2 IDE MPLAB - ASSEMBLER

Ocurre frecuentemente en la electrónica, o en las aplicaciones de software, que familiarizarse con un IDE resulta confiable y ágil, dado el conocimiento que se dispone del mismo. Afortunadamente la tecnología avanza y proyecta nuevos desarrollos y aplicaciones para las que se debe contar también con herramientas más potentes e integradoras. En esta línea entonces, dadas las crecientes familias de dispositivos de 16 y 32 bits, Microchip desarrolla el IDE MPLABX.

Para quien se inicia en esta disciplina, se le brinda la posibilidad de trabajar sus primeros pasos con MPLABX, potente herramienta de desarrollo. Y para aquellos habituados al trabajo con el IDE MPLAB, se les brinda también la forma de migrar a MPLABX, a través de los ejemplos de aplicación en robótica.

Se puede descargar el IDE MPLAB del sitio Web de Microchip.

Figura 6.35. IDE MPLAB. Descarga del website de Microchip.

Figura 6.36. IDE MPLAB.

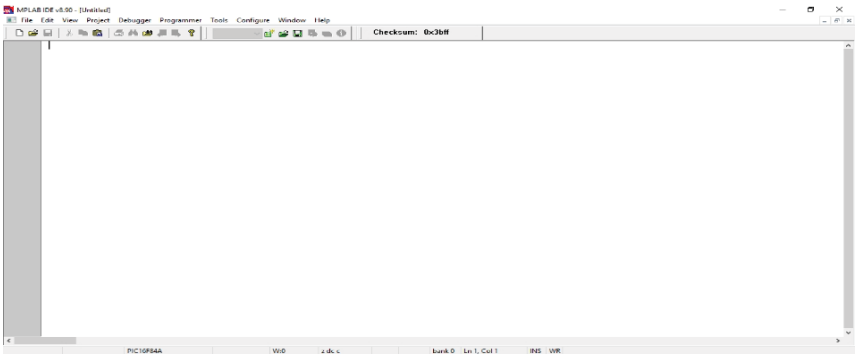


Figura 6.37. MPLAB. Formulario o sábana.

Ejemplo de aplicación en robótica.

Programa que permite leer, en el Pin 2 del Puerto A, el cambio de estado de un sensor CNY70. Si el sensor cambia de estado, se lee un 1 y se produce la activación de un actuador conectado al Pin 3 del Puerto A. Luego retorna para volver a leer el estado del sensor. Si el sensor no cambia de estado, se produce un salto a la etiqueta SENSAR, con el fin de volver a leer hasta que se produzca

un cambio de estado del sensor (el bloque de instrucciones del ejemplo hace referencia solamente a la lectura del seteo o activación del sensor y posterior encendido del actuador).

Se emplea el microcontrolador 16F84A.

```
;ROCHA DIA2. Leer CNY70

        INCLUDE <HEAD.inc>

        ORG 0
        BSF     STATUS,5    ;Cambia al banco 1.
        MOVLW   b'00000100' ;Selecciona el pin 2 del Port A como entrada.
        MOVWF   PORTA       ;Copia la configuración seteada al Port A.
        BCF     STATUS,5    ;Vuelve al banco 0.
SENSAR  BTFSS   PORTA, 2    ;Testea si el pin 2 cambió de estado.
        GOTO    SENSAR      ;Si no hubo cambio de estado, vuelve a esperar cambio.
        BSF     PORTA, 3    ;Si el bit está seteado enciende el actuador conectado al pin 3.
        GOTO    SENSAR      ;Producido un envento a través del Pin 3, vuelve a sensar.

        END
```

Figura 6.38. Ejemplo de aplicación en robótica del lenguaje Assembler.

Se observa en el encabezado del programa la inclusión o llamado de un archivo denominado `HEAD.inc`. Este archivo contiene datos necesarios para la configuración del microcontrolador a emplear, referida al hardware, seguridad, y alimentación.

El archivo ha sido generado con el fin de reducir líneas de código. Si se lo abre con el bloc de notas, se leen las siguientes líneas:

```
PROCESSOR PIC16F84A
INCLUDE<P16F84A.INC>
__CONFIG _XT_OSC &_WDT_OFF &_PWRTE_ON
```

Figura 6.39. Archivo `HEAD.inc`. Contenido de líneas de código.

En la primera línea, se ha detallado el modelo de microcontrolador a emplear con `PROCESSOR PIC16F84A`. En la segunda línea, se llama incluye al archivo **include** de este micro, `INCLUDE<P16F84A.INC>`, que contiene información sobre los registros internos del mismo. La tercera línea `__CONFIG`, está relacionada con el hardware `_XT_OSC`, un registro de seguridad _WDT_OFF y opciones de alimentación o temporización de encendido, `__PWRTE_ON`.

```
;ROCHA DIAZ. Leer CNY70

            INCLUDE <HEAD.inc>

            ORG 0
            BSF         STATUS,5    ;Cambia al banco 1.
            MOVLW       b'00000100' ;Selecciona el pin 2 del Port A como entrada.
            MOVWF       PORTA       ;Copia la configuración seteada al Port A.
            BCF         STATUS,5    ;Vuelve al banco 0.
SENSAR      BTFSS       PORTA, 2    ;Testea si el pin 2 cambió de estado.
            GOTO        SENSAR      ;Si no hubo cambio de estado, vuelve a esperar cambio.
            BSF         PORTA, 3    ;Si el bit está seteado enciende el actuador conectado al pin 3.
            GOTO        SENSAR      ;Producido un envento a través del Pin 3, vuelve a sensar.

            END
```

Figura 6.40. Vista ampliada. Código de programa.

Una vez realizado el código de programa, se debe pasar a la etapa descrita como compilación, en la que las líneas de código en lenguaje interpretado por las personas son convertidas a lenguaje binario o código de máquina.

El archivo debe ser guardado con la extensión `.asm`.

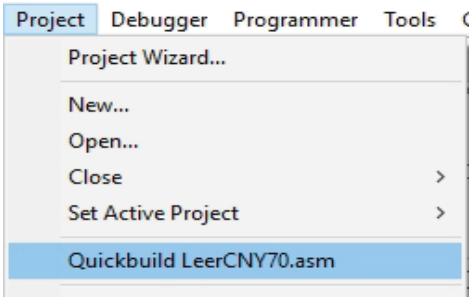


Figura 6.41. Pasos para la compilación del programa.

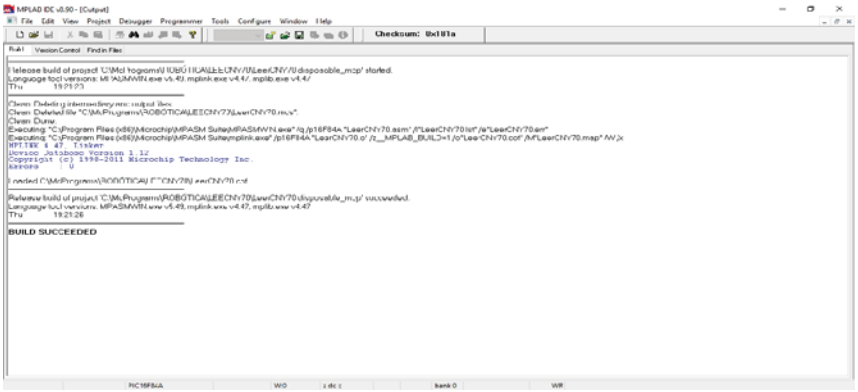


Figura 6.42. Compilación y generación de archivo ejecutable.

6.10.3 IDE MPLABX – LENGUAJE C

El IDE MPLAB X puede ser ejecutado en los sistemas operativos Windows, Mac, Linux. A diferencia del MPLAB, que fue desarrollado completamente por Microchip, éste se basa en el IDE **NetBeans** de código abierto de Oracle. Esto permite dar respuesta a características frecuentemente solicitadas, como así también ofrecer nuevas características en el futuro.

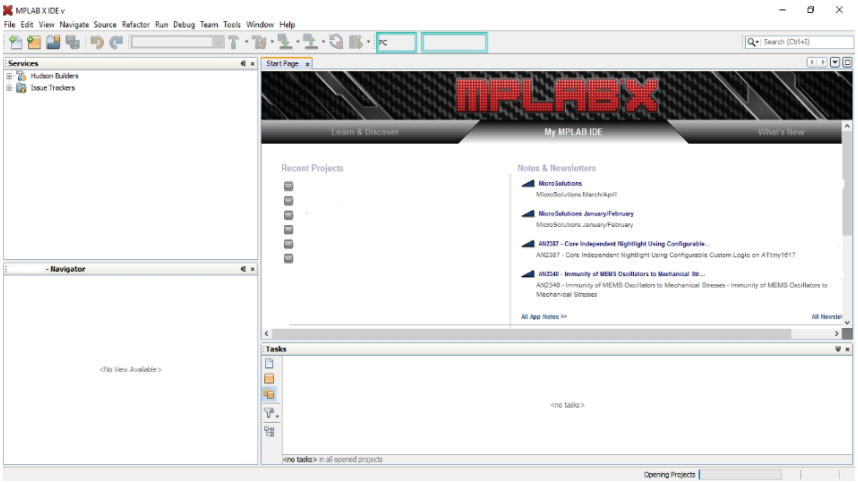


Figura 6.43. IDE MPLAB X.

A diferencia del programa desarrollado con el empleo del IDE MPLAB para el lenguaje Assembler, se deberá ahora generar un proyecto dentro del cual estará el programa.

Durante la creación del proyecto, se genera una carpeta con el nombre de este. Allí están incluidas las librerías, el archivo fuente, y los datos relacionados con la aplicación.

Se toma como ejemplo el programa desarrollado para el IDE MPLAB, pero esta vez con el empleo del microcontrolador 18F4610.

Este microcontrolador será presentado, al igual que el microcontrolador 16F84A, en el Capítulo 8 – Microcontroladores.

Ejemplo de aplicación en robótica.

Programa que permite leer en el Pin 2 del Puerto A, el cambio de estado de un sensor CNY70. Si el sensor cambia de estado, se lee un 1 y produce la activación de un actuador conectado al Pin 3 del Puerto A, y luego retorna para volver a leer el estado del sensor. Si el sensor no cambia de estado, se produce un salto a la etiqueta SENSAR, con el fin de volver a leer hasta que se produzca un cambio de estado del sensor.

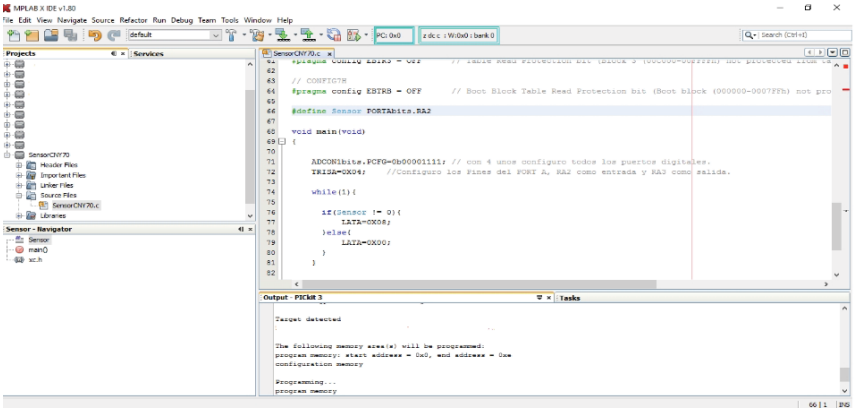


Figura 6.44. Ejemplo de aplicación en robótica del lenguaje C con MPLABX.

```
66  #define Sensor PORTAbits.RA2
67
68  void main(void)
69  {
70
71      ADCON1bits.PCFG=0b00001111; // con 4 unos configuro todos los puertos digitales.
72      TRISA=0X04;    //Configuro los Pines del PORT A, RA2 como entrada y RA3 como salida.
73
74      while(1){
75
76        if(Sensor != 0){
77            LATA=0X08;
78        }else{
79            LATA=0X00;
80        }
81      }
82
83      return;
84  }
```

Figura 6.45. Vista ampliada del código de programa

Se observa en la figura anterior el inicio de código en la línea 66. Esto, como se verá en el Capítulo 8, responde a que las líneas anteriores son empleadas para configuración de fusibles.

6.10.4 IDE DEV-C++

Una buena alternativa para el inicio del trabajo en programación y, con el lenguaje C, es descargar el IDE Dev-C++, que aplica al paradigma de GNU GPL.

Puede ser descargado del sitio Web:

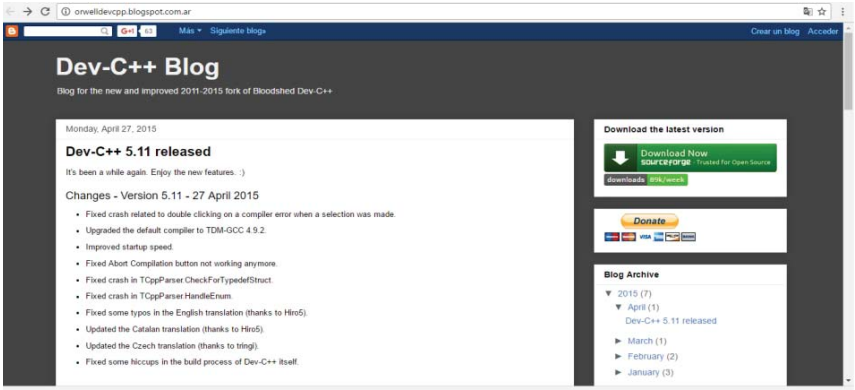


Figura 6.46. IDE Dev-C++ website. Descarga.

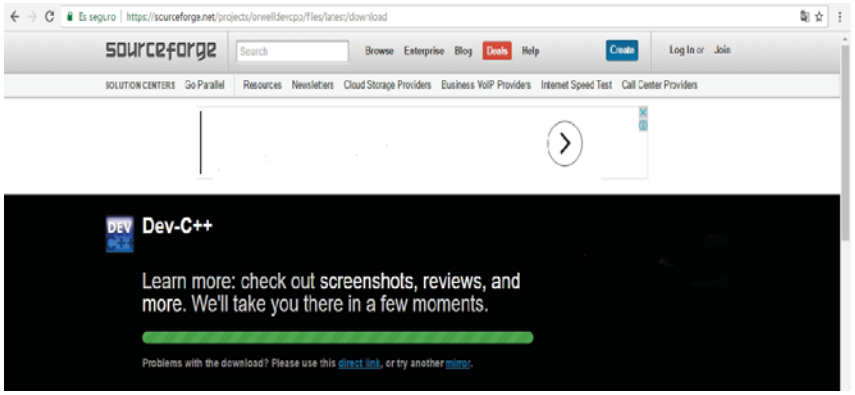
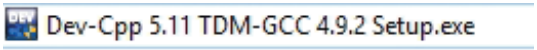


Figura 6.47. IDE Dev-C++ descarga.

La versión de descarga actual es:



Como ejemplo de aplicación de este IDE, se tomará el ejemplo trabajado para el IDE MPLAB y MPLABX.

Se debe tener en cuenta que la sintaxis será similar a la empleada en el IDE MPLABX, debido a que el lenguaje de programación es el mismo.

Sin embargo, en el ejemplo siguiente se genera la estructura del programa sin configuración de puertos o registros, ya que, a diferencia del trabajo con microcontroladores, no existe un cambio de estado por eventos provenientes de señales externas. Simulándose el mismo con la introducción de datos empleando la instrucción `scanf` para leer el estado del sensor.

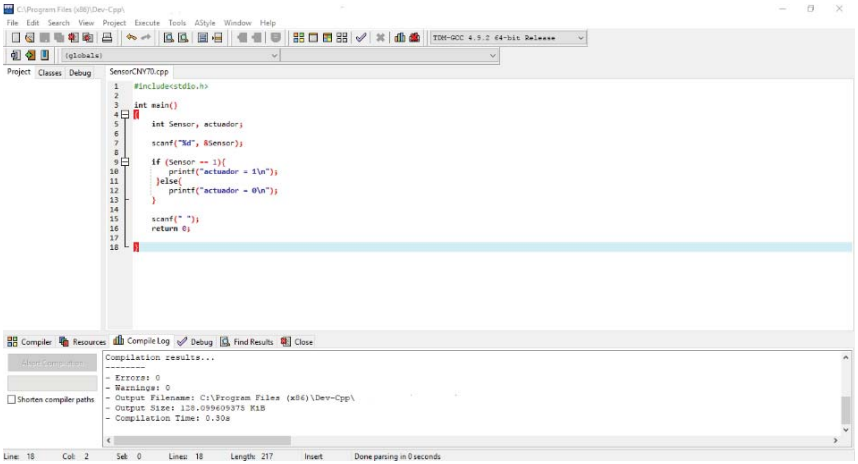


Figura 6.48. Ejemplo de aplicación en robótica empleando el IDE DEV-C++.

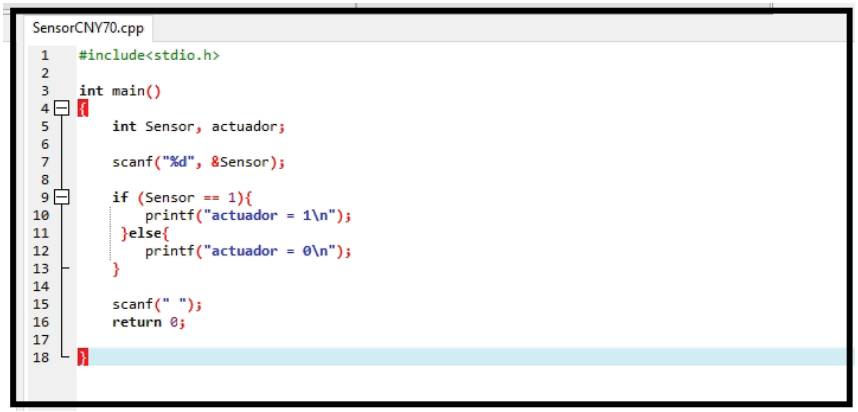


Figura 6.49. Vista ampliada del código de programa.

6.11 LENGUAJE C

C es, sin lugar a dudas, el lenguaje de programación más importante e influyente del que se tenga cuenta. De él derivan sistemas operativos, lenguajes de programación, y cientos de miles de aplicaciones.

En muchos casos, es considerado un lenguaje intermedio, dado que posee las características de alto nivel, pero permite control a muy bajo nivel.

Se expuso que ha sido extendido al empleo en la electrónica, debido al crecimiento en el hardware de los ordenadores. Esto permitió adicionar

compiladores y paquetes de software a la actividad de Microcontroladores, por ejemplo.

6.11.1 HISTORIA

El lenguaje C nace en los Laboratorios Bell de AT&T en el año 1972. Su historia y evolución están ligados al sistema operativo UNIX.

El lenguaje C cumple con las características y requisitos de la Programación Estructurada.

6.11.2 BIBLIOTECA ESTÁNDAR DE C

La biblioteca estándar del lenguaje C es libc. Y proporciona entre otras: declaraciones, tipos, funciones, macros, fórmulas matemáticas, archivos para manejo de memoria.

Una de las bibliotecas más empleadas es `stdio.h` (estándar de entrada, salida), que contiene, entre otras, las instrucciones o funciones de entrada (`scanf`), y salida (`printf`). Esto se produce porque originalmente el lenguaje C no poseía, como nativa, una función para dichos estándares.

6.11.3 FUNCIONES

Una función es un conjunto de instrucciones o líneas de código, que, al igual que su homónimo en matemática, realizan una tarea específica y retornan un valor o resultado.

Las instrucciones forman parte de los módulos de programa (funciones) expuestos anteriormente, los cuales hacen a la buena práctica de programación, dado que permiten, por ejemplo, implementar operaciones comúnmente requeridas, e invocarlas cuando sean necesarias, para reducir las líneas de código.

Hay dos tipos de funciones, las propias de la librería estándar o desarrolladas por empresas dedicadas al desarrollo de software, y las que define el programador.

Para el empleo de las bibliotecas que forman parte de la biblioteca estándar, se emplea la instrucción `#include`.

Ejemplo: inclusión de la biblioteca `stdio.h` en un programa

```
#include<stdio.h>
```

Para realizar funciones propias, se debe, en cambio, definir el prototipo de la función y desarrollar la misma (puede ser al pie del programa), e invocarla desde el programa o función principal `main`. Todo programa desarrollado en C posee una función principal, llamada `main`.

Ejemplo: desarrollo de una función llamada cuadrado, que permite duplicar la velocidad en los motores de un robot.

```
int cuadrado( int y)/* Declaro el prototipo de la función*/
    printf( "%d\n", cuadrado( x ) ); /*invoco a la función desde el programa */
/*Desarrollo de la función*/
int cuadrado( int y )
{
   return y * y;
 }
```

6.11.4 ESTRUCTURA BÁSICA DE UN PROGAMA EN C

Una buena práctica para el desarrollo de algoritmos se vio consiste en desarrollar un diagrama de Nassi Shneiderman, realizar luego la verificación de este empleando una prueba de escritorio y, finalmente realizar su codificación.

Para analizar la estructura de un programa en Lenguaje C, se tomará como ejemplo el programa trabajado en las figuras 6.48 y 6.49.

Donde:

```
1  #include<stdio.h>
2
3  int main()
4  {
```

Figura 6.50. Inicio de un programa en Lenguaje C.

En la línea 1: se incluye el archivo `stdio.h`, que posee declaraciones de funciones necesarias para el desarrollo de las aplicaciones.

Línea 3: muestra el inicio del programa en la función general llamada main.

Línea 4: la llave indica que contiene las líneas de código siguientes.

```
 5        int Sensor, actuador;
 6
 7        scanf("%d", &Sensor );
 8
 9        if (Sensor == 1){
10            printf("actuador = 1\n");
11         }else{
12            printf("actuador = 0\n");
13        }
```

Figura 6.51. Cuerpo del programa en Lenguaje C.

Las líneas 5 a la 13: representan el cuerpo de instrucciones del programa.

En la línea 5: se declaran las variables enteras Sensor y actuador.

Línea 7: se emplea la instrucción `scanf` para introducir el valor correspondiente al estado del sensor.

Líneas 9 y 11: se emplea la que será definida como estructura selectiva compuesta `if-else`, para analizar si es verdadera la condición de que Sensor sea igual a 1. En caso de que la condición sea verdadera o True (línea 9), se ejecuta la instrucción siguiente.

Línea 10: el sensor ha cambiado de estado, por lo que se enciende o pone en 1 el actuador (se simula **visualiza** un estado de activación).

Línea 11: sino es cierta la condición evaluada en la línea 9 (`if`), entonces se ejecuta/n la/las instrucciones contenida/s dentro del cuerpo (llaves `{ }`) de la instrucción `else`.

Línea 12: si Sensor vale 0, el actuador queda en estado bajo o apagado.

Línea 13: la llave cierra el cuerpo de la instrucción `else` (entonces), abierto en la línea 11.

Las llaves al igual que en la instrucción `if`, son empleadas como buena práctica de programación.

El fin de programa está representado por las líneas 15, 16 y 18.

Línea 15: la instrucción `scanf`(“ “) es empleada para congelar la imagen y visualizar la aplicación una vez compilada y ejecutada.

Línea 16: la instrucción `return 0` devuelve el flujo del control del programa a la función `main`.

Línea 18: obsérvese que la llave abre en la línea 4 para la izquierda, y cierra ahora para la derecha.

```
15        scanf(" ");
16        return 0;
17
18    }
```

Figura 6.52. Instrucciones de cierre de programa.

Un detalle a tener en cuenta es que en lenguaje C todas las instrucciones terminan con punto y coma (;).

Las líneas en blanco entre las instrucciones son empleadas para aportar claridad al desarrollo del programa.

6.11.5 APLICACIÓN DE DESARROLLO DE ALGORITMOS

Como aplicación de lo expuesto en el ítem 6.7, y la Figura 6.17 desarrollo de algoritmos, etapas:

DESARROLLO DE DIAGRAMA
PRUEBA DE ESCRITORIO
CODIFICACIÓN

Se desarrollará el diagrama y la prueba de escritorio para el ejemplo anterior:

Consigna: implementar diagrama de Nassi-Shneiderman para representar un algoritmo que permita: **declarar una variable de nombre** `Sensor`**, una variable de nombre** `actuador` (es declarada para simular su participación a nivel de hardware), ingresar un valor por teclado, y si el valor ingresado es igual a 1, mostrar la leyenda Actuador = 1. Si el valor introducido a la variable Sensor es 0, se debe mostrar la leyenda Actuador = 0.

Luego, debe ser realizada la prueba de escritorio (PE). Una vez validado el algoritmo, se debe realizar también el código de programa en lenguaje C.

ESTRUCTURA SELECTIVA

DIAGRAMA DE NASSI-SHNEIDERMAN (DNS)

Figura 6.53. Diagrama de Nassi-Shneiderman. Implementación con Structorizer.

PRUEBA DE ESCRITORIO (PE)

Instrucción	Sensor	actuador	Condición	Mostrar
Inicio				
Declarar	--	--		
Ingresar: valor a la variable Sensor	1	--		
Si Sensor = 1	1	--	V	
Mostrar	1	--	V	Actuador = 1
Fin				

Figura 6.54. Desarrollo de Prueba de Escritorio. Se supone que valor es = a 1.

Instrucción	Sensor	actuador	Condición	Mostrar
Inicio				
Declarar	--	--		
Ingresar: valor a la variable Sensor	0	--		
Si Sensor = 1	0	--	F	
Mostrar	0	--	F	Actuador = 0
Fin				

Figura 6.55. Desarrollo de Prueba de Escritorio. Se supone que valor es = a 0.

CÓDIGO DE PROGRAMA EN LENGUAJE C

En la línea 7: para la instrucción `scanf`, necesaria para ingresar valores. El operador %d es un especificador de tipo de datos, que indica que los datos introducidos son de tipo decimal. El símbolo & es un operador conocido como de dirección, que asocia el valor introducido a una variable o posición de memoria.

En las líneas 10 y 12: se emplea la instrucción `printf`, necesaria para mostrar leyendas, caracteres o valores por pantalla. Para el caso de líneas de texto, se muestra la cadena de texto entre comillas. En caso de querer visualizar un valor almacenado en una variable, se debe emplear también el especificador de tipo de dato a visualizar.

El código de programa ha sido presentado en las figuras 6.48 y 6.49:

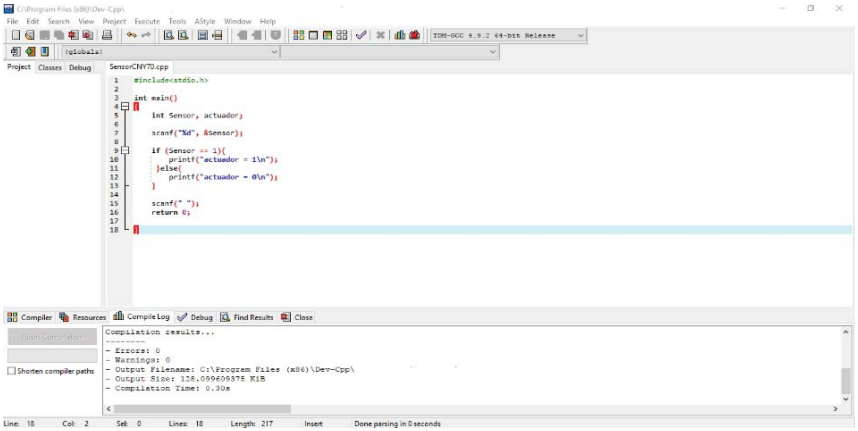


Figura 6.56. Código de programa en lenguaje C. IDE DEV-C++.

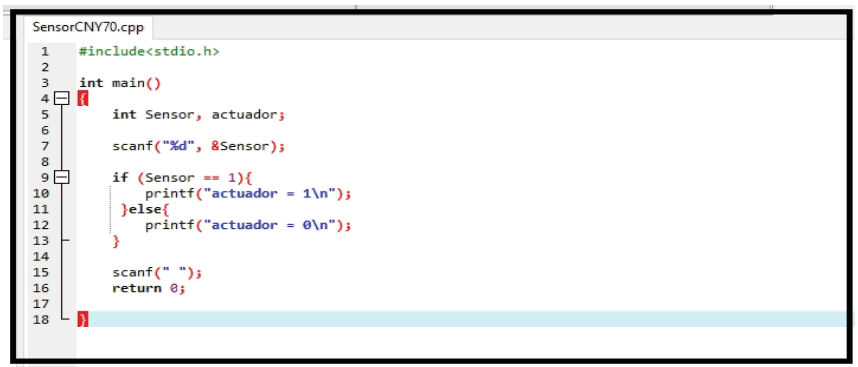


Figura 6.57. Imagen ampliada. Código de programa en lenguaje C.

El ejemplo anterior ha sido desarrollado empleando la estructura selectiva (compuesta `if-else`). La estructura selectiva es una de las tres estructuras básicas, propuestas para el paradigma de la programación estructurada.

Recuérdese que, para el ejemplo, la variable `actuador` pudo no haber sido declarada, ya que, en programación en alto nivel, **el efecto** es mostrar que si la variable `Sensor` vale 1, el `actuador` se enciende. Con excepción de aplicaciones que empleen los puertos o ports (ejemplo joystick o controlador), que habrán de requerir el empleo del nivel lógico almacenado en la variable.

Los siguientes ejemplos son empleados para representar las estructuras secuencial y repetitiva, aplicadas a la robótica.

La estructura secuencial no es aplicada de forma exclusiva para el desarrollo de aplicaciones de robótica, dada la necesidad de empleo de instrucciones de selección y, repetición o bucles de control para los robots.

ESTRUCTURA SECUENCIAL

La estructura secuencial, es aquella en la que las instrucciones están seguidas unas de otras (forman una secuencia). Para describirla, se puede tomar el ejemplo anterior, trabajado para la estructura selectiva.

Se puede observar que las primeras tres instrucciones del DNS son una secuencia:

Figura 6.58. Aplicación de secuencias en un programa.

También, puede ser visualizado en el programa:

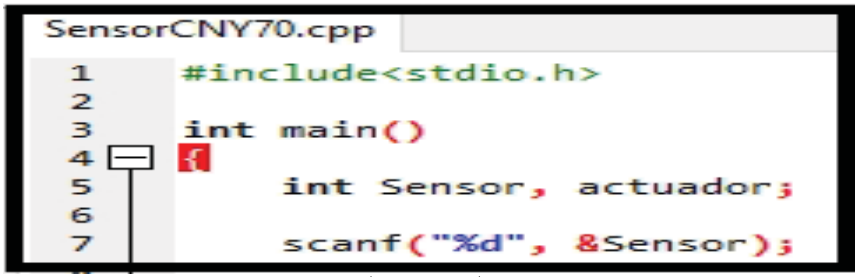


Figura 6.59. Las primeras líneas de código forman una secuencia.

ESTRUCTURA REPETITIVA

La estructura repetitiva, es aquella en que la ejecución de una instrucción se repite, mientras sea verdadera una determinada condición.

Ejemplo de aplicación en robótica:

Se emplea la instrucción mientras (`while`) para evaluar si el **sensor LDR** utilizado en el circuito de robot seguidor de luz, recibe luz. Se describe también (visualiza) la secuencia de funcionamiento.

Se desarrolla DNS, PE, y se codifica en lenguaje C.

DIAGRAMA DE NASSI-SHNEIDERMAN (DNS)

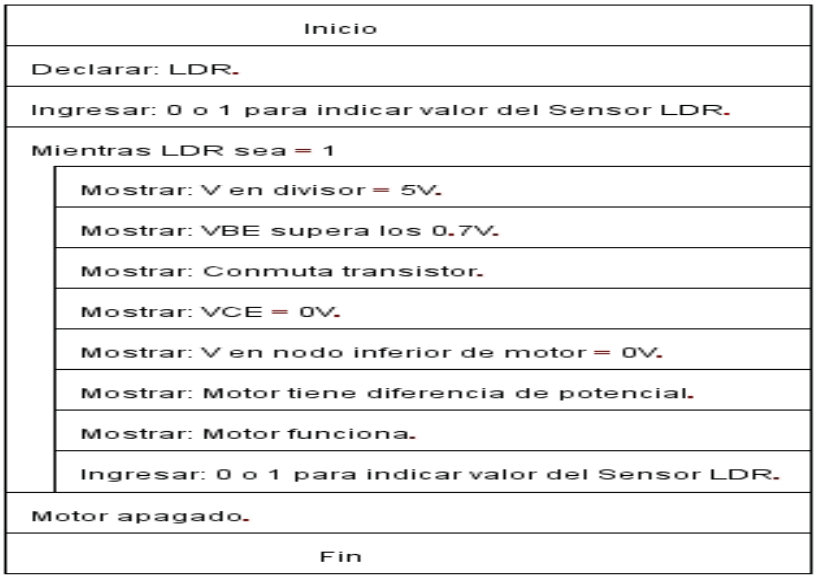


Figura 6.60. DNS aplicación de estructura repetitiva en robótica.

PRUEBA DE ESCRITORIO (PE)

Instrucción	LDR	Condición mientras	Mostrar
Inicio			
Declarar: LDR	--		
Ingresar: 0 o 1 para indicar valor del Sensor LDR.	1		
Mientras LDR sea = 1	1	V	
Mostrar: V en divisor = 5V.	1	V	V en divisor = 5V.
Mostrar: VBE supera los 0.7V.	1	V	VBE superior a 0.7V.
Mostrar: conmuta transistor.	1	V	conmuta transistor.
Mostrar: VCE = 0V.	1	V	VCE = 0V.
Mostrar: V en nodo inferior de motor = 0V.	1	V	V en nodo inferior de motor = 0V.
Mostrar: motor tiene diferencia de potencial.	1	V	motor tiene diferencia de potencial.
Mostrar: motor funciona.	1	V	motor funciona.
Ingresar: 0 o 1 para indicar valor del Sensor LDR	x		
Fin			

Figura 6.61. Desarrollo de Prueba de Escritorio. Se supone que valor es = a 1.

Instrucción	LDR	Condición mientras	Mostrar
Inicio			
Declarar: LDR	--		
Ingresar: 0 o 1 para indicar valor del Sensor LDR.	0		
Mientras LDR sea = 1	0	F	
Mostrar: motor apagado.	0	F	motor apagado.
Fin			

Figura 6.62. Desarrollo de Prueba de Escritorio. Se supone que valor es = a 0.

CÓDIGO DE PROGRAMA (CP)

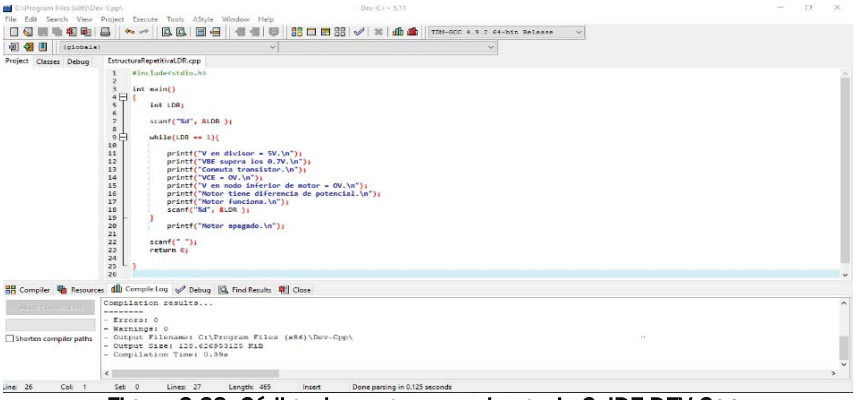


Figura 6.63. Código de programa en lenguaje C. IDE DEV-C++.

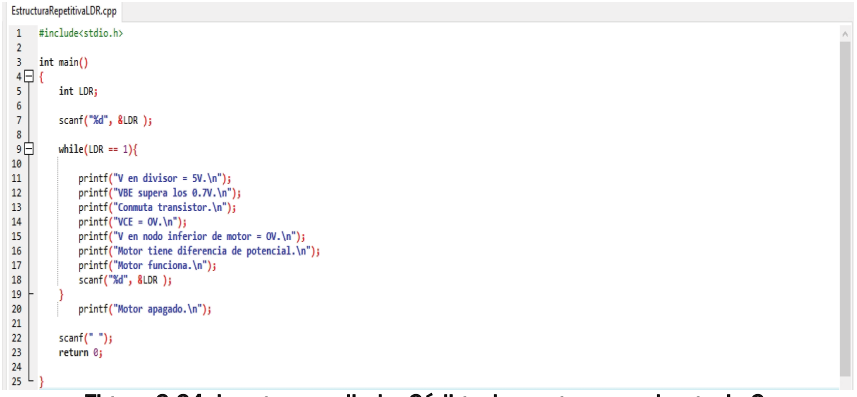


Figura 6.64. Imagen ampliada. Código de programa en lenguaje C.

Descripción de las líneas de programa:

Las líneas 1 a la 7: ya han sido desarrolladas en los programas anteriores.

Línea 9: si es verdadera la condición contenida en la instrucción `while`, se desarrollan las instrucciones contenidas dentro de su cuerpo de instrucción. El inicio del mismo está indicado en la línea 9 por la llave "`{`", que cierra en la línea 19 con la llave "`}`".

Línea 18: se realiza un nuevo ingreso para la variable `LDR`. De otra forma, si el valor ingresado inicialmente fuese 1, se produciría un bucle infinito, ya que cada vez que se verifique la condición de la variable `LDR`, su valor sería el mismo.

Línea 20: si la evaluación de la condición en la instrucción `while` es falsa, el control del programa pasa a esta línea, estableciendo la condición de `motor apagado`.

Líneas 22 a 25: representan el fin de programa.

Una vez codificado el programa, debe ser realizada la compilación de este con la `tecla F9`. Como se observa en la parte inferior de la Figura 6.65, si el programa no tiene errores, se genera el `archivo .hex` o ejecutable del mismo.

Luego, puede ser realizada la ejecución del mismo con la `tecla F10`.

```
C:\Program Files (x86)\Dev-Cpp\
1
V en divisor = 5V.
VBE supera los 0.7V.
Conmuta transistor.
VCE = 0V.
V en nodo inferior de motor = 0V.
Motor tiene diferencia de potencial.
Motor funciona.
```

Figura 6.65. Ejecución del programa para la condición LDR = 1.

```
C:\Program Files (x86)\Dev-Cpp\
0
Motor apagado.
```

Figura 6.66. Ejecución del programa para la condición LDR = 0.

CAPÍTULO 7

DISEÑO E IMPLEMENTACIÓN DE ROBOTS CON LÓGICA DE CIRCUITOS INTEGRADOS DIGITALES NO PROGRAMABLES

7.1 DISEÑO DE ROBOTS CON COMPUERTAS LÓGICAS

En robótica, dependiendo del diseño y consecuentemente de la complejidad requerida para el mismo; las compuertas lógicas pueden ser empleadas para desarrollar robots con lógica de compuertas en lugar de emplear un microcontrolador.

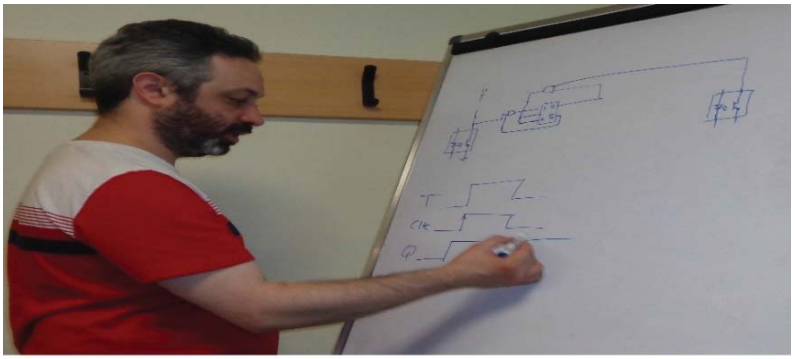


Figura 7.1. Diseño de robot con lógica de compuertas.
Imagen: Juan Pablo Bettini – socio fundador de Dedalord.

La consigna para el ejercicio de implementación de robot, empleando lógica de compuertas, fue la siguiente:

ROBÓTICA DISEÑO Y APLICACIÓN – MÓDULO III
YIYE ROCHA DIAZ

EJERCICIO DE IMPLEMENTACIÓN

CONSIGNA:

1) Implementar robot detector de línea blanca, empleando lógica adicional, pero no de microcontrolador.
2) Diseñar circuito propio.
3) Desarrollar la carpeta de proyecto, conteniendo:
   - **3.1.** Circuito eléctrico.
   - **3.2.** Descripción del principio de funcionamiento.
   - **3.3.** Justificación de las innovaciones introducidas.
   - **3.4.** BOM o Listado de Materiales.
   - **3.5.** Precio de componentes, y costo final de implementación.
4) Ecuaciones, leyes y principios aplicados.
5) Simulación del circuito.
6) Implementación física.

Figura 7.2. Consigna del ejercicio de implementación.

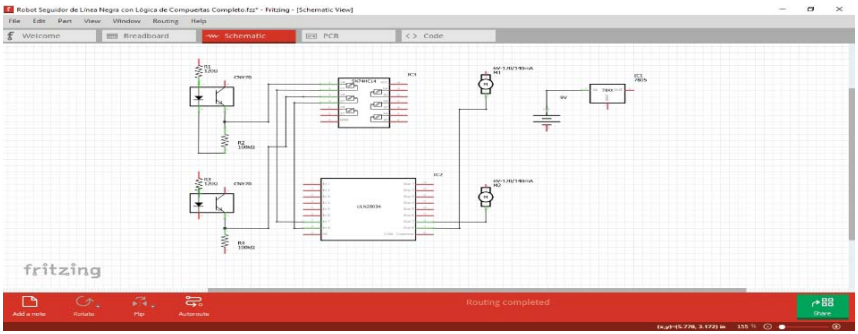


Figura 7.3. Imagen. Fase de diseño del circuito de robot con lógica de CI.

Se ve en la Figura 7.3, el inicio del montaje para el circuito de robot seguidor de línea negra con lógica de circuitos integrados digitales no programables (Figura 7.24).

7.2 COMPUERTAS LÓGICAS

Las compuertas lógicas son circuitos integrados digitales que desarrollan funciones lógico aritméticas.

Su funcionamiento está basado en el modelo matemático, desarrollado por George Boole dentro de la disciplina, que hoy se conoce como Álgebra de Boole.

El álgebra de Boole es la base de los sistemas digitales, dado que los circuitos integrados digitales empleados, por ejemplo, en los ordenadores, o inclusive en la composición interna de microcontroladores, microprocesadores, decodificadores, y multiplexores entre otros, poseen de acuerdo a su tipo y características, distintas escalas de integración de compuertas lógicas.

Las compuertas lógicas dependiendo del tipo de familia a la que pertenezcan, TTL O CMOS, emplearon históricamente tecnologías de transistores Bipolares o MOSFET, para su fabricación, pero en la actualidad se combinan dichas tecnologías, de modo de aprovechar las características de una y otra.

Las compuertas lógicas básicas, son tres:

NOT, OR, AND

Compuerta Lógica	Función Lógica Desarrollada
NOT o Inversora	Inversión o complemento (C1)
OR (O)	Suma lógica
AND (Y)	Producto lógico

Figura 7.4. Compuertas lógicas básicas y función lógica desarrollada.

De las compuertas básicas, derivan las compuertas:

NOR, NAND, XOR, XNOR

Compuerta Lógica	Función Lógica Desarrollada
NOR o OR Negada	Inversión o complemento a la salida de la función OR
NAND o AND Negada	Inversión o complemento a la salida de la función AND
XOR o OR Exclusiva	La salida vale 1, cuando exclusivamente uno de sus valores de entrada, es 1
XNOR o OR Exclusiva Negada	Inversión o complemento a la salida de la función XOR

Figura 7.5. Compuertas lógicas derivadas y función lógica desarrollada.

En robótica, dependiendo del diseño y en consecuencia de la complejidad requerida, las compuertas lógicas pueden ser empleadas para desarrollar robots con lógica adicional, en lugar de emplear microcontrolador.

Cuando se trabaja con compuertas lógicas, simplificará la labor recordar que se deben emplear siempre las cuatro propiedades que siguen:

Expresión
Símbolo lógico
Tabla de verdad
Tren de impulsos o diagrama temporal

Figura 7.6. Cuatro propiedades de las compuertas lógicas.

Descripción de las cuatro propiedades: conceptualmente, aplica a cualquier Circuito Integrado Digital.

Expresión representa la operación lógico-aritmética que desarrolla la compuerta lógica.

Símbolo lógico: es el símbolo o imagen que representa a cada compuerta.

Tabla de verdad: representación gráfica del o de los niveles lógicos de la salida de la compuerta en función de sus niveles de entrada.

Tren de impulsos o diagrama temporal: representación del comportamiento de las entradas y salida de la compuerta en función del tiempo.

7.3 DESCRIPCIÓN Y MODO DE FUNCIONAMIENTO

7.3.1 COMPUERTA INVERSORA (NOT)

La compuerta NOT, también conocida como buffer inversor, representa la función de complemento a 1, negación o inversión del valor presente en la entrada.

EXPRESIÓN

$$Y = \bar{A} = NOT\,(A)$$

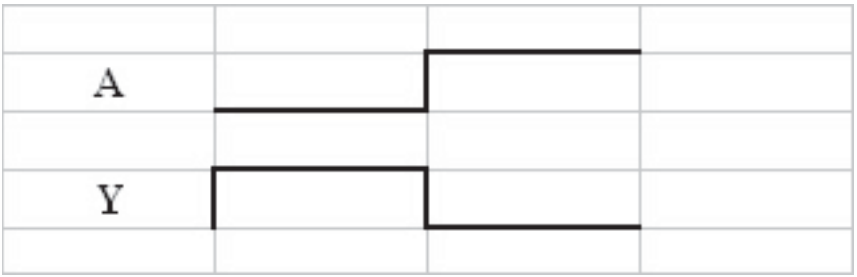
SÍMBOLO LÓGICO

TABLA DE VERDAD

$Y = \bar{A}$	
Input (Entrada)	Output (Salida)
A	Y
L	H
H	L

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



En el tren de impulsos o diagrama temporal, se observa que, en el primer instante de tiempo, cuando la entrada A está a nivel bajo (0 V), la salida Y, está a nivel lógico alto (5 V). En el segundo instante de tiempo cuando la entrada A está a nivel lógico alto (5 V), la salida Y está a nivel bajo (0 V).

7.3.2 COMPUERTA OR

La compuerta OR (O) representa la operación algebraica de suma. En esta compuerta, al tratarse de una suma, basta que una de sus entradas sea o valga 1 para que su salida esté también a nivel alto o 1 lógico (5 VCC).

EXPRESIÓN

$$Y = A + B$$

SÍMBOLO LÓGICO

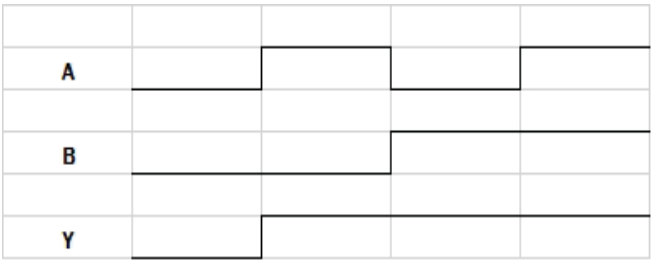
TABLA DE VERDAD

$$Y = A + B$$

Input (Entrada)	Input (Entrada)	Output (Salida)
B	A	Y
L	L	L
L	H	H
H	L	H
H	H	H

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



Tal cual se observa en el tren de impulsos, basta que una de las entradas de la compuerta valga 1 para que la salida se ponga a nivel alto.

7.3.3 COMPUERTA AND

La compuerta AND (Y) representa la operación algebraica de multiplicación. En esta compuerta, al tratarse de un producto, basta que una de sus entradas sea o valga 0 para que su salida esté también a nivel bajo o 0 lógico (0 V).

EXPRESIÓN

$$Y = A \,.\, B$$

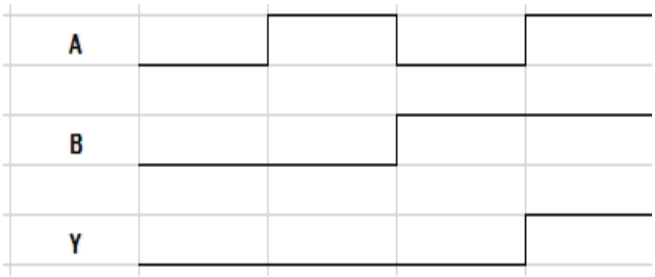
SÍMBOLO LÓGICO

TABLA DE VERDAD

$Y = A \,.\, B$		
Input (Entrada)	Input (Entrada)	Output (Salida)
B	A	Y
L	L	L
L	H	L
H	L	L
H	H	H

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



Tal cual se observa en el tren de impulsos, al tratarse de un producto, basta que una de las entradas de la compuerta lógica valga 0 para que la salida se ponga a nivel bajo.

7.3.4 COMPUERTA NOR

La compuerta NOR es la negación de la compuerta OR; por tanto, bastará que haya un 1 en alguna de las entradas para que la salida esté a nivel lógico 0 o bajo.

EXPRESIÓN

$$Y = \overline{A + B}$$

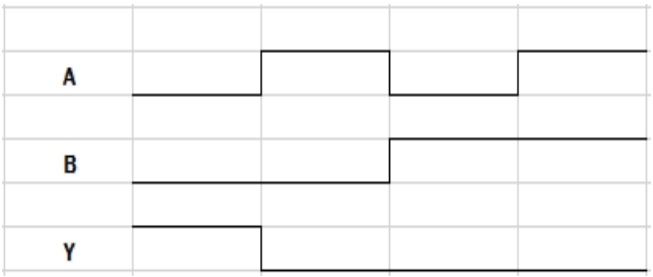
SÍMBOLO LÓGICO

TABLA DE VERDAD

$Y = \overline{A + B}$		
Input (Entrada)	Input (Entrada)	Salida (Output)
B	A	Y
L	L	H
L	H	L
H	L	L
H	H	L

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



Tal cual se observa en el tren de impulsos, basta que una de las entradas de la compuerta valga 1 para que la salida se ponga a nivel bajo, negando, de esta forma, la tabla de verdad de la compuerta OR.

7.3.5 COMPUERTA NAND

La compuerta NAND es la negación de la compuerta AND. Entonces bastará que haya un 0 en alguna de sus entradas para que la salida esté a nivel lógico alto.

EXPRESIÓN

$$Y = \overline{AB}$$

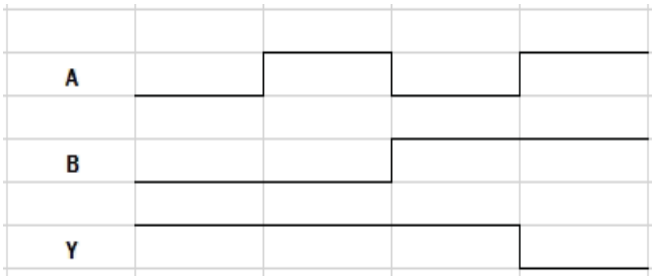
SÍMBOLO LÓGICO

TABLA DE VERDAD

$Y = \overline{AB}$		
Input (Entrada)	Input (Entrada)	Output (Salida)
B	A	Y
L	L	H
L	H	H
H	L	H
H	H	L

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



Tal cual se observa en el tren de impulsos, basta que una de las entradas de la compuerta valga 0 para que la salida se ponga a nivel lógico alto, negando, de esta forma, la tabla de verdad de la compuerta AND.

7.3.6 COMPUERTA XOR

La compuerta XOR tendrá un nivel lógico alto a la salida cuando exclusivamente una de sus entradas esté a nivel alto.

EXPRESIÓN

$$Y = A \oplus B$$

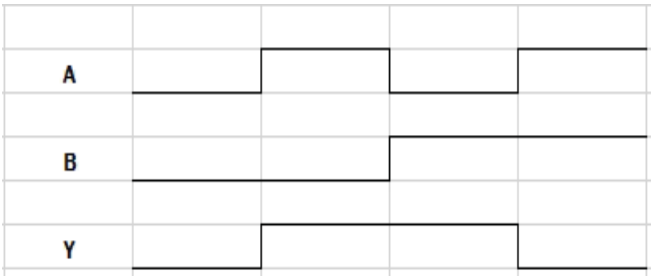
SÍMBOLO LÓGICO

TABLA DE VERDAD

$Y = A \oplus B$		
Input (Entrada)	Input (Entrada)	Output (Salida)
B	A	Y
L	L	L
L	H	H
H	L	H
H	H	L

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



Tal cual se observa en el tren de impulsos, la salida está ahora a nivel lógico alto cuando exclusivamente una de sus entradas es o vale 1.

7.3.7 COMPUERTA XNOR

La compuerta XNOR niega la tabla de verdad de la compuerta XOR, así que su salida tendrá un nivel lógico alto cuando no exista exclusividad en sus valores de entrada.

EXPRESIÓN

$$Y = A \overline{\oplus} B$$

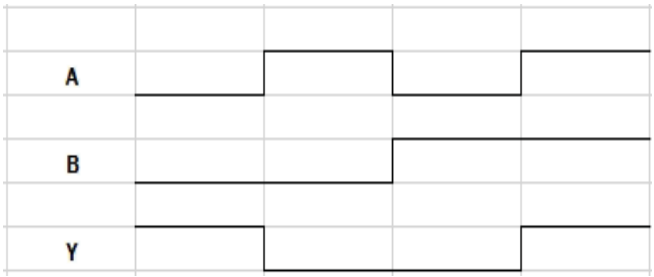
SÍMBOLO LÓGICO

TABLA DE VERDAD

$Y = A \overline{\oplus} B$		
Input (Entrada)	Input (Entrada)	Output (Salida)
B	A	Y
L	L	H
L	H	L
H	L	L
H	H	H

H = HIGH Logic Level (H = Nivel Lógico ALTO)
L = LOW Logic Level (L = Nivel Lógico BAJO)

TREN DE IMPULSOS



Tal cual se observa en el tren de impulsos, la salida está ahora a nivel lógico alto, cuando no existe la condición de exclusividad en sus niveles o valores de entrada.

7.4 CIRCUITOS INTEGRADOS

Un circuito integrado es:

Un circuito electrónico, ensamblado en un encapsulado de múltiples pines, integrado en una o más láminas semiconductoras, y con diversas cantidades de entradas y salidas, de acuerdo con la función para la cual ha sido desarrollado.

7.4.1 CLASIFICACIÓN

De acuerdo con el tipo de señal a procesar, los CI pueden ser clasificados en los siguientes tipos:

TIPO	EJEMPLO
Lineales (analógicos)	Amplificador Operacional Timer 555
Digitales	Compuerta Lógica Microcontrolador
Mixtos	Conversor Analógico Digital (ADC) Conversor Digital Analógico (DAC)

Figura 7.7. Clasificación de los circuitos integrados.

7.4.2 TECNOLOGÍA TTL

TTL es un acrónimo del inglés Transistor-Transistor-Logic o Lógica Transistor-Transistor. Hace referencia a su tecnología de fabricación con transistores bipolares.

Los CI TTL se identifican con la denominación de serie "74", y entre sus características más importantes destaca su reducido rango de tensión, comprendido entre los 0 y 5 VCC.

Un detalle importante para trabajar en este punto es considerar que los valores de tensión. No se asocian a los niveles lógicos de forma exacta e inmediata; esto significa que el 0 lógico, por ejemplo, no coincide únicamente con los 0 V al igual que ocurre con el nivel alto, o 1 lógico, el que no se detecta o produce exclusivamente en 5 V, dado que como observaremos en la siguiente tabla, existe un nivel mínimo y máximo de tensión, para indicar ambos estados.

ESPECIFICACIONES PARA LOS NIVELES DE TENSIÓN DE ENTRADA

VIL = Voltaje de Entrada de Nivel Bajo – indica el nivel de tensión a partir del cual, comienza el nivel lógico 0 o estado bajo
VIH = Voltaje de Entrada de Nivel Alto – indica el nivel de tensión a partir del cual, comienza el nivel lógico 1 o estado alto

Figura 7.8. Especificación para las tensiones de entrada.

ESPECIFICACIONES PARA LOS NIVELES DE TENSIÓN DE SALIDA

VOL = Voltaje de Salida de Nivel Bajo – representa el valor de tensión a partir del cual, tenemos un nivel lógico 0 o estado bajo
VOH = Voltaje de Salida de Nivel Alto – representa el valor de tensión a partir del cual, tenemos un nivel lógico 1 o estado alto

Figura 7.9. Especificación para las tensiones de salida.

En función de los datos trabajados en las figuras 7.8 y 7.9, se debe tener en cuenta entonces, que, para los valores reales, los rangos de tensiones de los niveles bajo y alto tienen un valor mínimo y un valor máximo, pero existe un rango intermedio, el cual se conoce como zona indeterminada. En esta franja, el fabricante no asegura ningún nivel lógico, y debe ser evitada.

En la figura siguiente se describen los niveles lógicos y de tensión, que emplea la tecnología TTL.

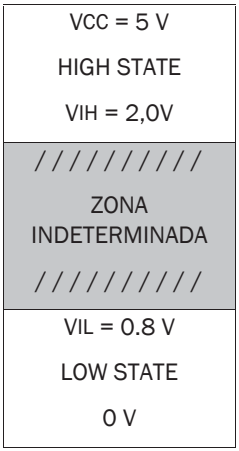


Figura 7.10. Niveles TTL estándar.

Otro dato necesario en el trabajo con compuertas lógicas, son los valores máximos y mínimos de las corrientes, en función de su operación como entrada o salida. Tal cual se representa en la siguiente tabla.

ESPECIFICACIONES PARA LOS NIVELES DE CORRIENTE
IIL = Corriente de Entrada de Nivel Bajo – indica el valor de corriente máxima que el dispositivo puede absorber, cuando recibe un nivel lógico 0
IIH= Corriente de Entrada de Nivel Alto – indica el valor de corriente máxima que el dispositivo suministra, cuando recibe un nivel lógico 1
IOL = Corriente de Salida de Nivel Bajo – representa el valor de corriente máxima que el dispositivo puede absorber, cuando se encuentra en nivel bajo
IOH = Corriente de Salida de Nivel Alto – representa el valor de la corriente de salida que puede suministrar el dispositivo, cuando se encuentra en el nivel lógico 1

Figura 7.11. Especificaciones de corriente para las entradas y salidas.

Es frecuente encontrar en la etapa de diseño de robots y dispositivos resultados indeseados o daño en los circuitos integrados, ante el desconocimiento de las especificaciones anteriores.

Vale destacar la diferencia entre las aplicaciones para las que muchos de estos dispositivos fueron concebidos inicialmente en los sistemas digitales, y nuestra aplicación en el ámbito de la robótica. No habrá desarrollo de ítems como el *fan out* de 10, o cantidad de dispositivos que se pueden cargar a la salida de una

compuerta lógica, ya que, en una instancia de diseño, se emplearía un microcontrolador, o una FPGA, en lugar de emplear tantos CI de forma discreta.

Para el CI DM74LS08, que integra cuatro compuertas AND de dos entradas, los valores de tensiones y corrientes tratados en las figuras 7.8 y 7.9 de acuerdo con su *data sheet* u hoja de datos, son los siguientes:

DATA SHEET CI DM74LS08 – ESPECIFICACIONES DE V - ENTRADA				
Symbol	Parameter	Min	Max	Units
VIL	LOW Level Input Voltage		0.8	V
VIH	HIGH Level Input Voltage		2	V

Figura 7.12. Data Sheet CI DM74LS08. tensiones de entrada

.

DATA SHEET CI DM74LS08 – ESPECIFICACIONES DE V - SALIDA						
Symbol	Parameter	Conditions	Min	Typ (Note 2)	Max	Units
VOL	LOW Level Output Voltage	VCC = Min, IOL = Max, VIL = Max		0.35	0.5	V
		IOL = 4 mA, VCC = Min		0.25	0.4	V
VOH	HIGH Level Output Voltage	VCC = Min, IOH = Max,VIH = Min	2.7	3.4		V

Figura 7.13. Data Sheet CI DM74LS08 – Tensiones de salida.

Note: All typicals are at VCC = 5V, TA = 25°C.

DATA SHEET CI DM74LS08 – ESPECIFICACIONES DE I - SALIDA					
Symbol	Parameter	Conditions	Min	Max	Units
IOL	LOW Level Output Current			8	mA
IOH	HIGH Level Output Current			-0.4	mA

Figura 7.14. Data Sheet CI DM74LS08. Corrientes de salida.

DATA SHEET CI DM74LS08 – ESPECIFICACIONES DE I - ENTRADA					
Symbol	Parameter	Conditions	Min	Max	Units
IIL	LOW Level Input Current	VCC = Max, VI = 0.4V		-0.36	mA
IIH	HIGH Level Input Current	VCC = Max, VI = 2.7V		20	µA

Figura 7.15. Data Sheet CI DM74LS08. Corrientes de entrada.

Los circuitos integrados en tecnología TTL para las compuertas lógicas desarrolladas en el presente capítulo, son los siguientes:

CI SERIE	ESPECIFICACIÓN
DM74LS04	Hex Inverting Gates (Seis Compuertas Inversoras)
DM74LS32	Quad 2-Input OR Gate (Cuádruple 2 entradas Compuerta OR)
DM74LS08	Quad 2-Input AND Gate (Cuádruple 2 entradas Compuerta AND)
DM74LS02	Quad 2-Input NOR Gate (Cuádruple 2 entradas Compuerta NOR)
DM74LS00	Quad 2-Input NAND Gate (Cuádruple 2 entradas Compuerta NAND)
DM74LS86	Quad 2-Input Exclusive-OR Gate (Cuádruple 2 entradas Compuerta OR Exclusiva)
DM74LS266	Quad 2-Input Exclusive NOR Gate With Open-Collector Outputs. (Cuádruple 2 entradas Compuerta NOR Exclusiva Con Salidas de Colector Abierto)

Figura 7.16. Serie y especificación de las compuertas lógicas desarrolladas en el apartado 7.3.

7.4.3 TECNOLOGÍA CMOS

Para el caso de la tecnología CMOS, su denominación es un acrónimo de *Complementary* MOSFET o *Complementary Metal-Oxide Semiconductor*.

En esta tecnología, se emplean dos tipos de transistores en el circuito de salida: uno de canal n, y otro de canal p tipo MOSFET. Tengamos presente también que un transistor tipo p es complementario de uno tipo n y viceversa.

La denominación para la familia de CI CMOS es 4000, y entre sus características principales, se encuentran:

- Tensión de alimentación de rango amplio
- Menor consumo de I
- Alta inmunidad al ruido
- En sus inicios, la velocidad de conmutación entre niveles lógicos era lenta

Figura 7.17. Características de la tecnología CMOS.

Para la tecnología CMOS, el equivalente del CI TTL DM74LS08, trabajado antes, es el CI CD4081B, el cual integra cuatro compuertas AND de dos entradas.

Las especificaciones de tensión y corriente son similares a las representadas para TTL, aunque adaptadas a las características CMOS.

Entre ellas, destaca:

Condiciones de Operación Recomendadas.

CHARACTERISTIC	LIMITS		UNITS
	MIN.	MAX.	
Supply-Voltage Range (For TA = Full Package Temperature Range)	3	18	V

Figura 7.18. Rangos de tensión para la tecnología CMOS.

Se observa en la Figura 7.17 una de las características de la tecnología CMOS, presentadas en la Figura 7.18, respecto a **tensión de alimentación de rango amplio**.

7.5 CIRCUITO INTEGRADO 40106BE

Un claro ejemplo de la importancia de los CI CMOS en Robótica es el CI 40106BE, que provee seis inversores Schmitt Trigger.

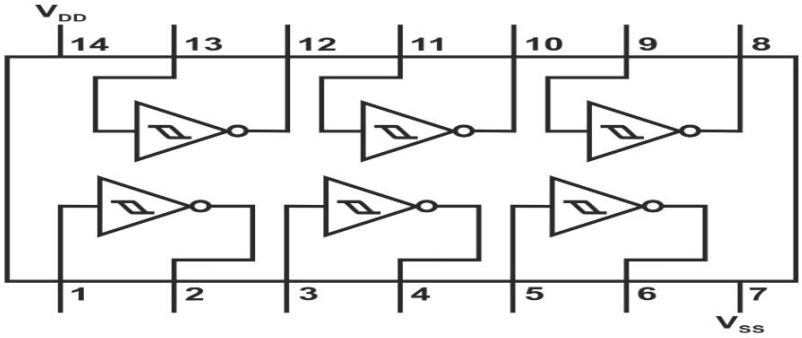


Figura 7.19. *Pin Out* o descripción de pines del CI Hex Schmitt Trigger.

Este CI resulta imprescindible, dado que, de forma clara y práctica, **permite asegurar los niveles lógicos 0 o 1**. De esta manera evita que el circuito del robot se vea afectado por niveles de tensión intermedios, inestables, o por transiciones de estados lentas.

Su principio de funcionamiento está basado en un circuito muy empleado en la electrónica: **el comparador** implementado con Amplificadores Operacionales.

Otro principio aquí presente, y con aplicación en los sistemas digitales secuenciales, es el de realimentación.

Los amplificadores operacionales (AO) fueron presentados en la Figura 7.7 dentro del tipo de CI lineales (analógicos).

Obsérvese de la Figura 7.19 pin out del CI 40106BE, que el símbolo distintivo para la tecnología Schmitt Trigger es el correspondiente a la representación gráfica del ciclo o proceso de histéresis.

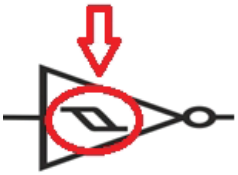


Figura 7.20. Símbolo distintivo para la tecnología Schmitt Trigger.

Al diseñar Robótica con este circuito integrado, se deberá tener presente su característica de inversor de la señal de entrada.

Esta característica, a su vez, puede ser intencionalmente explotada en la fase de diseño de robots. Su propósito es invertir en un diseño ya desarrollado, la lectura de una línea negra en lugar de blanca, o viceversa. De este mismo modo será trabajado para la implementación en el presente capítulo del robot seguidor de línea negra, con lógica de circuitos integrados digitales no programables.

Algunas características de este CI son:

- Ancho rango de voltaje de fuente: 3 V a 15 V.
- Alta inmunidad al ruido.
- Compatibilidad con tecnología TTL.

7.6 CIRCUITO INTEGRADO ULN2803

Este circuito integrado provee ocho pares de transistores de tipo NPN en configuración Darlington. Cada canal posee incorporado un diodo de *clamping* para la protección en la conmutación de cargas inductivas.

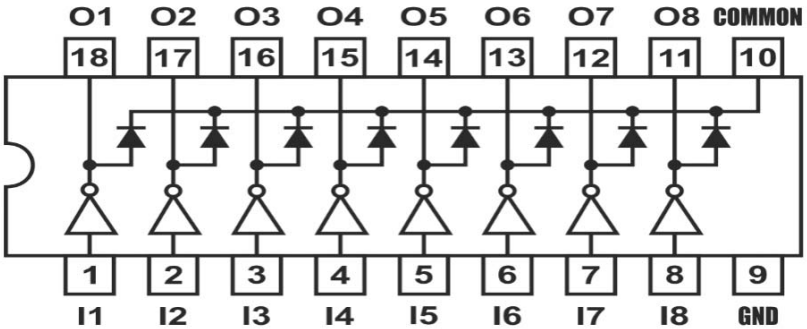


Figura 7.21. *Pin Out* o descripción de pines del CI ULN2803.

Recuérdese que en el transistor Darlington, el Beta o la ganancia de corriente está determinado por el producto de la ganancia de corriente de cada transistor.

Entonces el empleo de este circuito integrado está destinado como *driver* o amplificador de corriente.

Como se verá en el Capítulo 8, cada pin de un microcontrolador puede entregar una corriente máxima de 0.025 A ó 25 mA, por lo que se requerirá el empleo de un transistor de forma discreta, o en un circuito integrado, para amplificar la corriente requerida por los motores.

7.7 ROBOT SEGUIDOR DE LÍNEA NEGRA CON LÓGICA DE CIRCUITOS INTEGRADOS DIGITALES NO PROGRAMABLES

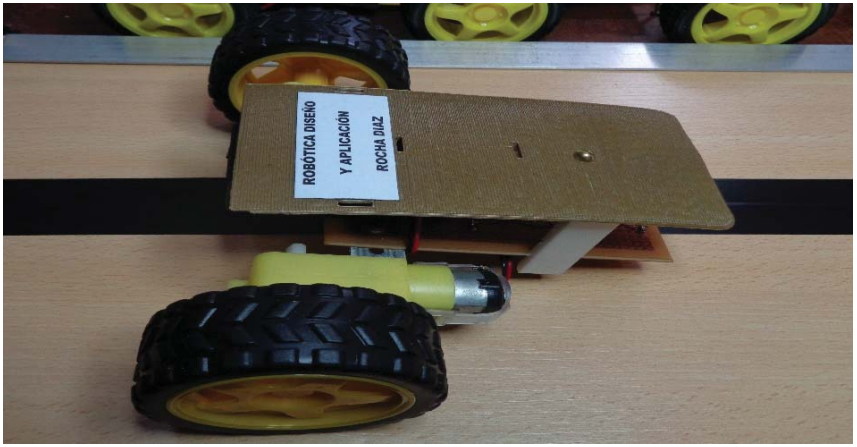


Figura 7.22. Robot seguidor de línea negra con lógica de CI digitales no programables.

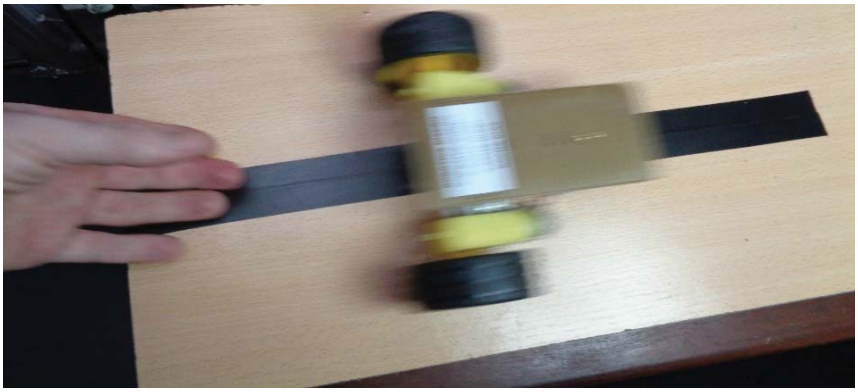


Figura 7.23. Robot seguidor de línea negra con lógica de CI en funcionamiento.

7.7.1 INTRODUCCIÓN

Se propone ahora la integración de los conocimientos desarrollados en el presente capítulo. El propósito es implementar un robot seguidor de línea negra con lógica de circuitos integrados digitales no programables. Como se verá, la base del circuito es la empleada en el apartado 5.3 para el robot seguidor de línea blanca sin lógica adicional. Sin embargo, los transistores en configuración Darlington serán reemplazados por su equivalente en circuito integrado (CI). Se emplea, además, el CI inversor para la lectura de línea negra en lugar de blanca y adecuación de los niveles lógicos.

7.7.2 CIRCUITO ELÉCTRICO

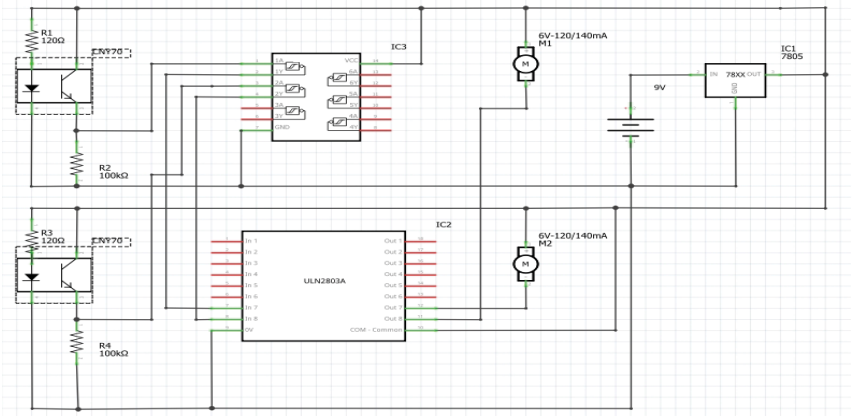


Figura 7.24. Circuito eléctrico. Robot seguidor de línea negra con lógica de CI.

El circuito eléctrico del robot seguidor de línea negra con lógica de circuitos integrados digitales no programables (Figura 7.24) ha sido diseñado empleando Fritzing.

7.7.3 PRINCIPIO DE FUNCIONAMIENTO

Como fue descrito en el comienzo de la obra, la etapa de entrada del circuito es similar a la empleada para el robot seguidor de línea blanca, (Figura 5.22).

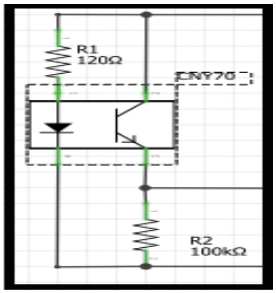


Figura 7.25. Etapa de entrada similar a la del Robot Seguidor de Línea Blanca.

Recuérdese por tanto del apartado 5.3.2:

En la etapa de entrada del sistema, se emplea el sensor CNY 70. Recuérdese este es un sensor óptico reflectivo u optoacoplador, es decir, acopla por luz un diodo infrarrojo (IR), con la base de un fototransistor. Los fototransistores y el sensor han sido presentados en el Capítulo 3.

Obsérvese en el circuito que el ánodo (+) del diodo IR está conectado en la rama de la resistencia de 120 Ω, al polo positivo de la fuente de tensión (5 VCC desde la salida del regulador 7805). Luego, el diodo IR emite luz; y si encuentra una superficie clara, el haz de luz producido rebota y excita la base del fototransistor.

El fototransistor tiene el emisor conectado a una resistencia de *pull down* fuerte de 100 KΩ. El colector está conectado al positivo de la fuente o 5 VCC.

El nodo (unión) entre el emisor y el colector ha sido ahora conectado desde el emisor, a la entrada del CI 40106BE. Como se ha descrito, este cumple una doble función: por un lado, invierte el nivel lógico o la tensión de entrada con la finalidad de leer la línea negra en lugar de la blanca. Por el otro, asegura los niveles lógicos para marcar la correcta lectura de un 0 o 1 lógicos.

Mientras se cumpla la condición de que la luz no reflecte en una superficie clara, la base del fototransistor no permitirá la conmutación del estado bajo de su emisor, al potencial de 5 VCC presente en el colector. De esta forma, la entrada (sea la referida al canal izquierdo o derecho del robot) del CI40106BE recibe un

estado lógico bajo y lo invierte. De este modo, entrega un nivel de salida alto al CI *driver* ULN2803.

Seguidamente, la señal de salida alta del CI inversor que ingresa al canal (elegido) del CI *driver* ULN2803 es amplificada en corriente y nuevamente invertida. Así entrega un nivel lógico o de tensión bajo a la salida.

El motor tiene su terminal inferior conectado a esta salida a nivel bajo del CI ULN2803, y el otro extremo o terminal conectado al terminal positivo de la fuente. Por lo tanto, se enciende y conduce.

Recuérdese que no es necesario el empleo del diodo de *clamp* o *clamping*, en paralelo con el motor, ya que es provisto en el CI ULN2803.

Cuando el diodo IR emite luz y ella encuentra una superficie clara, se propaga. Por lo tanto, la base del fototransistor permite la conmutación entre el terminal de emisor y el de colector.

Como el colector se encuentra conectado al terminal positivo de la fuente o 5 VCC, esta será ahora la tensión en el nodo entre colector y emisor. Así este nivel lógico alto ingresa al CI Inversor Schmitt Trigger 40106BE, el cual invierte su nivel alto de entrada y entrega un nivel bajo a la entrada del CI ULN2803. Éste vuelve a invertir la tensión de entrada, polarizando el extremo inferior del motor con un nivel eléctrico alto. Y como su terminal superior se encuentra conectado al terminal positivo de la batería, no existe diferencia de potencial. Es así que el motor estará en posición de apagado.

7.7.4 LISTADO DE MATERIALES O BOM

COMPONENTE	CANTIDAD	DESCRIPCIÓN
Placa	1	5 x 10 cm. Experimental o multipropósito. Pertinax o Epoxi
Batería de 9V	1	Recargable. 250 mAh. Ni.Mh. Estándar
Conector de Batería	1	Conector Broche
Gabinete para batería	1	Plástico o 3D. Dimensiones de la batería
Regulador de V	1	7805 - Regulador de tensión Positiva
Disipador de calor	1	Para TO220. Con tornillo y tuerca
Motores	2	6V - 120/140 mA – con reductor
Ruedas	2	Diámetro de 65 mm. con encastre para motor

Sensores	2	CNY70 (TFK 082 o TFK 102)
Alojamientos Hembra	5	Por dos pines. Se emplean para zócalos de sensores, conexión de motores y batería
CI 40106BE	1	Inversor Schmitt Trigger
CI ULN2803	1	8 canales – Darlington – Driver
Zócalo para CI	1	DIP. 2 x 7 pines
Zócalo para CI	1	DIP. 2 x 9 pines
Resistencias	2	120 Ω
Resistencias	2	100 KΩ
Pines rectos a 90º	1	Por dos pines. Para conexión de batería
Esquinero Ángulo	2	25 x 25 mm
Tornillos (sujeción de motor a ángulo)	2	Planos. 31 mm de largo x 3 mm diámetro Se requieren también tuercas y arandelas
Tornillos (sujeción de ángulo a la placa)	2	Planos. 15 mm de largo x 4 mm diámetro Se requieren también tuercas y arandelas
Tornillo (sujeción de gabinete para batería)	1	Plano. 12 mm de largo x 3 mm diámetro Se requieren también tuercas y arandelas
Cables	-	Varios. *wire wapping* o cable 0.125

Es imprescindible recordar el empleo de grasa siliconada entre el CI Regulador y el disipador de calor.

7.7.5 MONTAJE Y PASOS DE IMPLEMENTACIÓN

Los pasos propuestos para la implementación del robot seguidor de línea negra con lógica de circuitos integrados digitales no programables, es similar a los empleados para la implementación de los robots desarrollados en los capítulos anteriores.

Recuérdese:

- Analizar la disposición de motores, ángulos, batería y componentes.
- Marcar con lápiz los espacios a ocupar por motores, ángulos y batería.
- Realizar el montaje de los componentes de adelante hacia atrás.

- Calcular el largo de cable necesario para la unión de los componentes.
- Salvo distancias muy cortas, no emplear pistas de cobre.
- Medir continuidad y analizar el montaje a medida que avanza.

Importante:

- No realizar soldaduras o manipulación de componentes con el circuito energizado.
- No se deben montar los CI directamente a la placa. Por lo que es requerido el empleo de zócalos para su montaje.
- No se deben realizar soldaduras con los CI en los zócalos.
- No someter los componentes a temperaturas excesivas o prolongadas de soldado.
- Emplear siempre protección ocular.

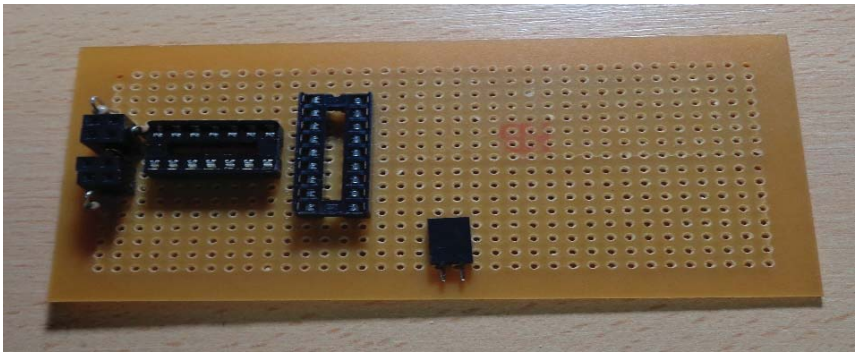


Figura 7.26. Inicio del montaje del robot seguidor de línea negra con CI.

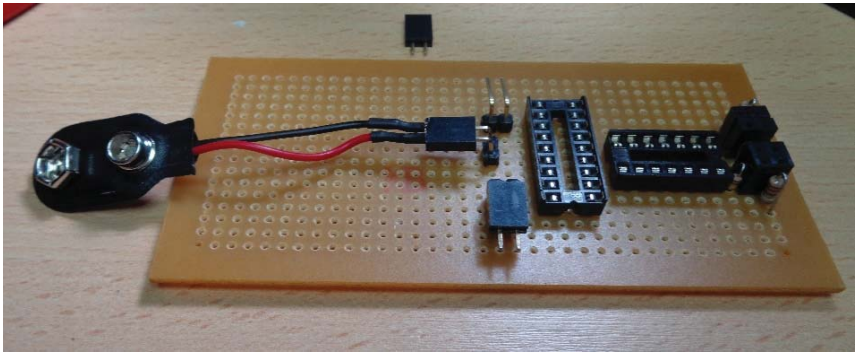


Figura 7.27. Fase de montaje del robot.

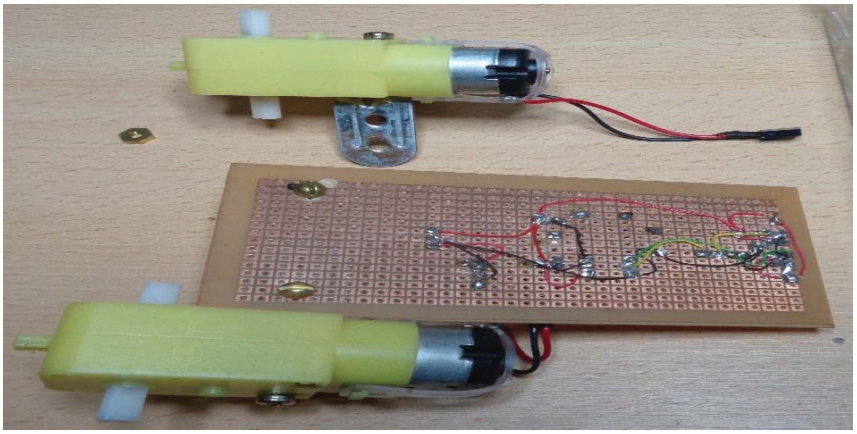


Figura 7.28. Fase de cableado y montaje de motores del robot.

Figura 7.29. Adaptación del gabinete plástico para la batería.

Para la implementación de este modelo de robot, serán empleadas en el montaje final algunas de las piezas presentadas en el Capítulo 11. Modeladas empleando Impresión 3D.

Las piezas son aplicadas para la personalización del chasis del robot.

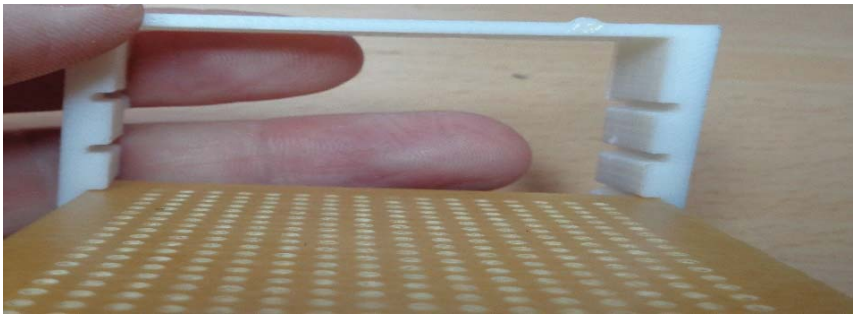


Figura 7.30. Prueba de encastre de accesorios a la placa.

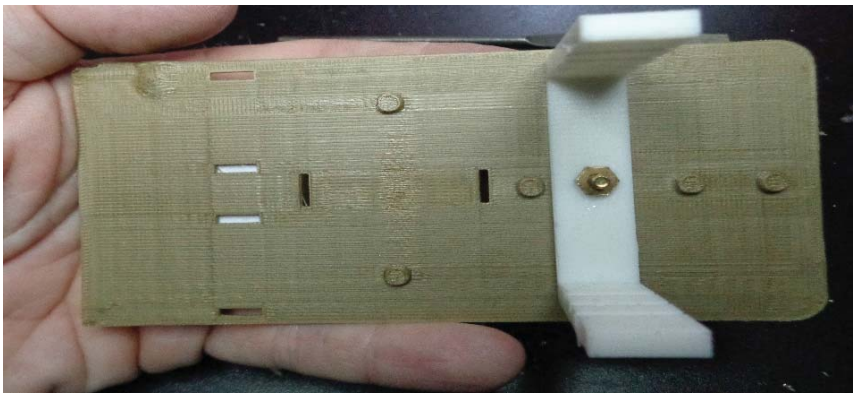


Figura 7.31. Encastre de accesorios para el chasis.

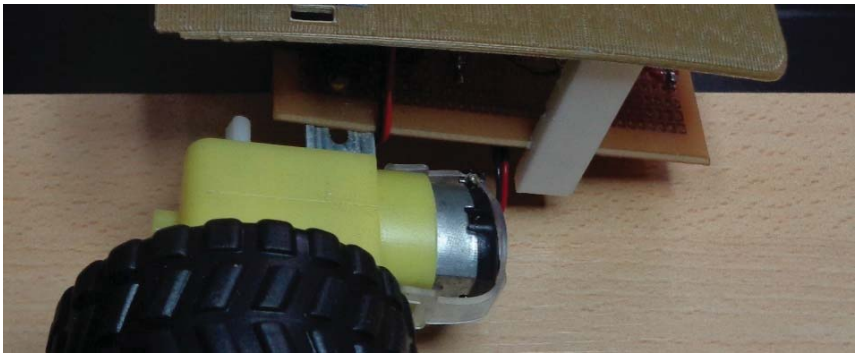


Figura 7.32. Montaje de accesorios al chasis.

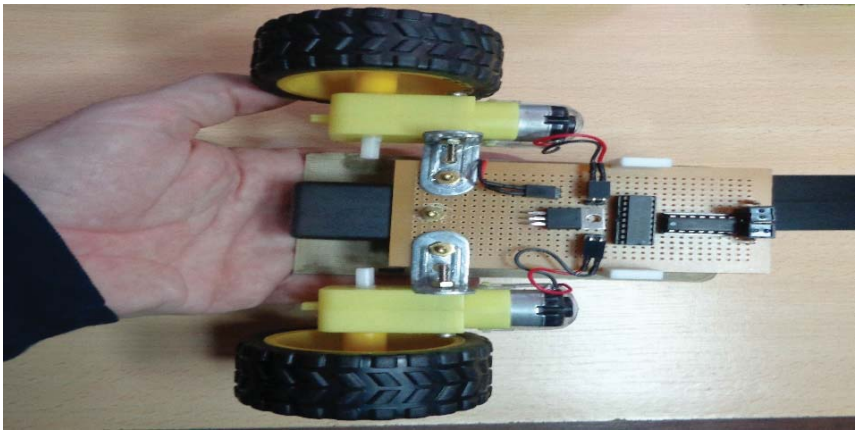


Figura 7.33. Vista de la imagen final del montaje del circuito.

Es fundamental para usos prolongados, el montaje del disipador de calor con la grasa siliconada para el CI Regulador 7805.

7.7.6 TESTEO Y PRUEBAS DE FUNCIONAMIENTO

Recuérdese, como buena práctica de testeo del funcionamiento, los pasos descritos en el Apartado 7.7.5 en el que se destaca:

Medir continuidad y analizar el montaje a medida que avanza.

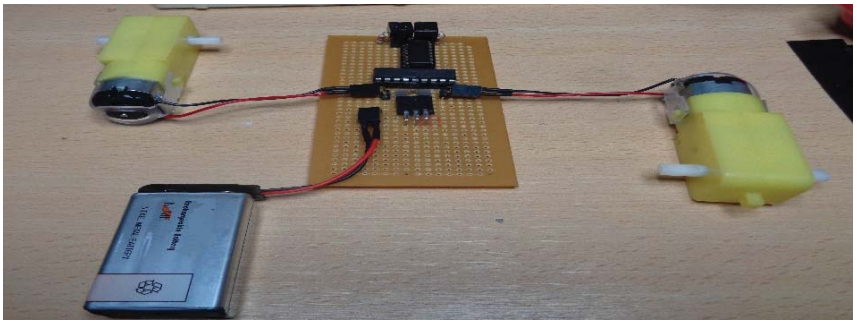


Figura 7.34. Prueba inicial de motores, CI y señales del circuito.

Es una buena práctica la generación de instancias de pruebas, similares a aquellas para las que los robots han sido desarrollados.

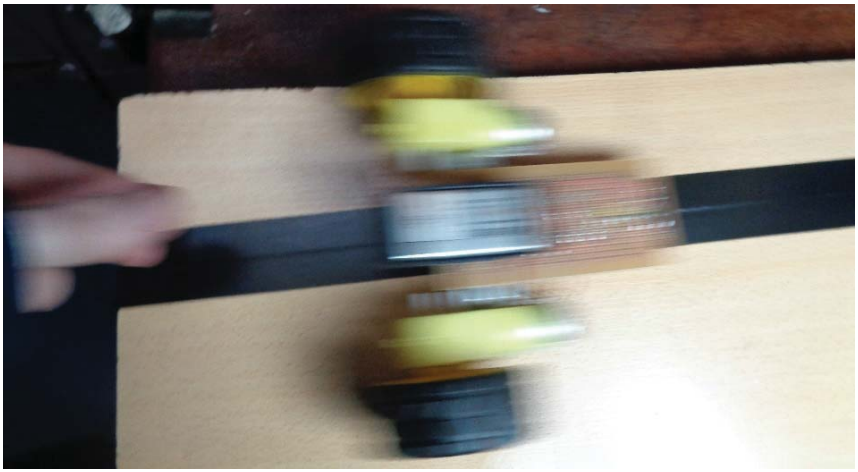


Figura 7.35. Prueba de funcionamiento del robot siguiendo la línea negra.

Figura 7.36. Al encontrar una superficie clara, el robot se detiene.

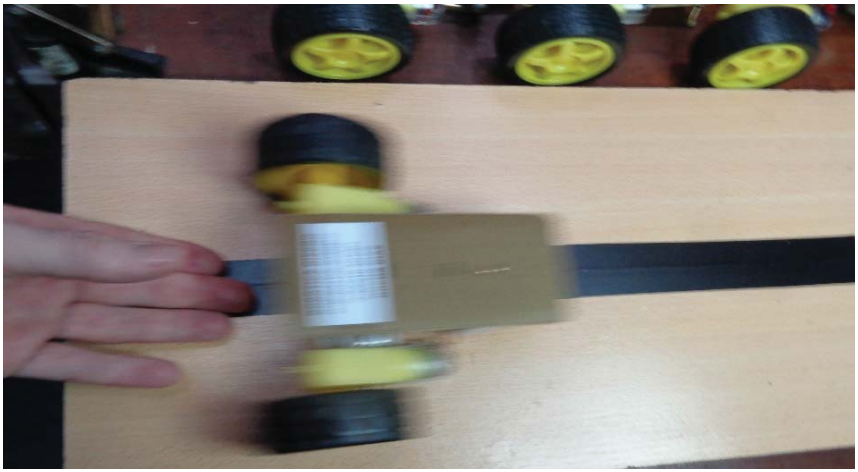


Figura 7.37. Prueba final del robot seguidor de línea con lógica de CI digitales.

Figura 7.38. Vista final de la implementación.

Figura 7.39. GRA. Enzo Ferrari, Yiye Rocha Diaz, David Balog, Verónica Chiez.

Figura 7.40. Enzo Ferrari.

Figura 7.40 implementación de robot seguidor de línea negra con lógica de compuertas lógicas.

CAPÍTULO 8
MICROCONTROLADORES Y SU APLICACIÓN EN ROBÓTICA

8.1 INTRODUCCIÓN A LOS MICROCONTROLADORES

Un microcontrolador es un circuito integrado digital programable.

En el Capítulo anterior se trabajó con circuitos integrados digitales no programables, tal fue el caso de las compuertas lógicas. Pero en la actualidad resulta muy difícil encontrar algún dispositivo electrónico de uso masivo, artefacto hogareño como los robots aspiradora, o maquinaria agropecuaria e industrial de alta tecnología, que no posea en su circuitería un microcontrolador.

Es así que cabe también la pregunta: ¿pero qué función cumplen los microcontroladores o circuitos integrados digitales programables?

La respuesta reside en su nombre mismo: micro refiere a una tecnología en sí misma, que produce integrados de forma microscópica con alta densidad de integración. Controlador aplica a que son dispositivos diseñados para control. De esta manera, se conjuga, su denominación de microcontrolador.

Se puede también realizar la analogía entre microcontrolador y sistema de control, de la siguiente forma.

SISTEMA DE CONTROL

Un sistema de control está formado por tres etapas básicas:

Figura 8.1. Microcontroladores. Sistema de Control. Etapas.

Estas tres etapas son también las que cumple el microcontrolador en su fase de control. Recibe señales (entrada), las procesa, y, en función del programa residente en memoria, provee un nivel lógico alto o bajo a la salida.

En el trabajo con microcontroladores, se deben integrar también los contenidos desarrollados en el Capítulo 6 (Introducción a la programación en Robótica), a fin de generar algoritmos eficientes para controlar sus aplicaciones de Robótica.

Los lenguajes de programación empleados en microcontroladores son Assembler y C.

8.2 MICROCONTROLADORES Y MICROPROCESADORES

En las actividades cotidianas, es habitual el empleo de la palabra microprocesador, ya sea para la elección de una PC, un móvil, o en los requerimientos particulares de actividades como, *gamer*, u oficinista.

Un Microprocesador es un circuito integrado digital, que interactúa en su trabajo con un Sistema Operativo, o conjunto de programas. Además, requiere para su interface y trabajo de los componentes montados en una placa madre o *motherboard*.

Un Microcontrolador, en cambio, tiene integrados todos los elementos de un ordenador, CPU, memorias, puertos. Conforma de esta forma un ordenador en sí mismo. También requiere, para su funcionamiento, del programa por usted desarrollado y cargado en memoria.

8.3 ARQUITECTURAS DE HARVARD Y VON NEUMANN

Se debe tener en cuenta ahora, dado el nivel de complejidad que los microcontroladores presentan para su abordaje inicial (el cual suele requerir capítulos y libros en sí mismo), que esta obra ha sido desarrollada con la finalidad

de acceder a estos conocimientos, aplicándolos al desarrollo en robótica. Se tratan específicamente características como: set de instrucción a emplear, velocidades de procesamiento, formas de acceder a la información. Estás características están determinadas por el tipo de arquitectura del dispositivo a emplear.

En Microcontroladores existe una arquitectura clásica, que lleva el nombre del matemático John Von Neumann, a quien se estudiará también en el Capítulo 11 (Máquinas Autorreplicantes de Von Neumann).

La arquitectura de un microcontrolador representa cómo está implementado internamente: sus operaciones, la unidad de proceso central y, por ende, las instrucciones que soporta.

La forma en que la estructura está ordenada (desarrollada), determinará también la forma en que se accede a los datos, el direccionamiento directo, indirecto, e indexado. Luego, por supuesto, la cantidad de operaciones y los ciclos de máquina empleados son una referencia directa de la velocidad del micro.

En la tradicional arquitectura de Von Neumann, se emplea una sola memoria, tanto para el almacenamiento de datos como de instrucciones del programa. Esto significa que las extracciones de estos datos no pueden ocurrir al mismo tiempo. Produciendo un sistema más lento.

Pero, por otra parte, esta arquitectura posee el beneficio de simplificar la lógica del microcontrolador, dada la existencia de un bus único,

La arquitectura de Harvard, es empleada, por ejemplo, por el fabricante de microcontroladores Microchip, mundialmente conocido por sus líneas PIC, y actual propietario de la compañía de microcontroladores ATMEL. Vale destacar que están presentes, por ejemplo, en las placas Arduino, ver Capítulo 10.

Los microcontroladores PIC, tal el caso del 16F84A, al poseer arquitectura de Harvard, tienen como característica la de disponer de dos memorias independientes, que pueden ser conectadas a buses separados.

En la Figura 8.2, se observará que el separar los buses, ofrece como beneficio de que la longitud de la memoria de datos y la memoria de programa pueden ser diferentes.

8.4 ARQUITECTURA INTERNA

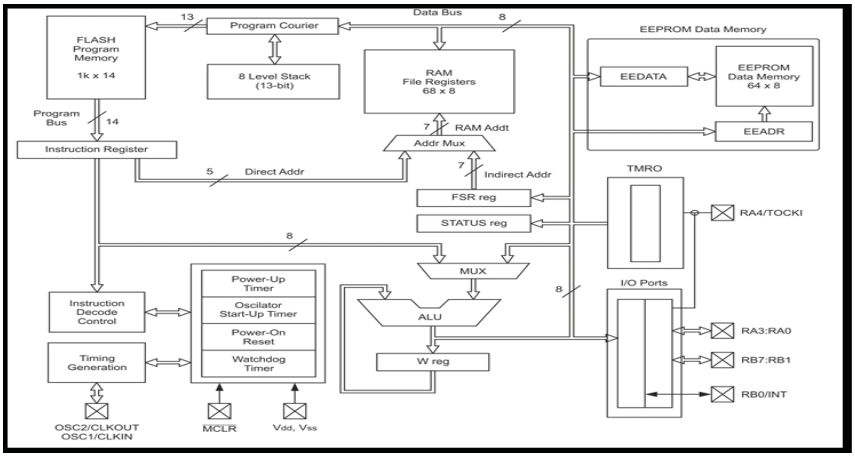


Figura 8.2. Diagrama en bloque del microcontrolador 16F84A.

Microcontrolador puede ser abreviado como µc.

Se puede destacar de la arquitectura interna del µc 16F84A:

2 puertos de entrada y salida: ports A y B. De 5 y 8 pines respectivamente
ALU de 8 bits
Memoria de programa tipo Flash de 1 K x 14 bits
Memoria de datos dividida en: • RAM: 15SFR y 68 registros de propósito general • EEPROM de datos: 64 registros de 8 bits

Figura 8.3. µc 16F84A. Componentes de su Arquitectura Interna.

8.5 SET DE INSTRUCCIONES CISC & RISC

En el apartado 6.9.1 (Set de instrucciones), se vio que el set de instrucciones será dependiente de la arquitectura (CPU).

Se presentarán a continuación las características de los sets de instrucciones CISC y RISC, en función de su arquitectura.

SET DE INSTRUCCIONES	ARQUITECTURA ASOCIADA
CISC	Von Neumann
RISC	Harvard

Figura 8.4. Set de Instrucciones. CISC & RISC. Arquitectura de origen.

8.5.1 SET DE INSTRUCCIONES CISC

Tradicionalmente, la arquitectura de Von Neumann empleaba set de instrucción CISC (Complex Instruction Set Computer) Computador de Set de Instrucciones Complejo. Poseen de 200 a 300 instrucciones dependiendo del microcontrolador.

Las instrucciones más complejas requieren muchos ciclos de reloj (clk), para ser ejecutadas.

8.5.2 SET DE INSTRUCCIONES RISC

Los microcontroladores de arquitectura Harvard emplean *set* de instrucciones RISC (Reduced Instruction Set Computer) Computador de *Set* de Instrucciones Reducido.

Los *sets* de instrucciones RISC emplean un número reducido de instrucciones. El 16F84A, por ejemplo, emplea únicamente 35 instrucciones.

La arquitectura RISC pretende set de instrucciones reducidas y simples, que empleen menor tiempo de ejecución.

Salvo las instrucciones de salto (*branch*), las instrucciones se ejecutan en un ciclo de máquina.

8.6 CICLOS DE MÁQUINA VS CICLOS DE CLOCK

La relación entre ciclos de máquina y ciclos de reloj es la siguiente: cada instrucción tarda en ejecutarse 4 ciclos de reloj (*clock* o clk).

1 CICLO MÁQUINA = 4 CICLOS DE RELOJ

Ejemplo: para un microcontrolador 16F84A con un cristal (oscilador externo) de 4Mhz, y aplicando que el período es la inversa de la frecuencia:

$$T = 1/f$$

Se debe aplicar que el tiempo que tarda un microcontrolador en ejecutar una instrucción viene determinado por la siguiente fórmula:

$$\mathrm{T} = 4\ (1/\mathrm{f})\ \mathrm{cm}$$

Donde cm = ciclos de máquina.

8.7 PIN OUT

Se vio en el apartado 3.2 (Tecnologías de componentes THT y SMD), que THT (*Through Hole Technology*) representa Tecnología de Agujeros Pasantes. A su vez, el PIN OUT de un microcontrolador da referencia tanto de la distribución de sus pines en el CI, como así también de la función que cumple cada pata o pin.

Los microcontroladores tienen la ventaja de poseer una amplia gama de opciones de encapsulado, provistos por los fabricantes para hacer eficiente su implementación en función del espacio disponible y el coste de producción.

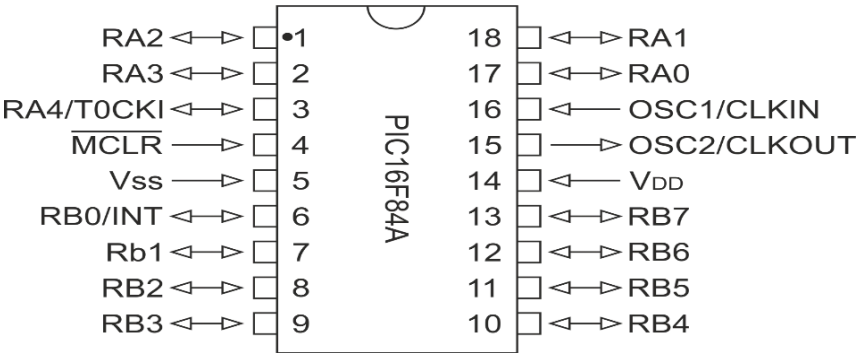


Figura 8.5. *Pin Out* del Microcontrolador 16F84A.

8.8 PUERTOS O PORTS

De acuerdo con la gama elegida, se dispondrá de una cantidad de puertos o *ports* que posea el microcontrolador, y asimismo de la cantidad de pines o patas asociadas a dichos puertos.

Los puertos, son por lo general, agrupaciones de 8 pines.

1 Puerto o *Port* = 8 pines = 1 byte o 8 bits.

Al igual que ocurre en los ordenadores, los puertos de un microcontrolador cumplen la función de conectarlo con el mundo exterior.

El microcontrolador 16F84A posee 2 puertos, denominados A y B:

PORT A = 5 pines (RA0-RA4).
PORT B = 8 pines (RB0-RB7).

Figura 8.6. Puertos A y B. Pines y distribución.

Los pines de cada puerto pueden ser configurados independientemente como entradas o salidas.

Si se configura:

0 = pin configurado como salida.
1 = pin configurado como entrada.

Figura 8.7. Configuración de pines como entradas o salidas.

8.9 SFR – REGISTROS DE FUNCIONES ESPECIALES

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power - on RESET
Bank 0										
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								- - - - - - - -
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	- - -x xxxx
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx
07h	—	Unimplemented location, read as '0'								—
08h	EEDATA	EEPROM Data Register								xxxx xxxx
09h	EEADR	EEPROM Address Register								xxxx xxxx
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					- - -0 0000
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x
Bank 1										
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								- - - - - - - -
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000
83h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx
85h	TRISA	—	—	—	PORTA Data Direction Register					- - - 1 1111
86h	TRISB	PORTB Data Direction Register								1111 1111
87h	—	Unimplemented location, read as '0'								—
88h	EECON1	—	—	—	EEIF	WRERR	WREEN	WR	RD	- - -0 x000
89h	EECON2	EEPROM Control Register 2 (not a physical register)								- - - - - - - -
8Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					- - -0 0000
8Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x

Figura 8.8. SFR o registros de funciones especiales del µc 16F84A.

Al comienzo, puede parecer que el mapa de la Figura 8.8 es incomprensible, pero como se verá en los ejemplos, su aplicación es simple y fundamental para realizar operaciones, y configuración de puertos y pines como entradas o salidas,

8.10 TIPOS DE MEMORIAS

Para el Microcontrolador 16F84A, se observa en la Figura 8.2 que se distinguen tres tipos (bloques) de memoria.

MEMORIA DE PROGRAMA
Memoria de Programa Flash (*Flash Program Memory*), 1024 palabras (1 K) de 14 bits Esta memoria es de tipo no volátil. Esto significa que el programa se mantiene, aunque se corte la alimentación

Figura 8.9. µc 16F84A. Memoria de Programa.

MEMORIA DE DATOS RAM
La memoria de datos dispone de dos zonas diferentes:
AREA DE **RAM ESTATICA (SRAM)** **22 REGISTROS DE FUNCIONES ESPECIALES (SFR)**. En realidad, son 24 posiciones de memoria, dos de las cuales no se encuentran implementadas
68 REGISTROS DE PROPOSITOS GENERALES (GPR)

Figura 8.10. µc 16F84A Memoria de Datos RAM.

MEMORIA EEPROM
Esta área o zona de memoria posee 64 registros de 8 bits Puede ser empleada opcionalmente y es no volátil

Figura 8.11. µc 16F84A Memoria EEPROM.

Una característica muy importante para los microcontroladores de Microchip es que poseen la memoria de datos, zona de memoria RAM. Dividida en lo que se conoce como bancos. El µc16F84A posee dos bancos de memoria: el Banco 0 (Bank 0) y el Banco 1 (Bank 1).

8.10.1 MAPA DE MEMORIA

Figura 8.12. µc 16F84A. Mapa de Memoria.

8.11 ELECCIÓN DEL MICROCONTROLADOR

Se propuso emplear el Microcontrolador 16F84A para desarrollar la programación en lenguaje Assembler, dada su popularidad, la simpleza de su arquitectura, y la cantidad de librerías y aplicaciones desarrolladas para este dispositivo, que aún están vigentes. Sin embargo, es importante también destacar que la elección de este Microcontrolador parte de la formación pedagógica del autor, quien entiende que no es aconsejable el inicio en esta área, empleando un microcontrolador que posee cinco bancos.

Por otra parte, cabe destacar que, independientemente de las decisiones comerciales de los fabricantes o la actualización de sus modelos, ocurre con el 16F84A lo que ha ocurrido con las líneas 6800 de Motorola (Freescale) como el clásico HC05 el J1 o la línea 8051 de Intel: que fueron empleadas para formar a los grandes soportes de esta industria durante décadas.

Lo mismo ocurre con la elección del microcontrolador 18F4610 de Microchip para la programación de microcontroladores en lenguaje C.

Luego, será importante analizar las características a nivel de hardware del Microcontrolador a emplear, en función del proyecto. El 16F84A, por ejemplo, no posee ADC (analog-to-digital converter) o conversores analógicos a digital incorporados, que son muy útiles en las aplicaciones de Robótica. De este modo, debe ser reconsiderada su elección, ante la necesidad de empleo para aplicaciones más avanzadas.

Una opción respecto a coste, tecnología o tamaño es el empleo del Microcontrolador 16LF1825:

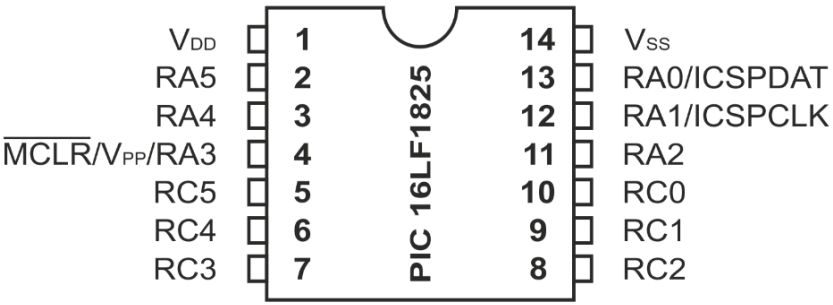


Figura 8.13. µc 16LF1825. Pin Out.

Recursos:

Microcontrolador 16LF1825

Set de Instrucciones	RISC CPU - 49 instrucciones
Pines	14
ADC – 10 bits de resolución	12 canales
CCP Módulo	2 módulos PWM de captura/comparación
ACM	Módulo Comparador Analógico
High current sink/source	25 mA/25 mA
2 MSSP, con SPI, I²C	Master Synchronous Serial Port
Módulo EUSART	Enhanced Universal Synchronous Asynchronous receiver transmitter
ICSP	In-Circuit Serial Programming
ICD	In-Circuit Debug

Figura 8.14. µc 16LF1825. Características.

Debe tenerse en cuenta que, como ha sido presentado en el apartado 2.22 (Niveles lógicos y niveles de tensión), el lenguaje que manejan o entienden los microprocesadores y microcontroladores es el formado por 0s y 1s o lenguaje de máquina. De esta forma, contando con el compilador necesario, cualquier microcontrolador puede ser empleado para trabajar en lenguaje Ensamblador o C, ya que posteriormente este código de instrucciones será transferido al lenguaje de máquina correspondiente.

Cabe, además, integrar el ítem anterior desarrollado en el Capítulo 2 junto a señales digitales y la importancia del CI 40106BE visto en el Capítulo 7, dado el comportamiento real de las señales digitales.

Por convención, en los sistemas digitales se emplea lógica positiva, pero se debe destacar la existencia del empleo de lógica negativa.

En lógica positiva, el 1 lógico está asociado al nivel alto o 5 V; y el 0 lógico está asociado al nivel bajo o 0 V.

En lógica negativa, el 0 lógico está asociado al nivel alto o 5 V; y el 1 Lógico está asociado al nivel bajo o 0 V.

Ambas convenciones son representadas en la figura siguiente:

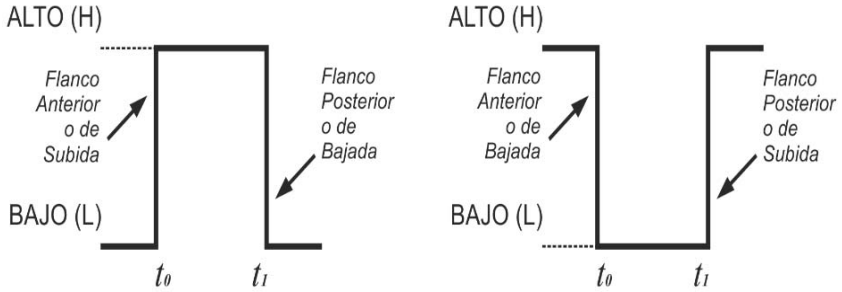


Figura 8.15. Convenciones de lógica positiva y lógica negativa.

Se observa en la Figura 8.15 la convención para lógica positiva (imagen izquierda). Y la convención para lógica negativa (imagen derecha).

En la siguiente figura, se representa el modelo real (no ideal) de una señal cuadrada, y su descripción. Recuérdese que los microcontroladores son dispositivos sincrónicos, por lo que el tiempo posee un rol fundamental.

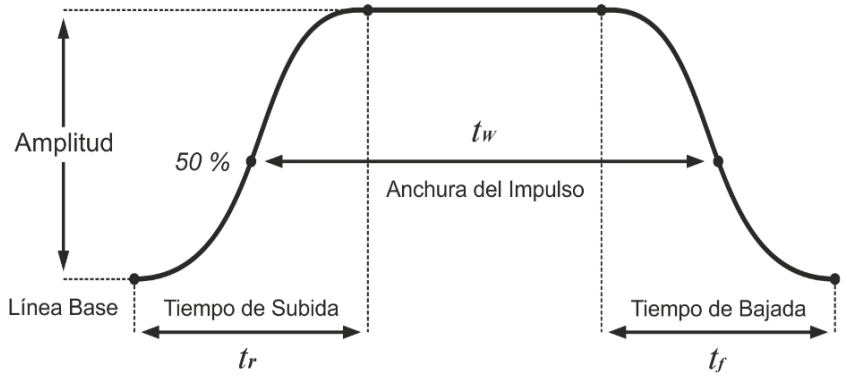


Figura 8.16. Modelo real de una señal cuadrada.

Siguiendo con la descripción de los microcontroladores a emplear en esta obra, se representa en la siguiente figura el *pin out* o diagrama de pines del Microcontrolador 18F4610:

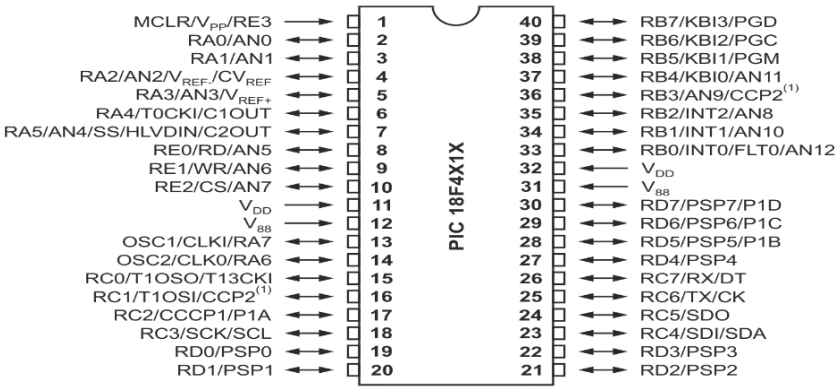


Figura 8.17. *Pin out* o diagrama de pines del Microcontrolador 18F4610.

Puede observarse en la imagen que el empleo de la X en la denominación sobre el cuerpo del circuito integrado hace referencia a que la descripción aplica a toda la familia de dispositivos, es decir: 18F4410, 18F4510, 18F4610.

La diferencia entre dispositivos dentro de una misma familia puede responder a la ampliación de las memorias de dato y programa, o a la incorporación de recursos a nivel de hardware. Estos afectan también a los SFR o Registros de Funciones Especiales.

Entre las características de este microcontrolador, se encuentran:

MEMORIA DE PROGRAMA
Memoria de Programa Flash (*Flash Program Memory*), 64K Esta memoria es de tipo no volátil. Esto significa que el programa se mantiene, aunque se corte la alimentación

Figura 8.18. µc 18F4610. Memoria de Programa.

La capacidad de Memoria de Programa Flash que posee este Microcontrolador da referencia de su potencial. Esta característica puede ser comparada con la presentada en la Figura 8.9. para el µc 16F84A, cuya Memoria de Programa flash es de 1K.

Microcontrolador 18F4610	
Set de Instrucciones	RISC CPU - 75 instrucciones
Set extendido	Incorporan 8 nuevas instrucciones
Arquitectura RISC	Optimizada para compilar en C
RAM	3968 bytes

Puertos	5. A, B, C, D, E
Pines	40
Pines de entrada/salida	36 pines configurables como I/O
Pines de alimentación	4: dos para VDD y dos para VSS
PWM	5 salidas
ADC – 10 bits de resolución	13 canales de entrada
Módulo Comparador Analógico	2 comparadores analógicos
CCP Módulo	Módulo PWM de captura y comparación
Alta corriente sink/source	25 mA/25 mA
ICSP	In-Circuit Serial Programming
ICD	In-Circuit Debug

Figura 8.19. µc 18F4610. Características.

8.12 HERRAMIENTAS DE TRABAJO

Para trabajar con microcontroladores aplicado a la Robótica, se debe contar con las siguientes herramientas de trabajo:

HERRAMIENTAS PARA TRABAJAR CON MICROCONTROLADORES
1 - Elección del Microcontrolador a emplear
2 - IDE MPLAB, MPLABX
3 - Programador PICkit 3
4 - Placa o Circuito a controlar

Figura 8.20. Microcontroladores. Herramientas de trabajo.

8.13 PROGRAMACIÓN. PASOS A SEGUIR

PROGRAMACIÓN DEL MICROCONTROLADOR - PASOS A SEGUIR
1 – IDE: Inicio
2 - Desarrollo de código de programa
3 - Compilación
4 - Enlace del Programador con el IDE
5 - Limpiar memoria
6 - Descargar código
7 - Ejecutar aplicación

Figura 8.21. Programación del µc. Pasos a seguir.

Las herramientas listadas en la Figura 8.20, así como los pasos a seguir de la Figura 8.21, serán profundizados y aplicados en el Capítulo 9, en la fase de implementación de robots con programación de microcontroladores en lenguajes Assembler y C.

8.14 PROGRAMADOR - PICKIT 3 DE MICROCHIP

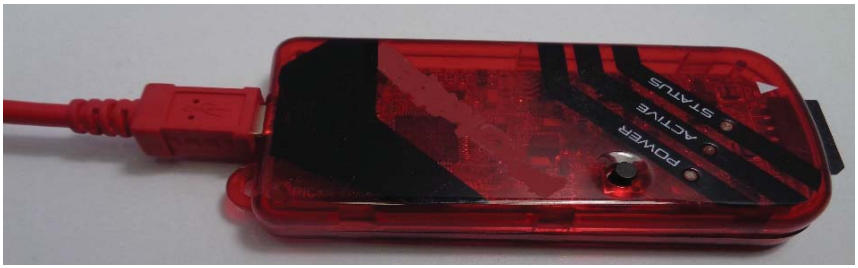


Figura 8.22. Programador PICkit3.

El PICkit3 es un depurador y programador en circuito de la Empresa Microchip.

Una opción interesante para las aplicaciones de Robótica es la de Debugger-System – Device With On-Board ICE Circuitry (Sistema Depurador – de Dispositivo en la Placa de Circuito, con Emulador En Circuito ICE). Este permite *debugear* y programar el microcontrolador desde la placa del robot, sin necesidad de desarme o extracción del micro.

8.14.1 ICSP – IN CIRCUIT SERIAL PROGRAMING

La opción de Programación Serie En Circuito es altamente empleada en la actualidad. Se verá en el Capítulo 10 que la tecnología Arduino integra también este sistema.

Su beneficio reside en que permite descargar el archivo `.hex` al microcontrolador, sin necesidad de desconectarlo de la placa que controla.

Para los casos de empleo del programador PICkit3, el ICSP permite la opción *debugger* en circuito. Esto significa correr el programa desde el IDE, línea a línea, para ver su efecto en la aplicación o placa.

La descripción del conector de pines de salida del PICkit 3 es la siguiente:

Pin	Signal
1	MCLR / Vpp
2	Vdd Target
3	Vss Ground
4	PGD (ICSPDAT)
5	PGC (ICSPCLK)

Figura 8.23. PICkit 3 Connector Pinout.

El pin 6 PGM (LVP) no es implementado.

8.14.2 IMPLEMENTACIÓN DE LA ICSP EN ROBÓTICA

La descripción de pines de la Figura 8.23 es empleada en la implementación física para proveer la interface entre el microcontrolador y la placa del robot.

En la imagen siguiente (Figura 8.24) se muestra su aplicación en la fase de programación del robot seguidor de luz con lógica de microcontrolador (se va a implementar en el Capítulo 9).

Puede observarse en la imagen que la programación del microcontrolador se realiza sin necesidad de extraer el dispositivo de la placa que controla. Este sistema, además de ser empleado en los sistemas Arduino (Capítulo 10), es similar al que se emplea para el control y actualización de software en los coches.

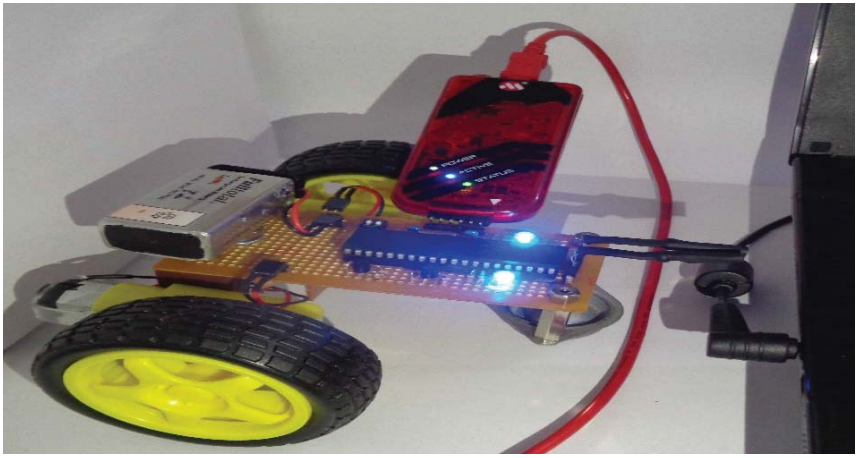


Figura 8.24. Implementación de circuito ICSP en placa de robot. Se emplea el µc18F4610.

8.15 IMPLEMENTACIÓN DE PLACA DE ENTRENAMIENTO

Desde mi experiencia profesional al haber desarrollado el primer trayecto de **Formación de Programadores de Microcontroladores para la** Industria en la rama de Formación Profesional en la Provincia de Buenos Aires, conozco y entiendo la dificultad inicial que puede presentar esta actividad. Entonces, una buena herramienta para su ejercitación es generar herramientas propias de práctica y/o entrenamiento.

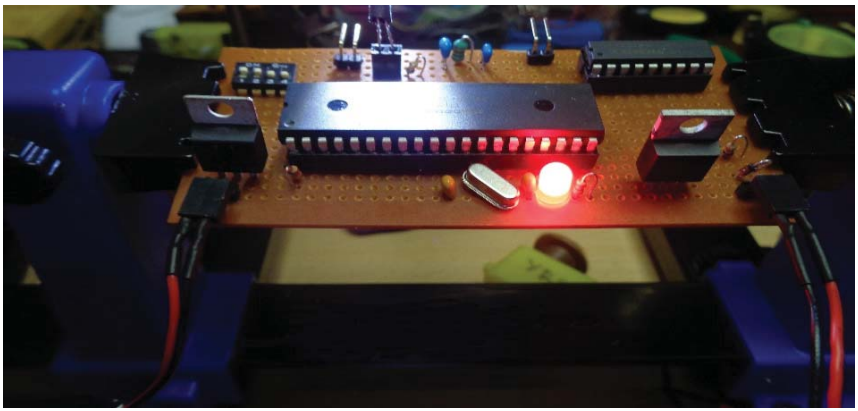


Figura 8.25a. Placa de entrenamiento para prototipos de robots con µc.

Puede resultar extraño que en el trabajo con microcontroladores la actividad consiste en leer y escribir 0s y 1s. Es por esto que se ha presentado en el apartado 2.20, Sistema de numeración binario.

En los apartados 8.7 y 8.8 *pin out* y puertos o ports, se resume la clave de esta actividad, ya que los pines y puertos del microcontrolador serán la interface para la entrada y salida de niveles lógicos o señales eléctricas.

Un componente o dispositivo, que ya ha sido presentado y puede ser de utilidad para la ejercitación e implementación inicial, es el diodo LED. Esto se debe a que sus estados ON/OFF (prendido/apagado) describen también los niveles lógicos 0 y 1.

Nuevamente, se puede emplear el circuito presentado en la Figura 2.26.(resistencia en serie con el diodo LED). Sin embargo, destaca ahora la salida del pin del microcontrolador como la fuente que ha de entregar los 5V, y una corriente máxima de 25 mA.

Figura 8.25b. ya vista en Figura 2.26. Resistencia en serie con el diodo LED.

En la imagen siguiente (Figura 8.26), se muestra el circuito de configuración básica para un Microcontrolador 16F84A, con diodo LED y resistencia en serie conectados a la salida del pin RA0.

Al igual que en el Capítulo 2, el circuito ha sido implementado con la versión demostración de Proteus.

Se observa en la imagen que resulta fundamental el empleo de un oscilador o cristal con sus respectivos capacitores para la configuración básica de un microcontrolador. Otro detalle importante es recordar que, en muchos casos, los simuladores no emplean los pines de alimentación de los circuitos integrados. Es así que no aparecen los pines de alimentación 5 (-) y 14 (+) del microcontrolador.

Otra herramienta de este potente software de simulación es la posibilidad de cargar el código de programa y ejecutar la simulación. Esto permite verificar el comportamiento, en estado ideal, de la aplicación a implementar.

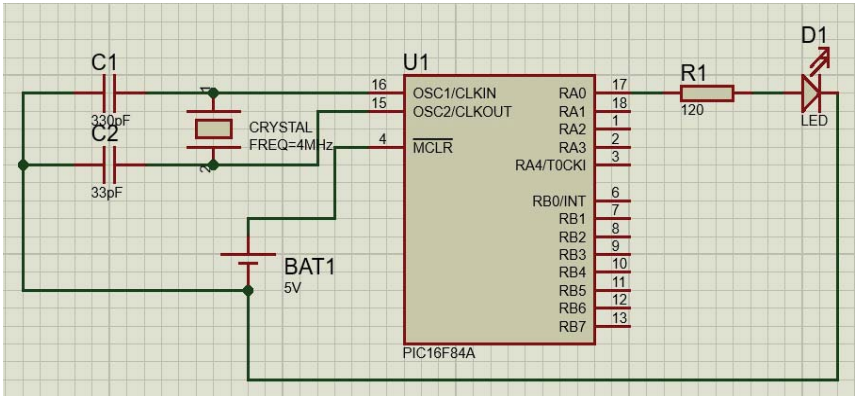


Figura 8.26. Circuito básico para ejercitación inicial con el µc 16F84A.

8.15.1 DESARROLLO DE LA APLICACIÓN

Siguiendo los pasos presentados en el apartado 6.10.2 (IDE MPLAB – Assembler), así como el ejemplo trabajado en la Figura 6.38, se propone integrar

el apartado 6.11.5 (Aplicación de desarrollo de algoritmos) para integrarlos a la aplicación propuesta en la Figura 8.26.

Consigna:

Implementar una aplicación con Microcontrolador 16F84A, que permita encender un diodo LED conectado al pin RA0.

Etapas o pasos:

1ª DESARROLLO DE DIAGRAMA
2ª PRUEBA DE ESCRITORIO
3ª CODIFICACIÓN

1ª Etapa:

DESARROLLO DE DNS (Diagrama de Nassi-Shneiderman)

Inicio
Inclusión de archivo: HEAD.inc
Inicio en posición 0 de memoria de programa
Seteo del bit 5 del registro estado: para cambiar al banco 1
Carga en el registro W: el valor necesario para configurar el pin 0 del puerto A como salida
Copia del valor del registro W en el registro de configuración del puerto A
Se limpia el bit 5 del registro de estado: para volver al banco 0
Seteo del bit 0 del puerto A para encender el LED
Fin

Figura 8.27. Desarrollo de DNS.

2ª Etapa:

PE (Prueba de Escritorio)

Instrucción	BANCO	Pin RA0
Inicio		
Inclusión de archivo: Head.inc	0	
Inicio en posición 0 de memoria de programa	0	
Seteo del bit 5 del registro estado: para cambiar al banco 1	0	
Carga en el registro W: el valor necesario para configurar el pin 0 del puerto A como salida	1	

Copia del valor del registro W en el registro de configuración del puerto A	1	
Se limpia el bit 5 del registro estado: para volver al banco 0	1	
Seteo del bit 0 del puerto A para encender el LED	0	1
Fin		

Figura 8.28. PE verificación del algoritmo desarrollado en DNS.

3ª Etapa:

CODIFICACIÓN

Una vez graficado el algoritmo, mediante el diagrama de Nassi-Shneiderman, y verificado mediante la prueba de escritorio, se procede a su codificación. Para el ejemplo, se emplea el lenguaje Assembler.

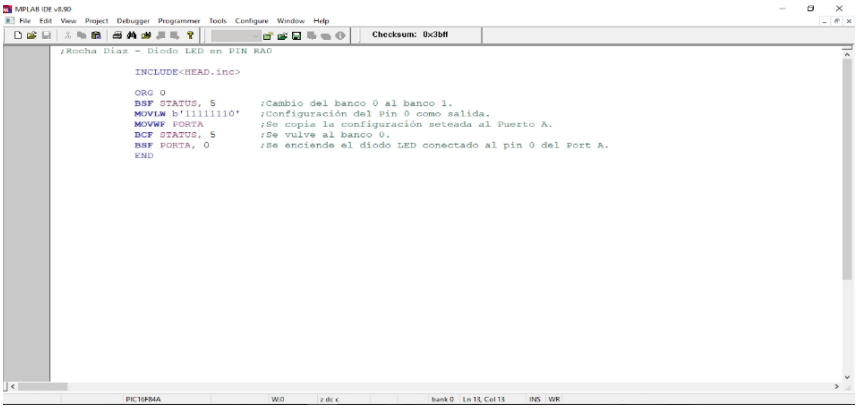


Figura 8.29. Código de programa en lenguaje Assembler. IDE MPLAB.

```
;Rocha Diaz - Diodo LED en PIN RA0

INCLUDE<HEAD.inc>

ORG 0
BSF STATUS, 5        ;Cambio del banco 0 al banco 1.
MOVLW b'11111110'    ;Configuración del Pin 0 como salida.
MOVWF PORTA          ;Se copia la configuración seteada al Puerto A.
BCF STATUS, 5        ;Se vulve al banco 0.
BSF PORTA, 0         ;Se enciende el diodo LED conectado al pin 0 del Port A.
END
```

Figura 8.30. Vista ampliada del código de programa.

En el Capítulo 9 se explicará la implementación de dos robots seguidores de luz. Para uno se empleará un microcontrolador programado en lenguaje C. Para el otro se empleará un microcontrolador programado en lenguaje Assembler. Y también se explicarán paso a paso tanto los detalles de implementación del hardware como los del software.

8.15.2 EXPLICACIÓN DE LA SINTAXIS DEL PROGRAMA

Para entender la sintaxis de los programas al trabajar con microcontroladores en lenguaje Ensamblador, debe desarrollarse el concepto sobre **bancos** de un microcontrolador. Este fue presentado al cierre del apartado 8.10 (Tipos de memorias).

Los bancos representan posiciones o bloques de memoria. El 16F84A, posee 2 bancos de trabajo: el banco 0 y el banco 1.

La forma más simple de entender su funcionamiento es recordar que cuando el microcontrolador está corriendo un programa, o procesando señales (eventos), se encuentra trabajando en el banco 0. Pero para configurar sus pines o puertos como entradas o salidas, es necesario acceder al banco 1.

El cambio de banco se produce seteando (se pone a '1') el `bit RP0`, correspondiente al `bit 5` del `registro STATUS` o `ESTADO`, cuya dirección o posición de memoria, corresponde a la `03h` (Figura 8.8 `SFR o Registros de Funciones Especiales` del µc 16F84A).

La h que acompaña a la posición de memoria 03 hace referencia al sistema de numeración hexadecimal. La base de este sistema es 16, emplea los dígitos 0 al 9, y las letras A, B, C, D, E, F, hasta expresar el equivalente decimal del número 15. De esta forma, se emplea un solo bit para representar 16 elementos o el equivalente a 2 bytes.

Una vez configurados los puertos con el valor requerido para la aplicación, se debe retornar al `banco 0`. Esto se produce de forma inversa a la empleada para el cambio de estado. Es decir, el `bit RP0` debe ser reseteado o puesto a '0', de esta forma, se ha retornado al `banco 0` para hacer correr el programa de control.

La línea:

`INCLUDE <HEAD.inc>` fue presentada en la Figura 6.39, donde se detalla su contenido.

```
ORG 0
```

Indica el inicio de programa en la posición cero de la memoria de programa.

```
BSF STATUS,5
```

La instrucción `BSF` o `Bit Set File`, significa Setear un Bit de un Registro. Se debe indicar, seguidamente, el archivo o registro en cuestión. En este caso, corresponde el Registro de Estado, el bit a setear es el 5, correspondiente, como se vio, al `bit RP0`, necesario para cambiar al `banco 0`.

```
MOVLW b'11111110'
```

Significa Copiar o Mover (del término inglés *move*): Literalmente, en el `registro W`, el valor binario correspondiente.

Recuérdese que, para configurar un pin como salida, debe ser seleccionado en 0 (Figura 8.7).

El `registro W` (del inglés *work*), es un registro de 8 bits empleado como `acumulador`, y a través del cual los datos son transferidos de un registro a otro.

Así, queda configurado el `bit 0` como salida (necesario para prender el diodo LED).

```
MOVWF PORTA
```

Esta instrucción significa Copiar o Mover el contenido del `Registro W` en un Registro o Archivo. Debe ser indicado el registro de destino; en este caso es el registro correspondiente al seteo del `PORTA`.

```
BCF PORTA, 0
```

Significa Limpiar (clear) un Bit de un Registro. Debe ser especificado tanto el Registro como el número de bit.

Recuérdese lo explicado para la configuración de puertos o pines como entrada/salida: se accedió al `banco 1`, seteando el `bit 5` del registro de estado. Una vez configurado el puerto con los valores requeridos para el desarrollo de la aplicación, se resetea el bit, y se retorna al `banco 0`, que es desde donde se corre la aplicación.

```
BSF PORTA, 0
```

Nuevamente, la instrucción `BSF` es empleada para Setear el Bit de un Registro; en este caso, el `Bit 0` del `PORTA` para encender el diodo LED.

```
END
```

Representa el fin del programa.

Recuérdese que las instrucciones que pueden ser empleadas para programar un microcontrolador en Assembler están determinadas por su Set de Instrucciones (apartado 6.9.1).

8.16 SET DE INSTRUCCIONES DEL µC 16F84A.

Mnemonic Operands		Description	Cycles	14-Bit Op-code Msb			Lsb	Status, Affected
BYTE-ORIENTED FILE REGISTER OPERATIONS								
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z
CLRF	f	Clear f	1	00	0001	ffff	ffff	Z
CLRW	—	Clear W	1	00	0001	0xxx	xxxx	Z
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff	
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff	
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z
MOVWF	f, d	Move W to f	1	00	0000	ffff	ffff	
NOP	—	No Operation	1	00	0000	0xx0	0000	
RLF	f, d	Rotate left f through Carry	1	00	1101	dfff	ffff	C
RRF	f, d	Rotate rigth f through Carry	1	00	1100	dfff	ffff	C
SUBWF	f, d	Substract W from f	1	00	0010	dfff	ffff	C,DC,Z
SWAPF	f, d	Swap ribbles in f	1	00	1110	dfff	ffff	
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z
BIT ORIENTED FILE REGISTER OPERATIONS								
BCF	f, b	Bit clear f	1	01	00bb	bfff	ffff	
BSF	f, b	Bit set f	1	01	01bb	bfff	ffff	
BTFSC	f, b	Bit test f, Skip if clear	1 (2)	01	10bb	bfff	ffff	
BTFSS	f, b	Bit test f, Skip if Set	1 (2)	01	11bb	bfff	ffff	
LITERAL AND CONTROL OPERATIONS								
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk	
CLRWDT	—	Clear Watch dog Timer	1	00	0000	0110	0100	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk	TO,PD
IORLW	k	Inclusive OR with W	1	11	1000	kkkk	kkkk	Z
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk	
RETFIE	—	Return from Interrupt	2	00	0000	0000	1001	
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk	
RETURN	—	Return from subrutine	2	00	0000	0000	1000	
SLEEP	—	Go into standby mode	1	00	0000	0100	0011	TO,PD
SUBLW	k	Substract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z

Figura 8.31. Set de instrucciones del Microcontrolador 16F84A.

8.17 CONFIGURACIÓN BÁSICA DE LOS µC A EMPLEAR

Para la implementación de los robots propuestos en el Capítulo 9, se deberá tener en cuenta la configuración básica requerida por cada microcontrolador.

8.17.1 CONFIGURACIÓN BÁSICA DEL µC 16F84A

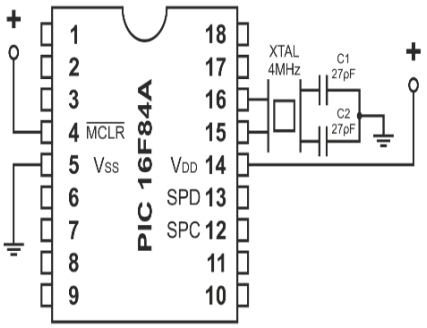


Figura 8.32. Circuito de configuración básica para el µc16F84A.

8.17.2 CONFIGURACIÓN BÁSICA DEL µC 18F4610

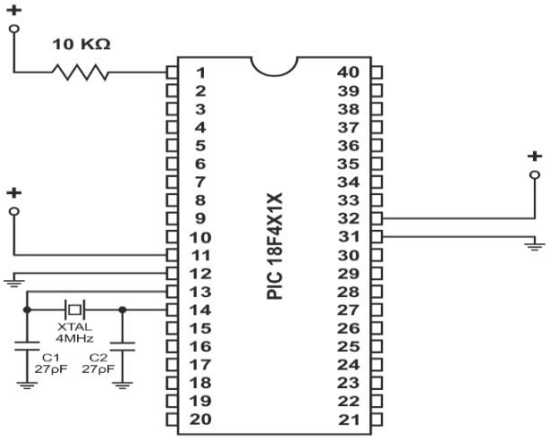


Figura 8.33. Circuito de configuración básica para el µc 18F4610A.

Un detalle a ser observado en las figuras anteriores es que se omitió el empleo del circuito de reset. El objetivo es simplificar el circuito; por tanto, el `pin 4` del Microcontrolador 16F84A es conectado de forma directa al positivo de la fuente. El `pin 1` del microcontrolador 18F4610 es conectado al positivo de la fuente a través de una resistencia de 10 KΩ.

CAPÍTULO 9
DISEÑO E IMPLEMENTACIÓN DE ROBOTS CON LÓGICA DE MICROCONTROLADOR

9.1 INTRODUCCIÓN

En el presente capítulo, se propone integrar la actividad trabajada en el capítulo anterior (Microcontroladores y su aplicación en Robótica) a la implementación realizada en el Capítulo 5 (Robot seguidor de luz sin lógica adicional), para implementar robots con mayor nivel de tecnología.

Los robots a implementar en el capítulo son los robots seguidores de luz con lógica de microcontrolador en lenguaje C, y robot seguidor de luz con lógica de microcontrolador en lenguaje Assembler o Ensamblador.

9.2 ROBOT SEGUIDOR DE LUZ CON LÓGICA DE MICROCONTROLADOR EN LENGUAJE C

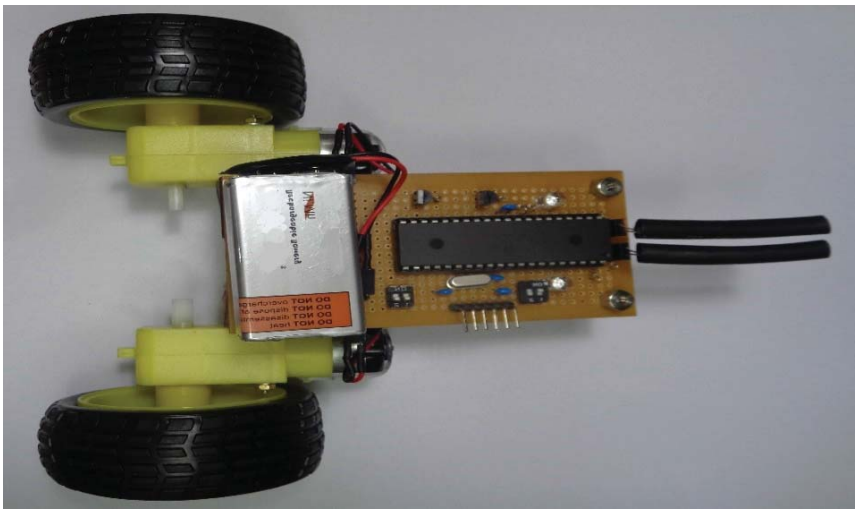


Figura 9.1 Robot seguidor de luz con lógica de microcontrolador en lenguaje C.

9.2.1 CIRCUITO ELÉCTRICO

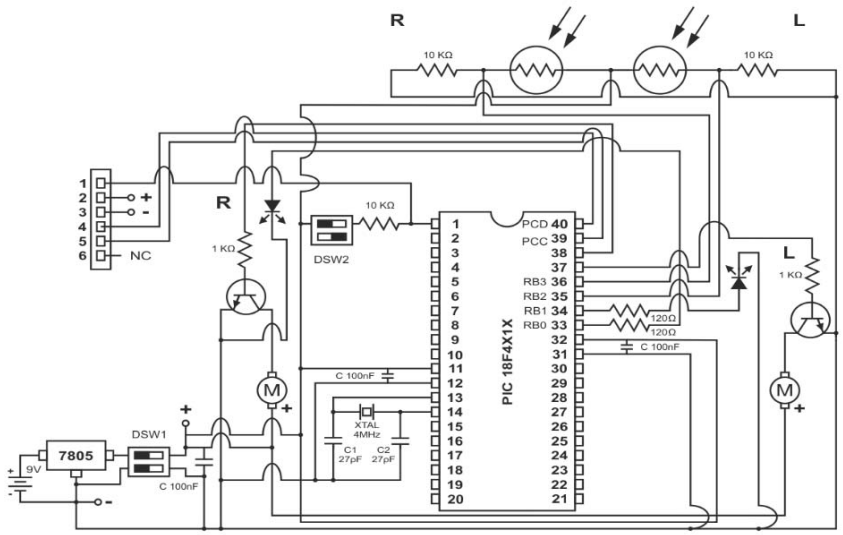


Figura 9.2 Circuito eléctrico del robot seguidor de luz con lógica de µC en lenguaje C.

9.2.2 PRINCIPIO DE FUNCIONAMIENTO

El principio de funcionamiento del Robot Seguidor de Luz con Lógica de Microcontrolador en Lenguaje C es similar al empleado en el Capítulo 5 para el robot seguidor de luz sin lógica adicional. Sin embargo, se añade un microcontrolador para controlar la conmutación (cambio de estado) de las señales provenientes de los LDR, los diodos LED empleados como diodos testigo, y la lógica de transistores para la activación o desactivación de los motores.

Los dos canales del robot son simétricos. Se toma para el análisis el lado derecho, en la Figura 9.2 (correspondiente al canal izquierdo o L (*left*), visto de frente).

La primera etapa (entrada), al igual que para el robot seguidor de luz sin lógica adicional, está compuesta por un divisor de tensión, formado por el sensor o LDR (Resistencia Variable con la Luz) y una resistencia de 10 KΩ.

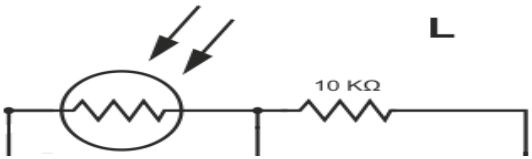


Figura 9.3. Etapa de entrada. Divisor de tensión entre LDR y R de *pull down*.

Cuando no incide luz en el LDR, este presenta una resistencia muy grande. Por tanto, el pin 35 (36 para el lado derecho) del microcontrolador recibe un estado bajo o nivel lógico cero. Esta es la tensión en el terminal superior de la R de 10 KΩ, que actúa como *pull down* (obsérvese en la Figura 9.2 que su terminal superior está conectado a 0 V, del mismo modo que el emisor del transistor).

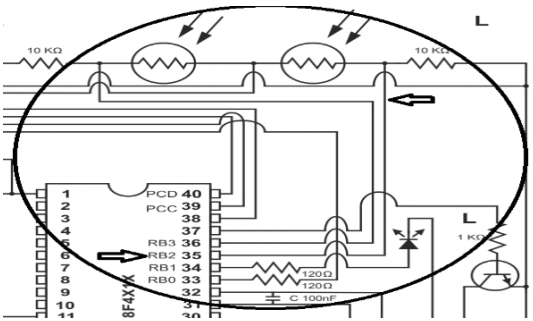


Figura 9.4. Nodo del divisor de tensión conectado al pin 35 del µC. Canal izquierdo.

Cuando la luz incide en el LDR, su resistencia baja a un valor cercano a los 50 Ω. De esta forma, los 5 V de tensión a la salida del regulador están ahora presentes en el nodo que forman el terminal inferior de la R de 10 KΩ, el LDR y el pin del microcontrolador.

Cuando el microcontrolador recibe un nivel lógico alto, envía por el pin de control de la base del transistor, pin 37 (38 para el lado derecho), un nivel lógico alto. Por lo que el transistor conmuta y actúa como una llave entre el emisor y el colector. De esta forma, como el emisor está a un potencial de 0 V, el terminal inferior del motor queda referido a este potencial y el motor funciona, ya que posee 5 V en su terminal superior.

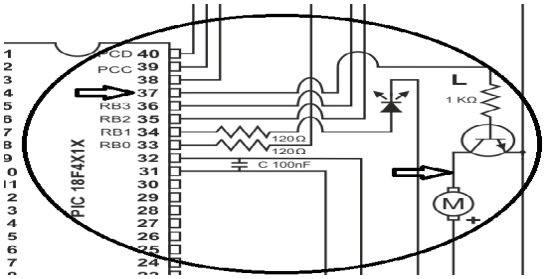


Figura 9.5. El pin 37 del µC controla la conmutación del transistor. Motor L.

Cada vez que la fuente de luz es retirada del LDR, este vuelve a su estado inicial de resistencia muy grande, y, por lo tanto, reinicia el ciclo de apagado.

Un diodo LED está asociado a cada canal para indicar la incidencia de luz en los LDR. Para el canal izquierdo, el LED está conectado al pin 34 del microcontrolador (33 para el lado derecho).

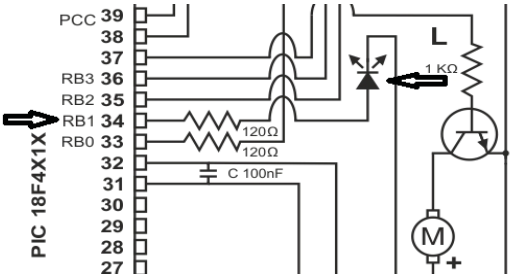


Figura 9.6. Diodo LED asociado al pin 34 del µC para el canal izquierdo.

Obsérvese en la Figura 9.2 que, a diferencia de lo trabajado en el Capítulo 5, los motores no están cruzados (en su cableado respecto a los LDR). Esto se debe a la facilidad que otorga el control de dispositivos mediante software. Bastará simplemente el seteo del pin 38 del µC (motor del lado derecho), al incidir luz en el LDR del lado izquierdo, para producir un giro a la izquierda.

Si la luz incide sobre los dos LDR al mismo tiempo, se desarrollan las instrucciones de programa necesarias para permitir el movimiento de los dos motores. Esto impulsa el movimiento del robot hacia adelante.

9.2.3 LISTADO DE MATERIALES O BOM

COMPONENTE	CANTIDAD	DESCRIPCIÓN
Placa	1	5 x 10 cm. Experimental o multipropósito. Pertinax
Microcontrolador	1	Microchip 18F4610
Zócalo DIP	1	2 x 20 pines para el µC
Batería de 9V	1	Recargable. 250 mAh. Ni.Mh. Estándar
Conector de Batería	1	Conector Broche
Imán de Neodimio	1	Opciones vistas en el Capítulo 3
Regulador de V	1	7805 - Regulador de tensión Positiva
Disipador de calor	1	Para TO220. Con tornillo y tuerca
Motores	2	6 V - 120/140 mA – con reductor
Ruedas	2	Diámetro de 65 mm con encastre para motor
Rueda loca	1	Símil Figura 3.106 u otra en existencia
LDR	2	5 mm
Termocontraíbles	2	Trozos de 4 cm para protección de LDR.
Alojamientos Hembra	5	Por 2 pines. Uso: zócalos de sensores, conexión de motores y broche de batería
Diodos LED	2	5 mm. Azules de alto brillo
Transistores	2	NPN. CTPN2222A
Resistencias	3	10 KΩ (2 en pull down, 1 en Pin 1 del µC)
Resistencias	2	1 KΩ (base de transistores)
Resistencias	2	120 Ω (Polarización de LED)
Capacitores	3	100 nF. Multicapa (para filtros)
Capacitores	2	27 pF. Multicapa (en XTAL)
XTAL	1	4 MHz (pines 13 y 14 del µC)
DIP Switch	2	x 2 contactos (alimentación y pin 1 del

		µC)
Pines rectos	6	A 90º - tira (para conexión ISP)
Pines rectos	3	x 2 pines (conexión de motores y batería)
Esquinero Ángulo	2	25 x 25 mm (sujeción de motores a placa)
Tornillos (sujeción de motor a ángulo)	2	31 mm de largo x 3 mm diámetro. Se requieren también tuercas y arandelas.
Tornillos (sujeción de ángulo a la placa)	2	15 mm de largo x 4 mm diámetro. Se requieren también tuercas y arandelas
Tornillo de cabeza plana	1	Diámetro de cabeza acorde a imán
Separadores metálicos de placa	2	25 mm. Para sujeción de rueda loca a la placa. Provee tuerca y rosca
Separadores de placa metálicos con rosca.	2	10 mm. Para sujeción de separador metálico por 25 mm a la placa

Figura 9.7. Listado de materiales o BOM.

9.2.4 MONTAJE Y PASOS DE IMPLEMENTACIÓN

Se verán a continuación las imágenes de los pasos efectuados para la implementación del Robot Seguidor de Luz con Lógica de Microcontrolador en Lenguaje C.

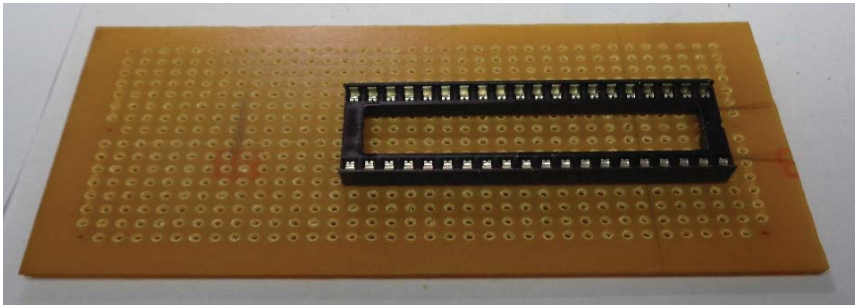


Figura 9.8. Montaje de zócalo de microcontrolador a la placa.

Recuérdese, como método de montaje, que el primer paso debe ser la distribución de componentes en la placa. Se inicia el montaje por el zócalo del microcontrolador, por tratarse del componente central y el que requerirá más espacio.

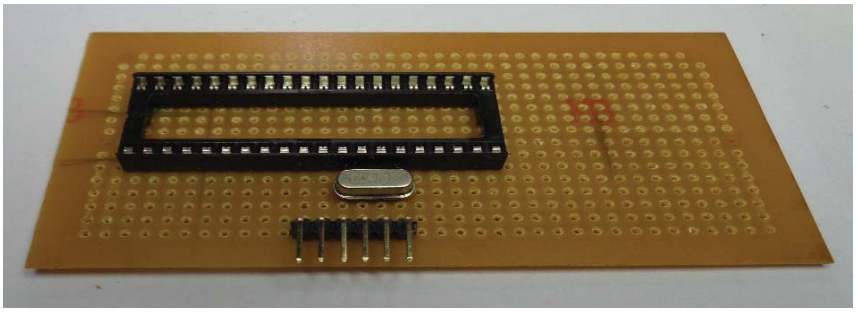


Figura 9.9. Montaje de Cristal y pines rectos a 90º para la conexión ICSP.

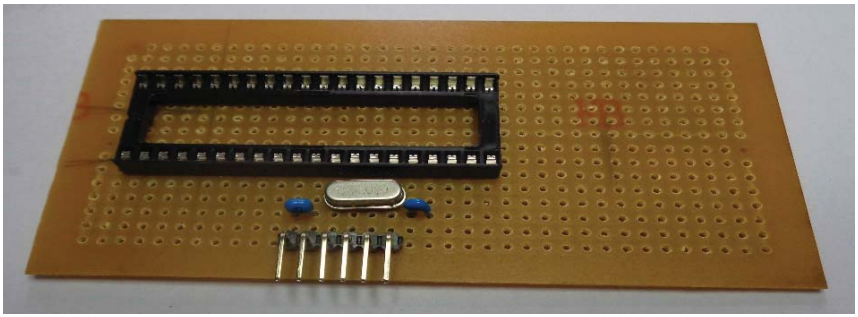


Figura 9.10. Montaje de capacitores multicapa para circuito de oscilador.

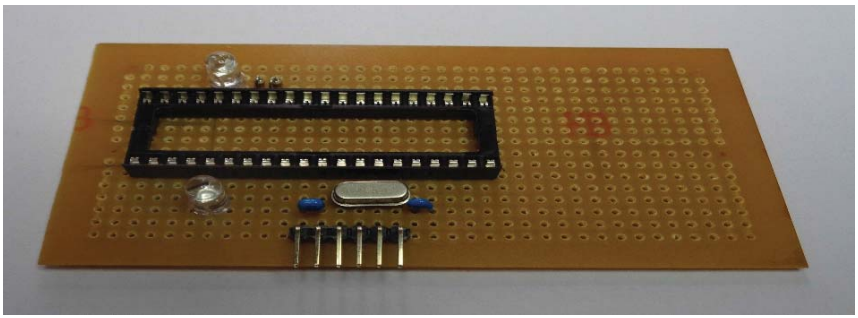


Figura 9.11. Montaje de diodos LED y sus resistencias de polarización.

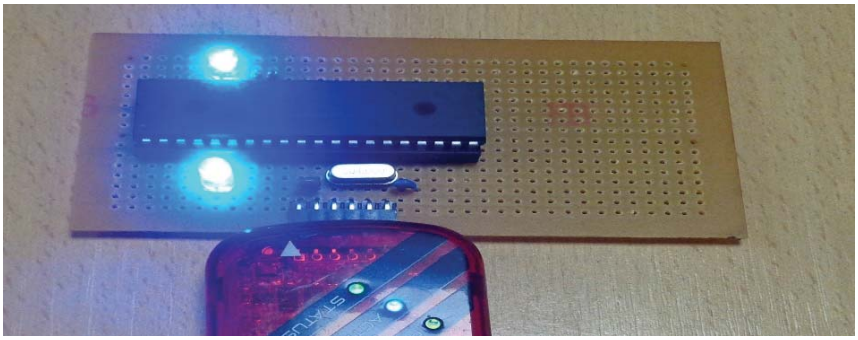


Figura 9.12. Testeo de la primera parte de montaje del circuito empleando ICSP.

Figura 9.13. Prueba de LDR a emplear como sensores.

Para leer el valor de resistencia máxima del LDR, se debe emplear termocontraíble para evitar la incidencia de luz.

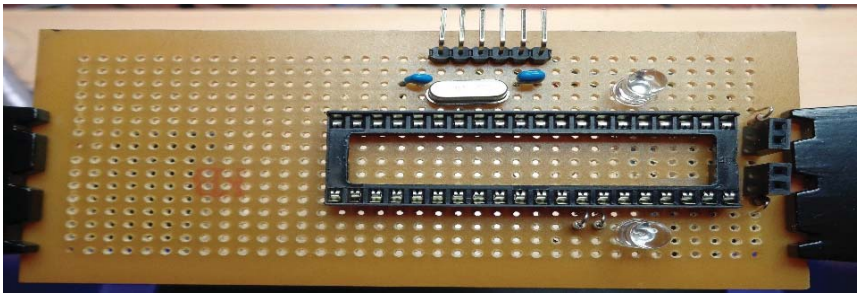


Figura 9.14. Montaje de alojamientos hembra como zócalos para sensores.

Se observa en la Figura 9.14 el montaje de la etapa de entrada, compuesta por un divisor de tensión entre los LDR y las resistencias de *pull down* para cada canal.

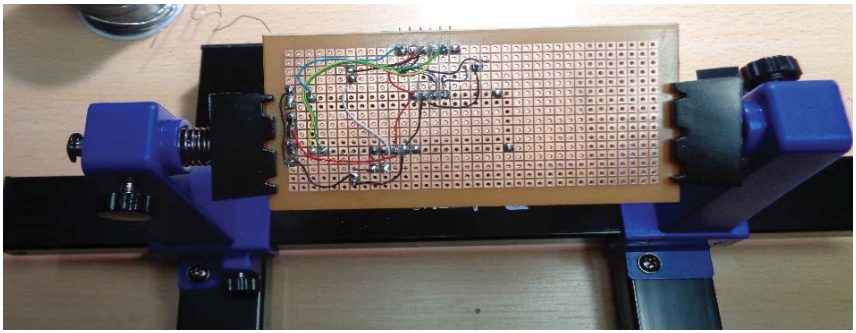


Figura 9.15. Vista del cableado parcial de la placa.

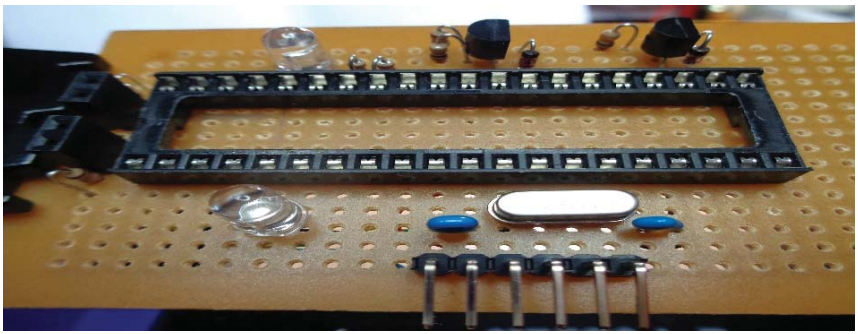


Figura 9.16. Montaje de transistores y sus resistencias de polarización.

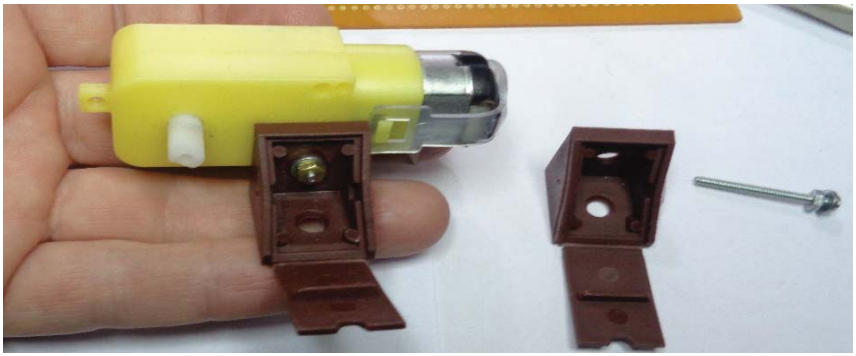


Figura 9.17. Montaje de motores a esquineros ángulos.

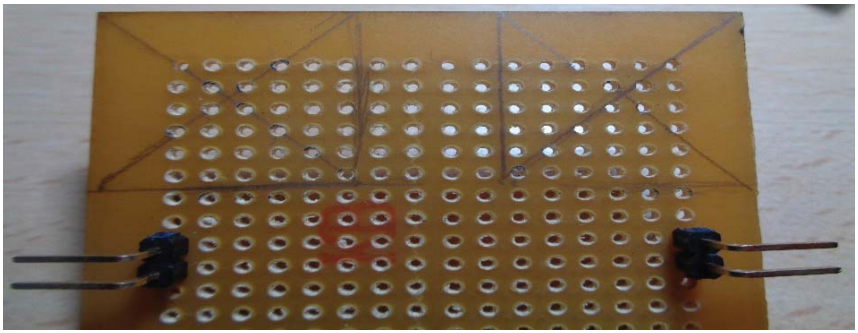


Figura 9.18. Montaje de pines rectos a 90º para conexión de motores.

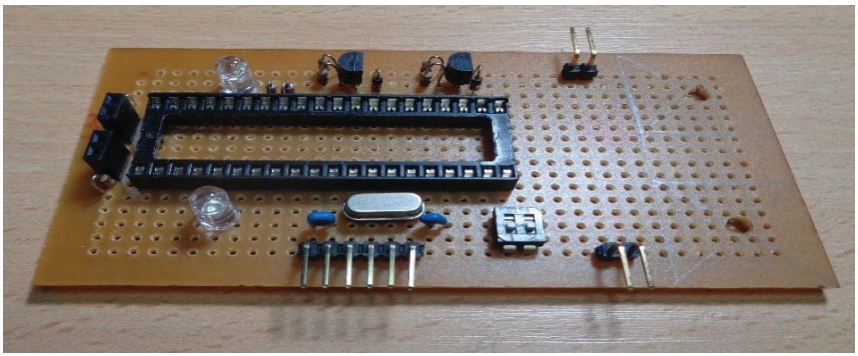


Figura 9.19. Montaje de DIP Switch para control de alimentación.

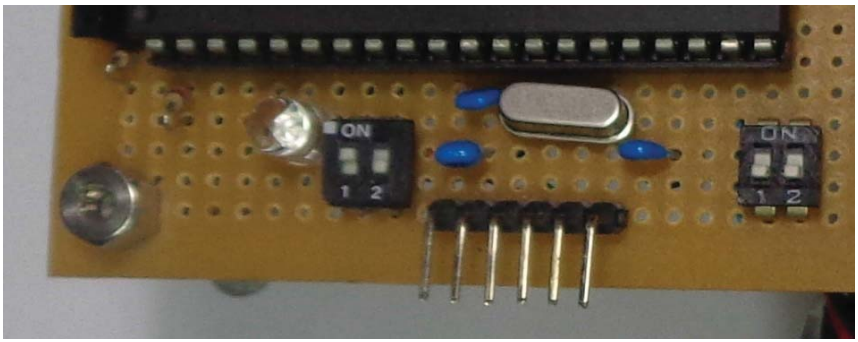


Figura 9.20. Montaje de segundo DIP Switch para alimentación del pin 1.

En el pin 1 del Microcontrolador, se encuentra el MCLR (limpiar memoria de programa). Este es activo a nivel bajo durante la programación, (y requiere que se lleve, el resto del tiempo, a un nivel alto). Sin embargo, el fabricante indica el empleo de una R de 10 KΩ para proteger su circuitería interna. Se emplea un DIP Switch para asegurar el nivel alto cuando no se emplea la programación vía ICSP.

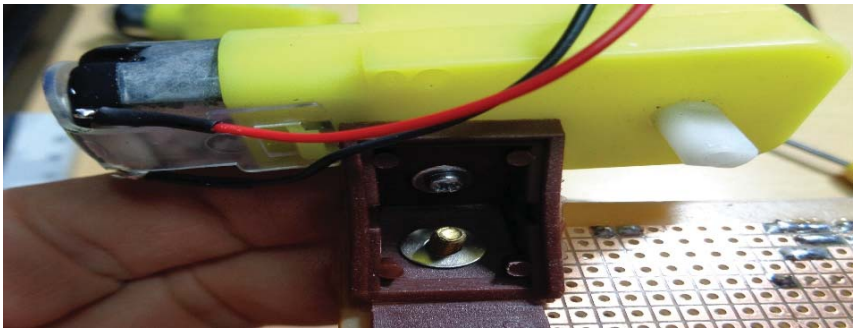


Figura 9.21. Montaje de esquinero ángulo y motor a la placa.

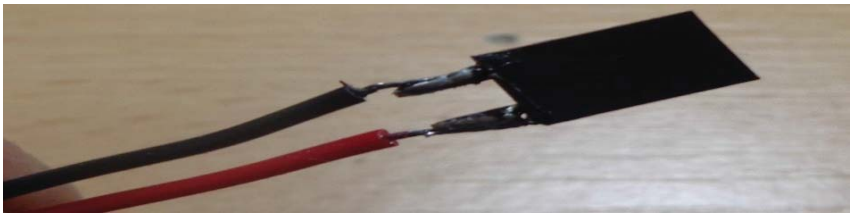


Figura 9.22. Armado de conectores para conexión de motores y batería.

Se observa, en la figura anterior, la posibilidad de implementar nuevamente herramientas propias. En este caso, se usan los alojamientos hembra y termocontraíble con el fin de generar conectores más seguros y prácticos para la conexión de los motores y la batería al circuito.

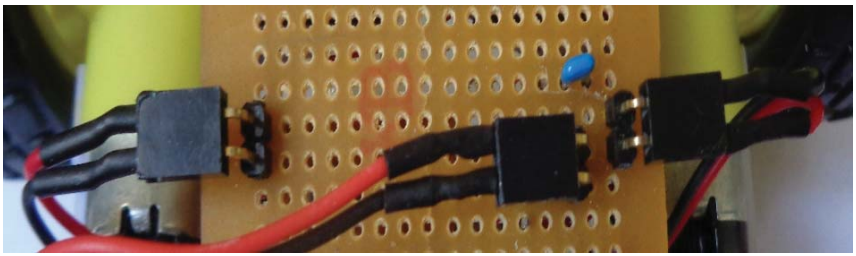


Figura 9.23. Alojamientos hembra para la implementación de conectores.

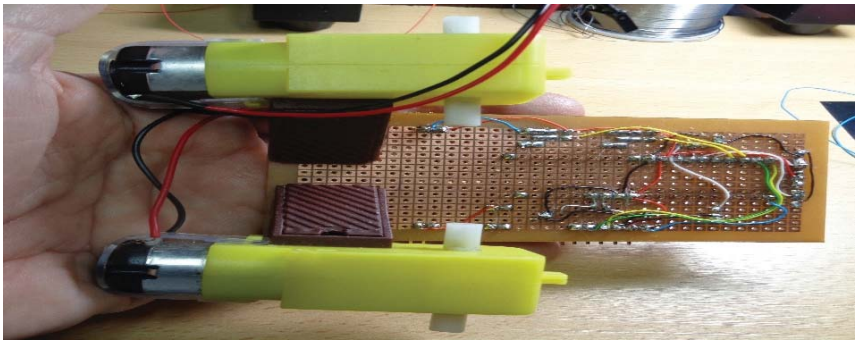


Figura 9.24. Montaje de motores y vista de cableado de la placa.

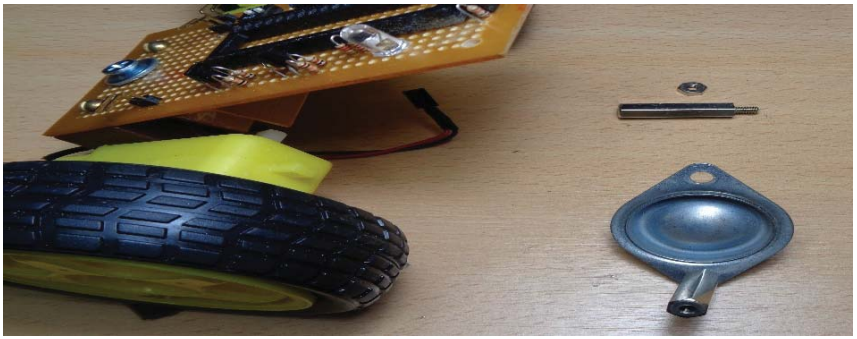


Figura 9.25. Montaje de rueda loca.

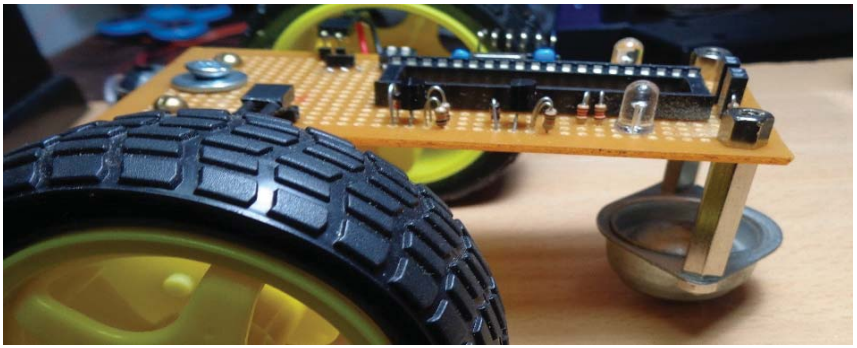


Figura 9.26. Vista del montaje final de ruedas y motores.

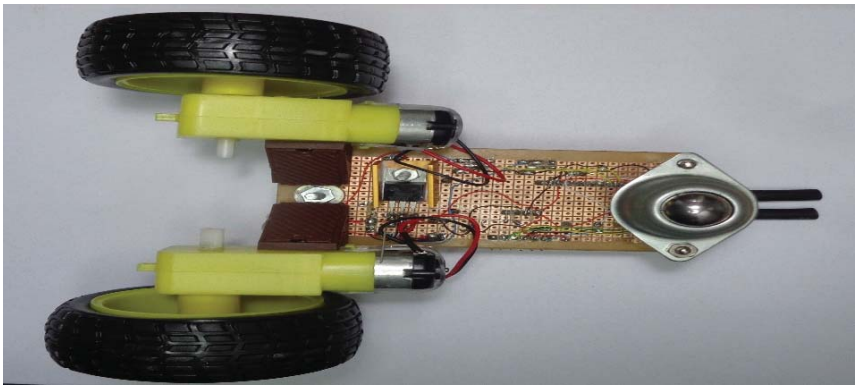


Figura 9.27. Vista de regulador y disipador en montaje final de robot.

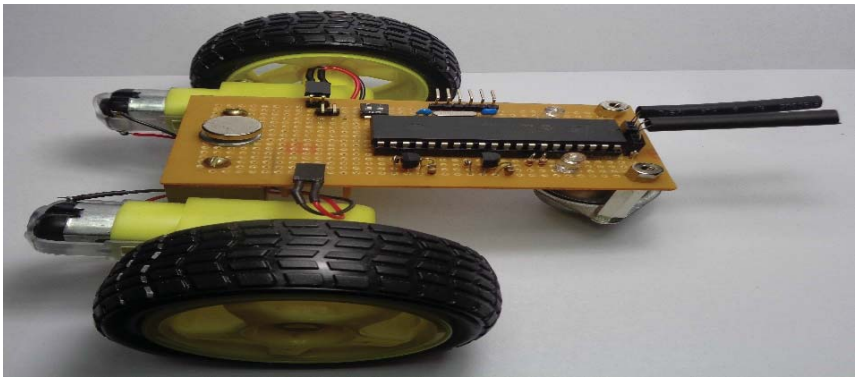


Figura 9.28. Vista del robot sin batería.

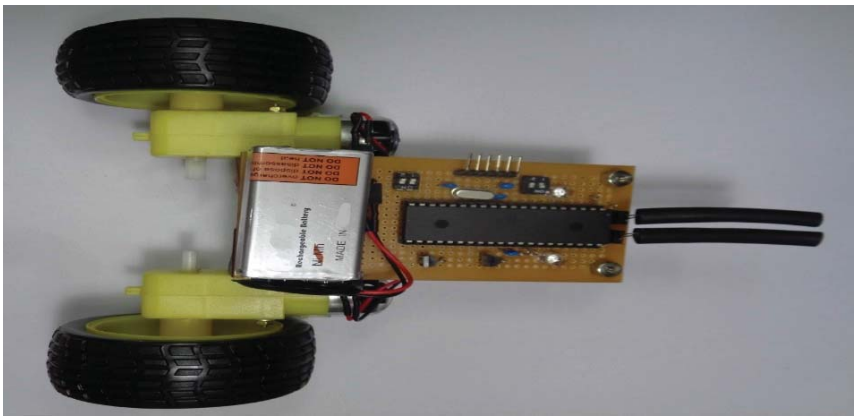


Figura 9.29. Implementación final.

9.2.5 TESTEO Y PRUEBAS DE FUNCIONAMIENTO

En la implementación de robots con lógica de microcontrolador, la fase de testeo está dividida en las siguientes etapas:

- Testeo del hardware.
- Testeo del software.
- Testeo del software en la implementación de hardware.

9.2.5.1 Testeo del hardware

El método de testeo del hardware es el propuesto inicialmente para el montaje de robots. Se recomienda testear las etapas de montaje a medida que se realizan.

En esta instancia, es necesaria la integración con el software. Es por esto que, como método de testeo debe incorporarse la aplicación del microcontrolador y las herramientas de programación.

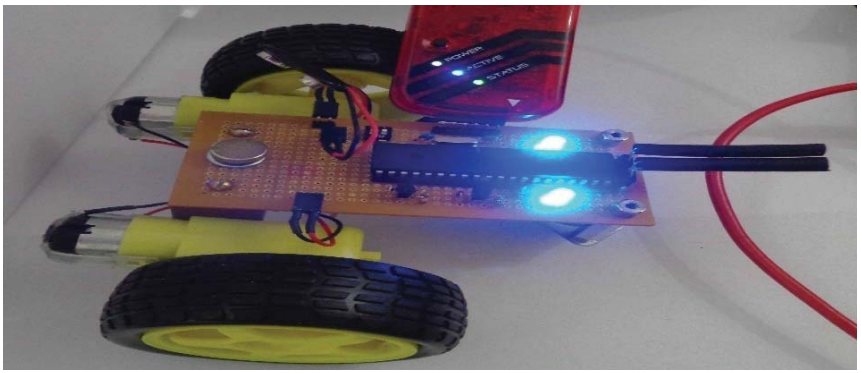


Figura 9.30. Testeo del Hardware a través del software con sistema ISP.

Otro procedimiento necesario consiste en testear la continuidad en los puntos claves del transistor. Llámese emisor a negativo; colector a extremo inferior del motor; pin de salida del microcontrolador a la resistencia de base del transistor, entre otras.

9.2.5.2 Testeo del software

Se ha señalado en el Capítulo 6 que una buena práctica en programación consiste en dividir el programa en módulos o funciones.

Este mismo método puede ser empleado para realizar pequeños programas o módulos, que permitan realizar la prueba de las distintas etapas del robot.

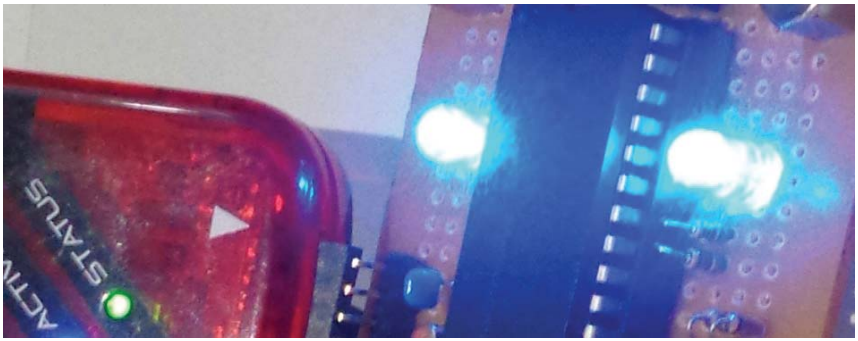


Figura 9.31. *Debugger* o prueba de funcionamiento de las rutinas de software.

9.2.5.3 Testeo del software en la implementación de hardware

Una vez desarrollados los programas a presentarse en el ítem siguiente, se procede a realizar sus pruebas. Se espera obtener las respuestas del hardware para cada requerimiento.

Figura 9.32. Prueba del software a través de la respuesta del hardware.

9.2.6 DESARROLLO DE PROGRAMAS

A continuación, se presentan dos programas empleados para las pruebas del hardware y programación del robot seguidor de luz con lógica de microcontrolador en lenguaje C.

PRIMER PROGRAMA EJEMPLO – LECTURA DE LDR Y ENCENDIDO DE LED.

Para el primer programa, se configuran los `fusibles` y `puertos` del microcontrolador a emplear. Se declara la constante simbólica `#define LDRleft PORTBbits.RB2`, que asocia la variable LDR izquierdo al pin 2 del puerto B.

Para el módulo de instrucciones:

```
while(1){
        if(LDRleft == 1){
      LATB=0X02;
```

Si el pin lee o recibe un 1 lógico, setea el pin RB1 a fin de encender el diodo LED testigo, asociado a este canal.

Para el módulo de instrucciones:

```
else
                LATB=0X00;
```

Si el LDR no recibe luz, el diodo LED testigo asociado al canal izquierdo recibe un cero lógico del pin RB1, manteniéndose en estado bajo o apagado.

Una constante simbólica es un identificador (nombre) que es reemplazado en tiempo de compilación con el texto de reemplazo establecido.

SEGUNDO PROGRAMA EJEMPLO – OPCIONES DE FUNCIONAMIENTO (ACTIVACIÓN).

Para el segundo ejemplo de programa, se realiza la configuración de `fusibles` y `puertos` para el microcontrolador a emplear.

Se declaran las constantes simbólicas `#define LDRleft PORTBbits.RB2` y `#define LDRright PORTBbits.RB3`.

Se realiza la configuración del `puerto B` en función de la necesidad de que sus `pines` sean entradas o salidas.

Se genera un *loop* infinito mediante la instrucción `while(1){`.

Dentro del cuerpo de esta instrucción `while`, se desarrollan sucesivos anidamientos para contemplar los posibles estados de los LDR (combinaciones).

Para el módulo de instrucciones:

```
while(LDRleft &&  LDRright == 0){
            LATB=0X00;//    Corresponde    al    binario
00000000. Todo apagado.
```

```
}
```

Si las entradas a los pines del microcontrolador, provenientes de LDRleft y LDRright, son iguales a cero (no incide luz en ellas), los diodos LED y los motores quedan apagados.

Para el módulo de instrucciones:

```
while(LDRleft &&  LDRright != 0){
            LATB=0XFF;//    Corresponde    al    binario
11111111. Todo encendido.
}
```

Si las entradas a los pines del microcontrolador, provenientes de LDRleft y LDRright, son distintas de cero (incide luz en ellas), los diodos LED y los motores se prenden.

Para el módulo de instrucciones:

```
while(LDRleft != 0 &&  LDRright == 0){
            LATB=0X22;//    Corresponde    al    binario
00100010.
}
```

Si el LDR del canal izquierdo recibe luz y el canal derecho (LDR) no la recibe, se enciende el LED correspondiente al lado Izquierdo (L) y el motor correspondiente al lado derecho (R) para producir un giro a la izquierda (hacia la fuente de luz).

Para el módulo de instrucciones:

```
while(LDRright != 0 &&   LDRleft == 0){
            LATB=0X11;//    Corresponde    al    binario
00010001.
}
```

Si ahora el LDR del canal derecho recibe luz y el canal izquierdo (LDR) no la recibe, se enciende el LED correspondiente al lado derecho (R) y el motor correspondiente al lado Izquierdo (L) para producir un giro a la derecha (hacia la fuente de luz).

Recuérdese, como principio físico, que, para girar a la derecha, se frena el motor derecho y se enciende el motor izquierdo. Para girar a la izquierda, se realiza el proceso opuesto: se frena el motor izquierdo y se enciende el motor derecho.

Los pines RB7, RB6, RB3 y RB2 están configurados como entradas; por tanto, es indistinto enviarles un 0 o un 1.

PRIMER PROGRAMA EJEMPLO – CÓDIGO DE PROGRAMA.

```
/// PIC18F4610 Configuration Bit Settings

#include <xc.h>


// CONFIG1H

#pragma config OSC = INTIO67 // Oscillator Selection bits (XT oscillator).

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor Enable bit (Fail-Safe Clock Monitor disabled).

#pragma config IESO = OFF // Internal/External Oscillator Switchover bit (Oscillator Switchover mode disabled).


// CONFIG2L

#pragma config PWRT =ON // Power-up Timer Enable bit (PWRT enabled).

#pragma config BOREN = SBORDIS // Brown-out Reset Enable bits (Brown-out Reset enabled in hardware only (SBOREN is disabled)).

#pragma config BORV = 3 // Brown Out Reset Voltage bits (Minimum setting).


// CONFIG2H

#pragma config WDT = OFF // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit)).

#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768).


// CONFIG3H

#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1).

#pragma config PBADEN = ON // PORTB A/D Enable bit (PORTB<4:0> pins are configured as analog input channels on Reset).

#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation).
```

```
#pragma config MCLRE = OFF // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled).

// CONFIG4L
#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset).
#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled).
#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode)).

// CONFIG5L
#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-003FFFh) not code-protected).
#pragma config CP1 = OFF // Code Protection bit (Block 1 (004000-007FFFh) not code-protected).
#pragma config CP2 = OFF // Code Protection bit (Block 2 (008000-00BFFFh) not code-protected).
#pragma config CP3 = OFF // Code Protection bit (Block 3 (00C000-00FFFFh) not code-protected).

// CONFIG5H
#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected).

// CONFIG6L
#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-003FFFh) not write-protected).
#pragma config WRT1 = OFF // Write Protection bit (Block 1 (004000-007FFFh) not write-protected).
#pragma config WRT2 = OFF // Write Protection bit (Block 2 (008000-00BFFFh) not write-protected).
#pragma config WRT3 = OFF // Write Protection bit (Block 3 (00C000-00FFFFh) not write-protected).
```

```
// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected).

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected).


// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-003FFFh) not protected from table reads executed in other blocks).

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (004000-007FFFh) not protected from table reads executed in other blocks).

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (008000-00BFFFh) not protected from table reads executed in other blocks).

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (00C000-00FFFFh) not protected from table reads executed in other blocks).


// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks).

#define LDRleft PORTBbits.RB2

int main(void)

{

ADCON1bits.PCFG=0b00001111; // Configuración de los 4 puertos como digitales.

    TRISB=0XCC;    //Configuracióndel    Port    B    como Entradas/Salidas. En binario 11001100.

while(1){

        if(LDRleft == 1){

            LATB=0X02;//0X02   corresponde   al   binario 00000010 necesario para encender el LEDizquierdo (View Front) conectado al pin RB1.
```

```
}
       else
                LATB=0X00;//Si no incide luz en el LDR,se
mantiene el LED apagado.
    }

return;
 }
```

SEGUNDO PROGRAMA EJEMPLO – CÓDIGO DE PROGRAMA.

```
 /// PIC18F4610 Configuration Bit Settings

 #include <xc.h>

 // CONFIG1H
 #pragma config OSC = INTIO67 // Oscillator Selection
bits (XT oscillator).
 #pragma config FCMEN = OFF // Fail-Safe Clock Monitor
Enable bit (Fail-Safe Clock Monitor disabled).
 #pragma config IESO = OFF // Internal/External
Oscillator Switchover bit (Oscillator Switchover mode
disabled).

 // CONFIG2L
 #pragma config PWRT =ON // Power-up Timer Enable bit
(PWRT enabled).
 #pragma config BOREN = SBORDIS // Brown-out Reset Enable
bits (Brown-out Reset enabled in hardware only (SBOREN is
disabled)).
 #pragma config BORV = 3 // Brown Out Reset Voltage bits
(Minimum setting).

 // CONFIG2H
 #pragma config WDT = OFF // Watchdog Timer Enable bit
(WDT disabled (control is placed on the SWDTEN bit)).
```

```
#pragma config WDTPS = 32768 // Watchdog Timer Postscale Select bits (1:32768).

// CONFIG3H
#pragma config CCP2MX = PORTC // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1).
#pragma config PBADEN = ON // PORTB A/D Enable bit (PORTB<4:0> pins are configured as analog input channels on Reset).
#pragma config LPT1OSC = OFF // Low-Power Timer1 Oscillator Enable bit (Timer1 configured for higher power operation).
#pragma config MCLRE = OFF // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled).

// CONFIG4L
#pragma config STVREN = ON // Stack Full/Underflow Reset Enable bit (Stack full/underflow will cause Reset).
#pragma config LVP = OFF // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled).
#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled (Legacy mode)).

// CONFIG5L
#pragma config CP0 = OFF // Code Protection bit (Block 0 (000800-003FFFh) not code-protected).
#pragma config CP1 = OFF // Code Protection bit (Block 1 (004000-007FFFh) not code-protected).
#pragma config CP2 = OFF // Code Protection bit (Block 2 (008000-00BFFFh) not code-protected).
#pragma config CP3 = OFF // Code Protection bit (Block 3 (00C000-00FFFFh) not code-protected).

// CONFIG5H
#pragma config CPB = OFF // Boot Block Code Protection bit (Boot block (000000-0007FFh) not code-protected).
```

```
// CONFIG6L

#pragma config WRT0 = OFF // Write Protection bit (Block 0 (000800-003FFFh) not write-protected).

#pragma config WRT1 = OFF // Write Protection bit (Block 1 (004000-007FFFh) not write-protected).

#pragma config WRT2 = OFF // Write Protection bit (Block 2 (008000-00BFFFh) not write-protected).

#pragma config WRT3 = OFF // Write Protection bit (Block 3 (00C000-00FFFFh) not write-protected).


// CONFIG6H

#pragma config WRTC = OFF // Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write-protected).

#pragma config WRTB = OFF // Boot Block Write Protection bit (Boot block (000000-0007FFh) not write-protected).


// CONFIG7L

#pragma config EBTR0 = OFF // Table Read Protection bit (Block 0 (000800-003FFFh) not protected from table reads executed in other blocks).

#pragma config EBTR1 = OFF // Table Read Protection bit (Block 1 (004000-007FFFh) not protected from table reads executed in other blocks).

#pragma config EBTR2 = OFF // Table Read Protection bit (Block 2 (008000-00BFFFh) not protected from table reads executed in other blocks).

#pragma config EBTR3 = OFF // Table Read Protection bit (Block 3 (00C000-00FFFFh) not protected from table reads executed in other blocks).


// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protection bit (Boot block (000000-0007FFh) not protected from table reads executed in other blocks).
```

```
#define LDRleft PORTBbits.RB2
#define LDRright PORTBbits.RB3

int main(void)
{
    ADCON1bits.PCFG=0b00001111; // Configuración de los
4 puertos como digitales.

   TRISB=0XCC;   //Configuración   del   Port   B   como
Entradas/Salidas. En binario 11001100.

while(1){

        LATB=0X00;

        while(LDRleft &&  LDRright == 0){
            LATB=0X00;
         }

        while(LDRleft &&  LDRright != 0){
            LATB=0XFF;
}
        while(LDRleft != 0 &&  LDRright == 0){
            LATB=0XD2;
         }

        while(LDRright != 0 &&   LDRleft == 0){
            LATB=0XE1;
}

    }
```

```
    return;
}
```

9.3 ROBOT SEGUIDOR DE LUZ CON LÓGICA DE MICROCONTROLADOR EN LENGUAJE ASSEMBLER

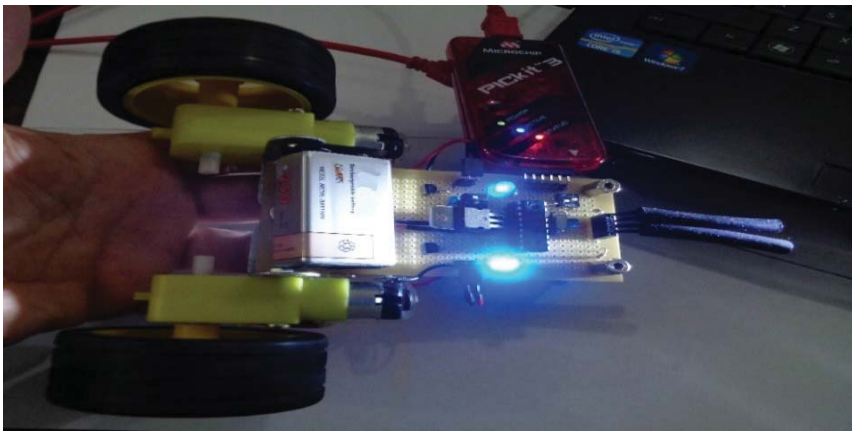


Figura 9.33. Robot seguidor de luz con lógica de µC en lenguaje Assembler.

9.3.1 CIRCUITO ELÉCTRICO

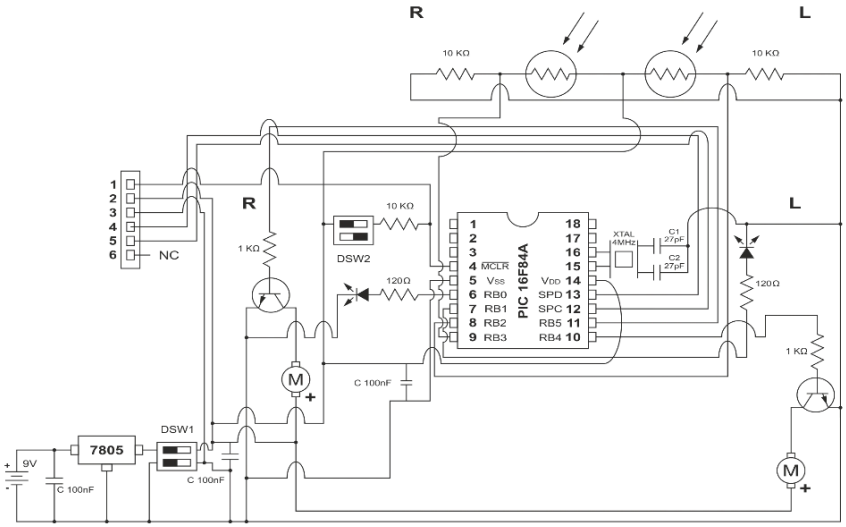


Figura 9.34. Circuito. Robot seguidor de luz con lógica de µC en lenguaje Assembler.

9.3.2 PRINCIPIO DE FUNCIONAMIENTO

El principio de funcionamiento es similar al presentado para el robot seguidor de luz con lógica de microcontrolador en lenguaje C, pero empleando el Microcontrolador 16F84A para ser programado en lenguaje Ensamblador. Ambos microcontroladores han sido presentados en el Capítulo 8.

9.3.3 LISTADO DE MATERIALES O BOM

El listado de materiales es similar al presentado en la Figura 9.7 (Listado de materiales o BOM), pero empleando el Microcontrolador 16F84A.

9.3.4 MONTAJE Y PASOS DE IMPLEMENTACIÓN

Nuevamente, al tratarse del mismo circuito, los pasos para el montaje e implementación son similares a los utilizados en el apartado 9.2.4, pero con el empleo del Microcontrolador 16F84A.

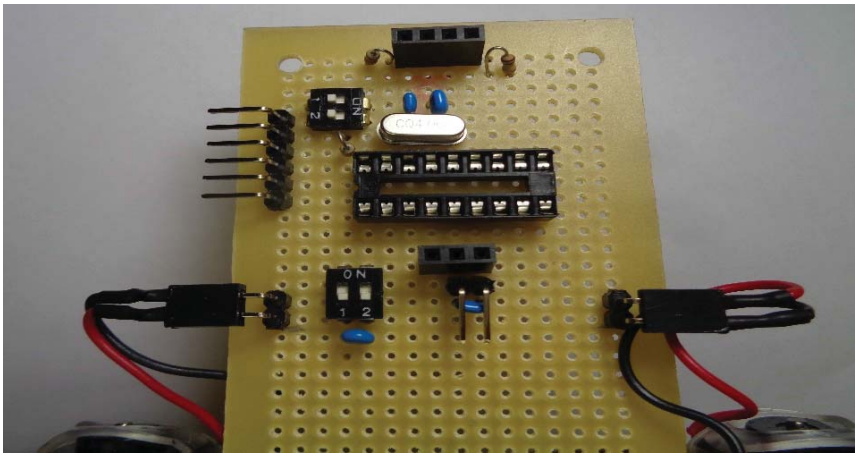


Figura 9.35. Montaje de robot con µC 16F84A.

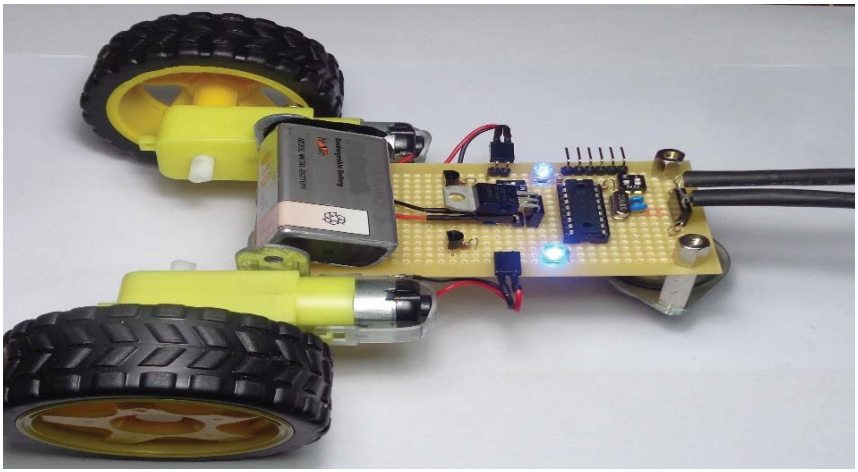


Figura 9.36. Fase de montaje e implementación.

9.3.5 TESTEO Y PRUEBAS DE FUNCIONAMIENTO

Recuérdese que, en los robots implementados con microcontrolador, la fase de testeo y prueba comprende las etapas descritas en el apartado 9.2.5 Testeo y pruebas de funcionamiento.

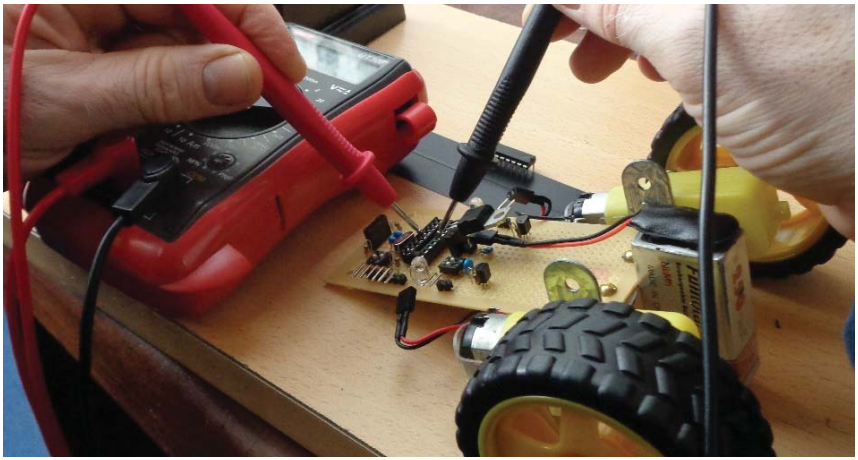


Figura 9.37. Fase de testeo del hardware para el robot con µC 16F84A.

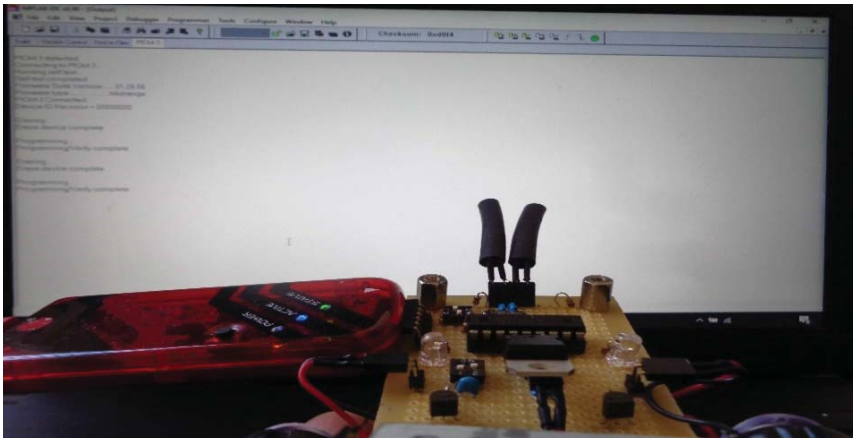


Figura 9.38. Fase de testeo del software para el robot con µC 16F84A.

9.3.6 DESARROLLO DE PROGRAMAS

PRIMER PROGRAMA EJEMPLO - TESTEO DE LOS DIODOS LED.

Recuérdese, una práctica recomendable para el desarrollo de robots con microcontroladores consiste en montar el hardware, realizar el testeo de este y, luego, iniciar la prueba de las distintas etapas mediante la realización de módulos de programa.

A continuación, se describe el desarrollo en lenguaje ensamblador del primer programa implementado. El objetivo es probar los diodos LED izquierdo y derecho del robot.

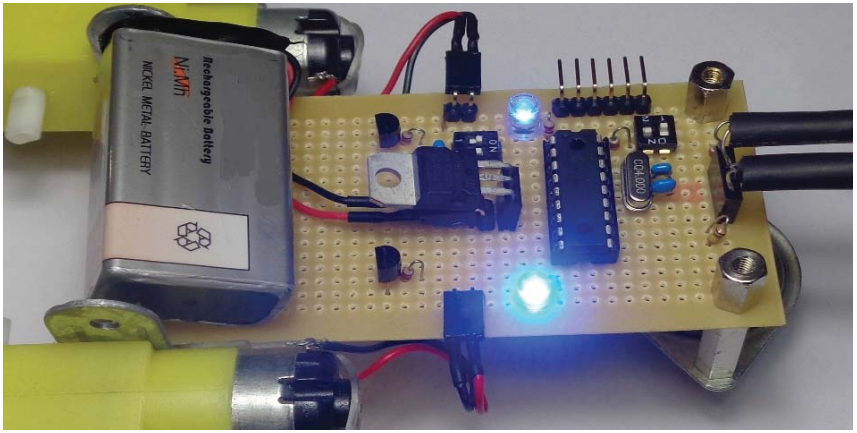


Figura 9.39. Programa de prueba de los diodos LED izquierdo y derecho.

Se emplea el IDE MPLAB, trabajado en el apartado 6.10.2.

Las instrucciones empleadas para programar el microcontrolador son las correspondientes a su Set de Instrucciones, Figura 8.31.

Obsérvese de la Figura 9.34 (circuito del robot) las siguientes conexiones a los pines del PORTB. Serán necesarias al momento de la programación para declarar los pines del puerto como entradas o salidas.

PIN	HARDWARE CONECTADO
0	Conexión del LED Derecho
1	Conexión del LED Izquierdo
2	LDR Izquierdo
3	LDR Derecho
4	Transistor de lado izquierdo
5	Transistor de lado derecho
6	Serial Programming Clock. A pin 5 del ICSP
7	Serial Programming Data. A pin 4 del ICSP

Figura 9.40. Descripción del hardware conectado a los pines del PORTB.

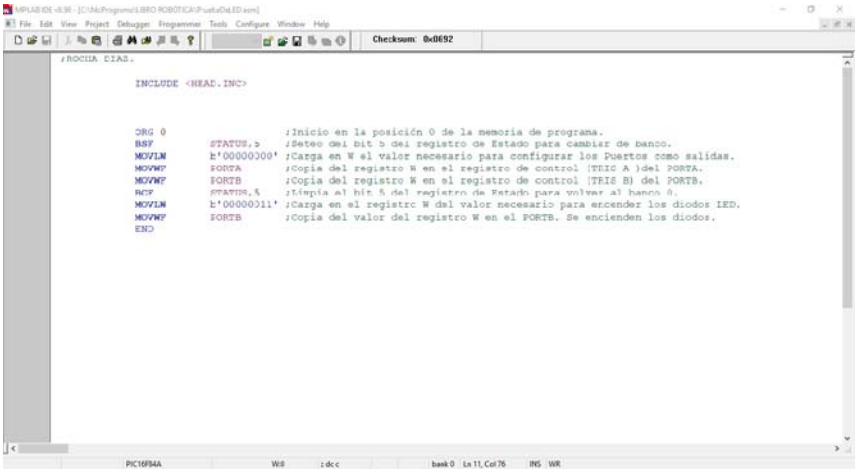


Figura 9.41. Programa en lenguaje Assembler para probar los diodos LED.

Se muestra, a continuación, la vista ampliada del código de programa de la Figura 9.41, desarrollado para la prueba de los diodos LED del robot.

```
;ROCHA DIAZ.

        INCLUDE <HEAD.INC>


        ORG 0                   ;Inicio en la posición 0 de la memoria de programa.
        BSF     STATUS,5        ;Seteo del bit 5 del registro de Estado para cambiar de banco.
        MOVLW   b'00000000'     ;Carga en W el valor necesario para configurar los Puertos como salidas.
        MOVWF   PORTA           ;Copia del registro W en el registro de control (TRIS A )del PORTA.
        MOVWF   PORTB           ;Copia del registro W en el registro de control (TRIS B) del PORTB.
        BCF     STATUS,5        ;Limpia el bit 5 del registro de Estado para volver al banco 0.
        MOVLW   b'00000011'     ;Carga en el registro W del valor necesario para encender los diodos LED.
        MOVWF   PORTB           ;Copia del valor del registro W en el PORTB. Se encienden los diodos.
        END
```

Figura 9.42. Vista ampliada. Programa de prueba para los diodos LED.

Explicación de la sintaxis del programa:

Los caracteres:

`;ROCHA DIAZ.`

En la línea 1, hacen referencia al autor. Recuérdese que el texto posterior al punto y coma (;) es una línea de comentarios (el compilador no la considera).

En la línea 3

`INCLUDE <HEAD.INC>`

Su significado fue presentado en la Figura 6.39.

La configuración de bits que este archivo posee puede ser visualizada de la siguiente forma:

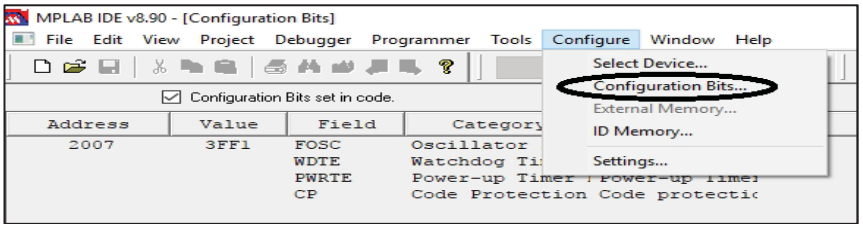


Figura 9.43. Descripción de configuración de bits para el archivo `HEAD.INC`.

La línea del programa en análisis puede ser identificada posicionando el cursor en ella, y leyendo su posición en la parte inferior del IDE. Se observan también otras características, tales como: el microcontrolador empleado y las configuraciones del proyecto.

Figura 9.44. Información del número de línea y otros datos del proyecto.

Para el bloque de instrucciones:

```
ORG 0        ;Inicio en la posición 0 de la memoria de programa.

BSF STATUS,5       ;Seteo del bit 5 del registro de Estado para cambiar de banco.

MOVLW b'00000000' ;Carga en W el valor necesario para configurar los Puertos como salidas.

MOVWF PORTA        ;Copia del registro W en el registro de control (TRIS A) del PORTA.

MOVWF PORTB ;Copia del registro W en el registro de control (TRIS B) del PORTB.

BCF STATUS,5       ;Limpia el bit 5 del registro de Estado para volver al banco 0.
```

En el ítem 8.15.2 fue presentado el funcionamiento de los `puertos` (`Ports`), y el procedimiento para la configuración general o individual de cada pin.

Para este proyecto, no se emplea el PORTA; por lo tanto, se configuran todos los pines de este como salida, para evitar cualquier posibilidad de ingreso de señal de ruido.

En la línea de instrucción:

```
MOVLW              b'00000011'
```

Como se expresa en la línea de comentarios, se carga literalmente en el registro W el binario equivalente para setear los pines RB0 y RB1.

En la línea de instrucción:

```
MOVWF        PORTB
```

Se copia (mueve) el valor binario cargado en el `registro W` al `PORTB`. De esta forma, se encienden los diodos LED izquierdo y derecho.

La instrucción `END` indica el cierre o fin del programa.

Para la compilación del programa, se deben seguir los siguientes pasos:

Figura 9.45. Compilación del programa para obtener el archivo `.hex`.

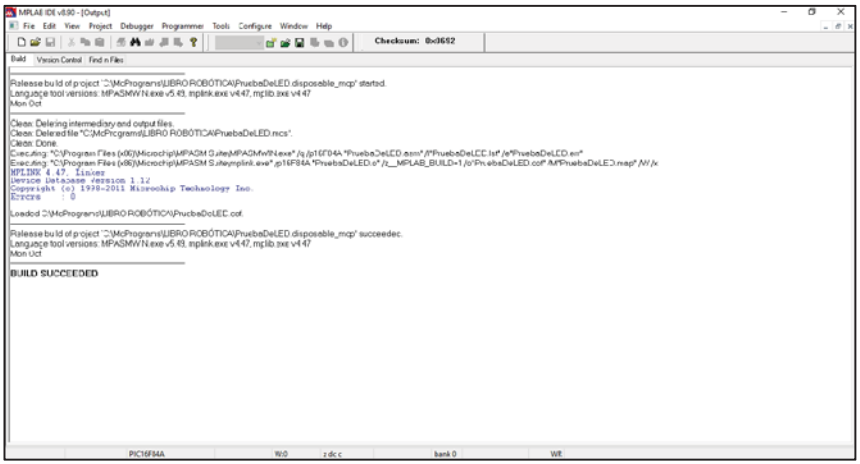


Figura 9.46. Compilación exitosa.

Una característica en el trabajo con los microcontroladores de Microchip es que si Ud. observa en la carpeta del proyecto en curso. El archivo ejecutable o `.hex`, se representa en mayúscula, como `.HEX`.

Figura 9.47. Sintaxis de la extensión del archivo ejecutable.

Una vez que el programa ha sido compilado, se debe conectar el programador para poder descargar, mediante el sistema ICSP, el código de programa a la memoria del microcontrolador.

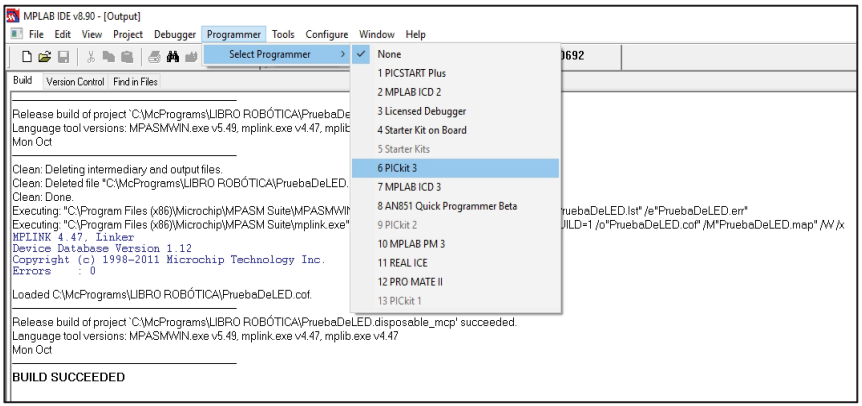


Figura 9.48. Conexión del Programador PICkit3.

Si al momento de conectar el programador aparece la siguiente leyenda:

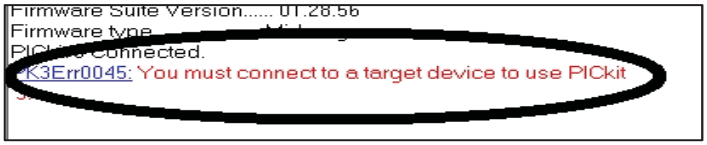


Figura 9.49. Mensaje de solicitud. Falta tensión de alimentación en la placa.

Se debe proceder de la siguiente forma:

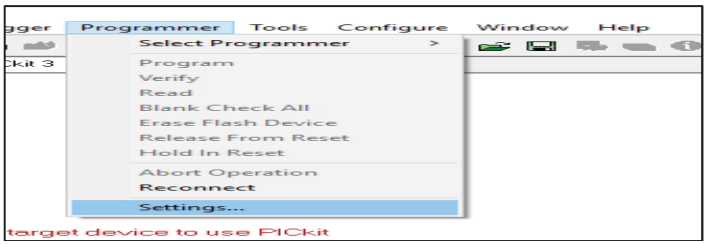


Figura 9.50. Procedimiento para alimentar la placa.

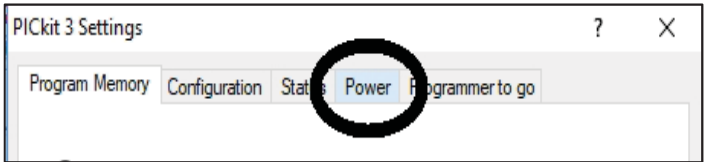


Figura 9.51. Seteo de alimentación de la placa.

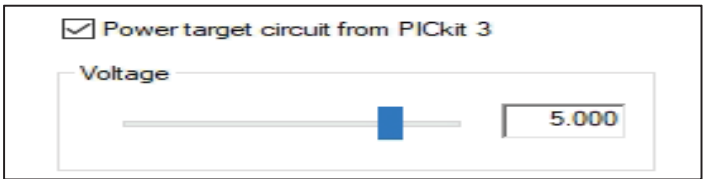


Figura 9.52. Clic en opción de alimentación.

Luego, al momento de seleccionar la solapa `Aplicar` aparece el siguiente cuadro:

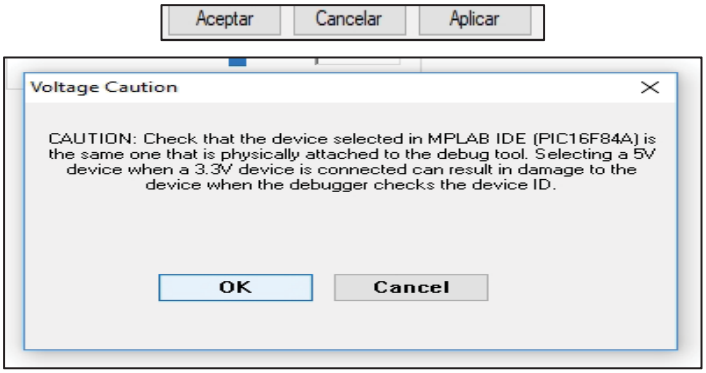


Figura 9.53. Aviso de precaución sobre tensión de trabajo del µC.

Se selecciona `OK`, debido a que las familias de microcontroladores empleadas para el trabajo requieren de 5 V.

Luego se selecciona `Aceptar`; aparecerán sobre el margen superior derecho de la pantalla los siguientes íconos:

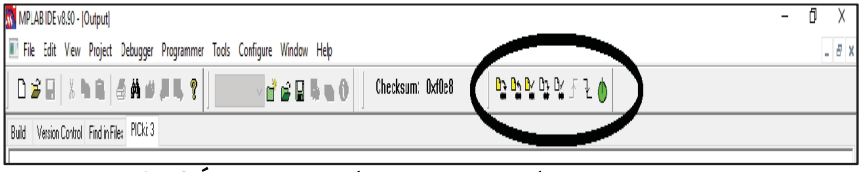


Figura 9.54. Íconos de opción de programación del microcontrolador.

Las opciones más empleadas son:

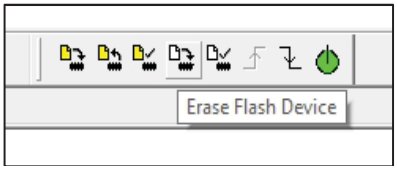


Figura 9.55. Opción de borrado de la memoria del dispositivo.

Luego:

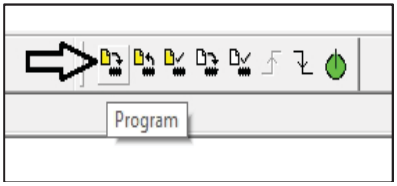
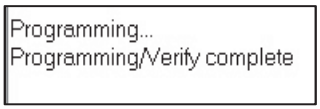


Figura 9.56. Opción de programación del dispositivo.



SEGUNDO PROGRAMA EJEMPLO - TESTEO DE LDR Y LED IZQUIERDOS.

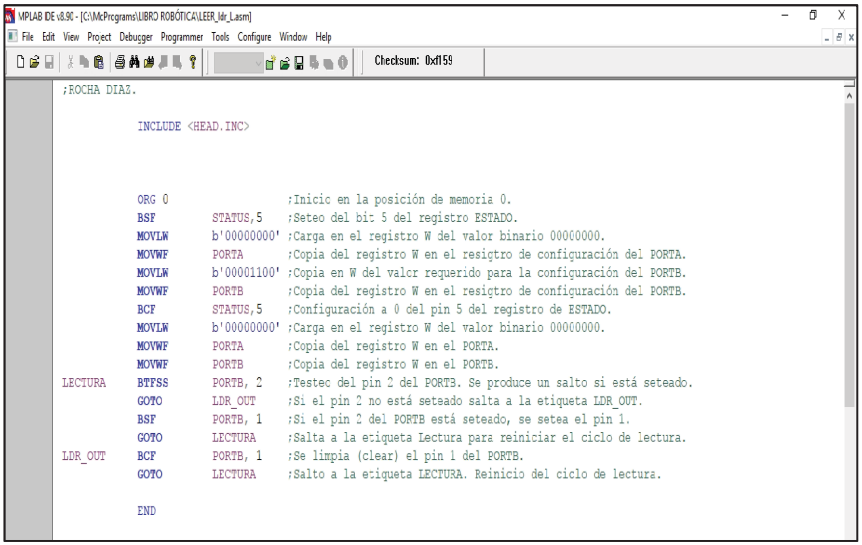


Figura 9.57. Programa de prueba. Testeo de LDR y encendido de LED izquierdo.

Las primeras líneas del programa (o bloque de instrucciones), han sido descritas en el ejemplo anterior. Recuérdese también que, para los microcontroladores de Microchip, los pines a configurar como entrada son seteados a 1. Por tanto, se configuran los pines 2 y 3 como entradas de los LDRs L (izquierdo) y R (derecho).

`MOVLW b'00001100'` ;Copia en W del valor requerido para la configuración del PORTB.

Luego de configurar los puertos (pines) de acuerdo a la necesidad del proyecto, se retorna al banco 0 para la ejecución de las instrucciones.

`BCF STATUS,5` ;Configuración a 0 del `pin 5` del registro de ESTADO.

En las tres líneas de instrucciones siguientes, se carga el `registro W` con el valor binario necesario para limpiar posteriormente los pines de los `puertos A` y `B`. Esto se realiza para iniciar el proceso con los pines en estado bajo (*reset*).

`MOVLW b'00000000'` ;Carga en el `registro W` del valor binario `00000000`.

`MOVWF PORTA` ;Copia del `registro W` en el `PORTA`.

`MOVWF PORTB` ;Copia del `registro W` en el `PORTB`.

En la línea siguiente, se realiza la lectura correspondiente al `pin 2` del `PORTB`, en el cual está conectado el sensor izquierdo 'L' mediante un divisor de tensión con la R de 10KΩ.

Se emplea la instrucción `BTFSS f,b` (testear bit f y saltar si está seteado) cuya descripción es la siguiente:

- Si el pin está seteado (tiene un nivel lógico alto por incidencia de la luz), se ignora la línea siguiente; y se produce un salto para ejecutar la próxima instrucción.
- Si el pin no está seteado (no hubo incidencia de luz en el LDR), se ejecuta la instrucción siguiente. Para salir de esta instrucción, se deberá emplear una instrucción de salto (`goto` o ir a).

En el margen izquierdo del formulario (pantalla), `LECTURA` y `LDR_OUT`, son etiquetas. Las etiquetas marcan o indican el destino de una instrucción de salto.

En la línea de instrucción:

`BTFSS PORTB, 2` ; Testeo del `pin 2` del `PORTB`. Se produce un salto si está seteado.

Del lado izquierdo de la instrucción, se encuentra la etiqueta `LECTURA`, por lo que se observa como `LECTURA BTFSS PORTB, 2`.

En la línea de instrucción:

`GOTO LDR_OUT` ;Si el pin 2 no está seteado, ejecuta la instrucción siguiente, que requiere un salto a la etiqueta `LDR_OUT`.

Si el pin no está seteado, se ejecuta la instrucción siguiente, que determina ir (`goto`) a la etiqueta `LDR_OUT`.

En la línea de instrucción:

`BSF PORTB, 1` ;Si el pin 2 del PORTB está seteado, se setea el `pin 1`.

Si el pin 2 del PORTB está a nivel alto (seteado), se setea el `pin 1` del `PORTB`, en el cual está conectado el LED del lado Izquierdo del robot.

En la línea de instrucción:

`GOTO LECTURA` ;Salta a la etiqueta `LECTURA` para reiniciar el ciclo de lectura.

Ya cumplida esta instrucción, se produce un salto nuevamente a la etiqueta `LECTURA`, a fin de reiniciar el ciclo de lectura del pin 2.

De haberse producido un salto a la etiqueta `LDR_OUT`, se desarrollan las siguientes acciones:

`BCF PORTB, 1` ;Se limpia (clear) el pin 1 del PORTB.

De esta forma, se apaga el LED. Del lado izquierdo de la instrucción, se encuentra la etiqueta LDR_OUT, por lo que se observa como `LDR_OUT BCF PORTB, 1.`

En la línea de instrucción:

`GOTO LECTURA`; Salto a la etiqueta `LECTURA`. Reinicio del ciclo de lectura.

Una vez que el diodo LED ha sido apagado, se produce un salto a la etiqueta `LECTURA`, a fin de reiniciar el ciclo de lectura del sensor LDR.

La instrucción `END`, indica el cierre o fin del programa.

TERCER PROGRAMA EJEMPLO - TESTEO DE LDR Y LED DERECHOS.

El mismo procedimiento puede ser aplicado para la prueba del LED R (LED del lado derecho), empleando la configuración descrita en la Figura 9.40.

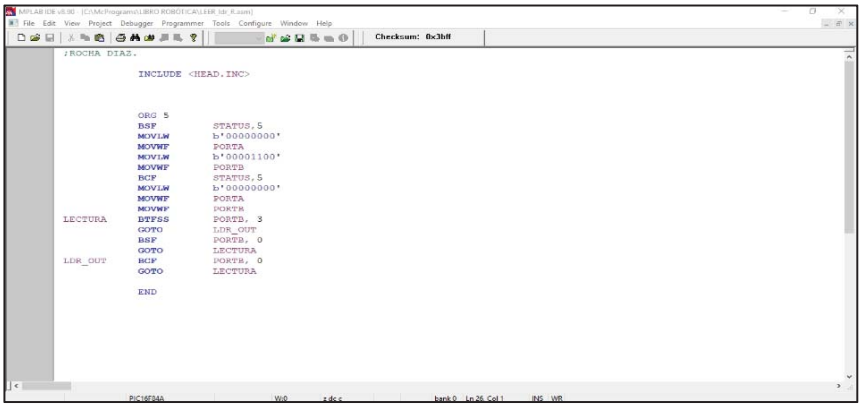


Figura 9.58. Programa de prueba. Testeo de LDR y encendido de LED derecho.

```
;ROCHA DIAZ.
                INCLUDE <HEAD.INC>

                ORG 5
                BSF         STATUS,5
                MOVLW       b'00000000'
                MOVWF       PORTA
                MOVLW       b'00001100'
                MOVWF       PORTB
                BCF         STATUS,5
                MOVLW       b'00000000'
                MOVWF       PORTA
                MOVWF       PORTB
LECTURA         BTFSS       PORTB, 3
                GOTO        LDR_OUT
                BSF         PORTB, 0
                GOTO        LECTURA
LDR_OUT         BCF         PORTB, 0
                GOTO        LECTURA

                END
```

Figura 9.59. Vista ampliada. Programa de prueba. Testeo de LDR y LED derecho.

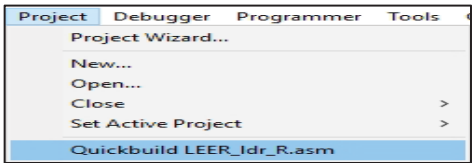


Figura 9.60. Compilación del programa para obtener el archivo `.hex`.

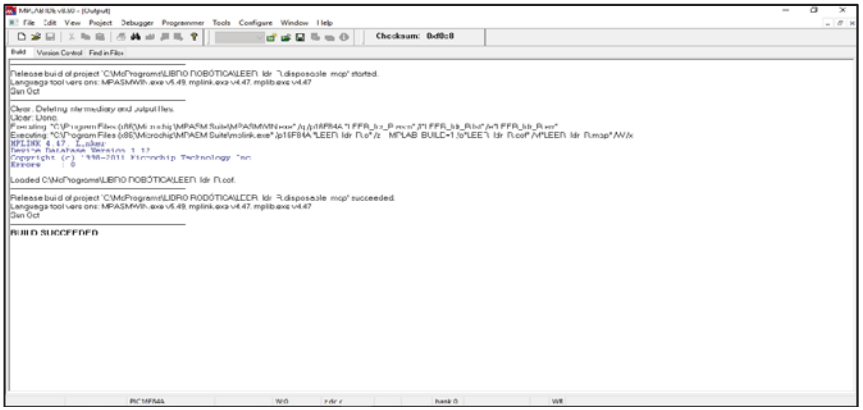


Figura 9.61. Compilación exitosa.

Por último son aplicados los pasos descritos a partir de la Figura 9.48 (Conexión del Programador PICkit3), para conectar el programador y descargar el programa a la memoria del microcontrolador.

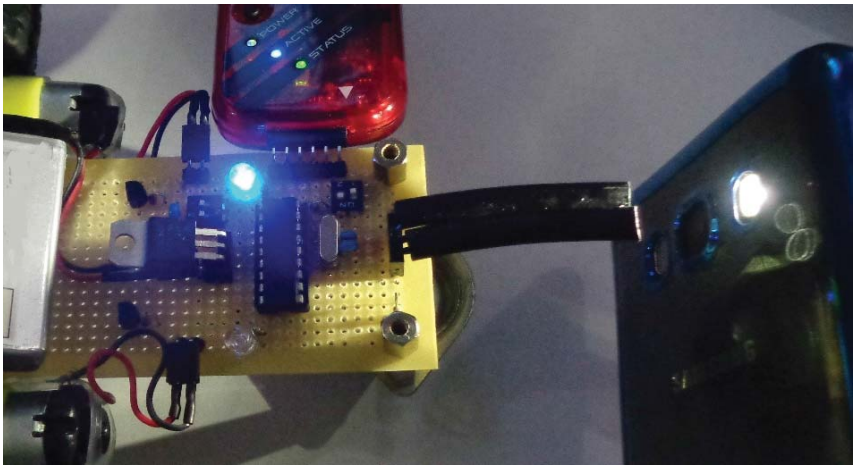


Figura 9.62. Programa de prueba. Testeo de LDR y encendido de LED derecho.

Tomando como referencia los tres programas desarrollados anteriormente (los cuales pueden ser considerados módulos para la lectura de los LDR y encendido de sus respectivos LED) se cierra el presente capítulo, con el programa final o completo, que posee también la señal de conmutación para los transistores.

CUARTO PROGRAMA EJEMPLO –PROGRAMA FINAL

```
;ROCHA DIAZ.
                    INCLUDE <HEAD.INC>

                    ORG 5
                    BSF         STATUS,5
                    MOVLW       b'00000000'
                    MOVWF       PORTA
                    MOVLW       b'00001100'
                    MOVWF       PORTB
                    BCF         STATUS,5
                    MOVLW       b'00000000'
                    MOVWF       PORTA
                    MOVWF       PORTB
LECTURA             BTFSS       PORTB, 2
                    GOTO        LDR_R
```

```
                    GOTO        ENC_LL
LDR_R               BTFSS       PORTB, 3
                    GOTO        LECTURA
ENC_LR              BSF         PORTB, 0
                    BSF         PORTB, 4
                    BTFSS       PORTB, 2
                    GOTO        OFF_LL
                    GOTO        LECTURA
OFF_LR              BCF         PORTB, 0
                    BCF         PORTB, 4
                    BTFSS       PORTB, 2
                    GOTO        OFF_LL
                    GOTO        ENC_LL
ENC_LL              BSF         PORTB, 1
                    BSF         PORTB, 5
                    BTFSS       PORTB, 3
                    GOTO        OFF_LR
                    GOTO        ENC_LR
OFF_LL              BCF         PORTB, 1
                    BCF         PORTB, 5
                    BTFSS       PORTB, 3
                    GOTO        OFF_LR
                    GOTO        ENC_LR
                    END
```

Obsérvese entonces, que las líneas de programa empleadas son similares a los módulos de programa desarrollados para los ejemplos de programas 1, 2 y 3.

Fueron agregadas las instrucciones:

`BSF PORTB, 4` para conmutar el transistor del lado izquierdo. Y la instrucción `BSF PORTB, 5` para conmutar el transistor del lado derecho.

Recuérdese la descripción de los pines del PORTB empleados para el robot, presentados en la Figura 9.40.

Son empleadas también las instrucciones `BCF PORTB, 4` y `BCF PORTB, 5` para apagar los motores a través de los transistores (VBE = 0V).

La conmutación de los transistores (encendido de motores) está dentro de la etiqueta `ENC_LL` (encender LED izquierdo) para el motor derecho (recuérdese que los motores están cruzados físicamente respecto a los sensores, para permitir que el robot gire siguiendo la fuente de luz), y `ENC_LR` (encender LED derecho) para el motor izquierdo.

De esta forma, queda establecido que los diodos LED hacen referencia al sensor (LDR) que está activo, mientras que se activa el motor del lado opuesto, a fin de realizar el correspondiente giro.

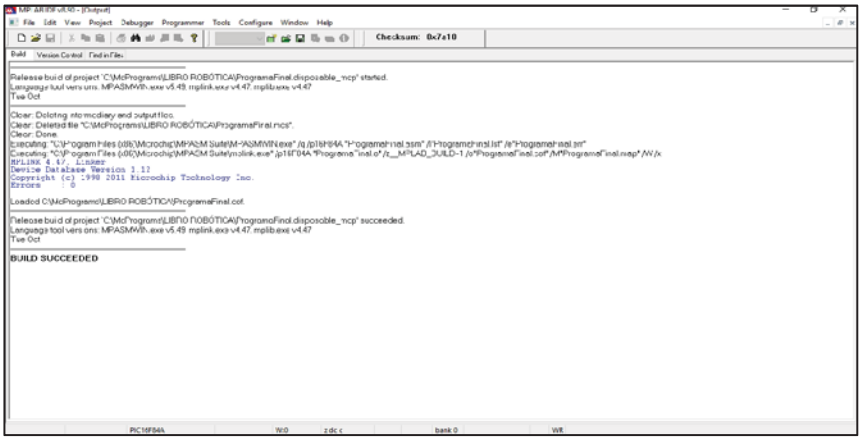


Figura 9.63. Compilación de programa.

Se aplican los pasos realizados a partir de la Figura 9.48. (Conexión del Programador PICkit3) para enlazar el programador y bajar el programa a la memoria del microcontrolador.

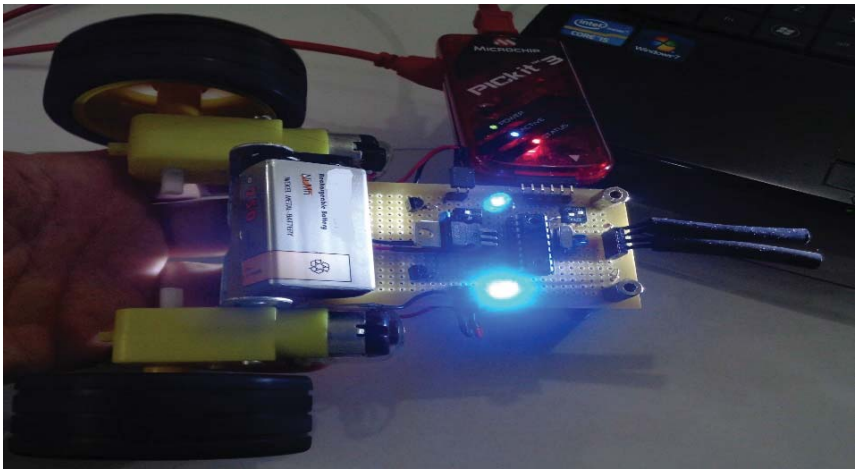


Figura 9.64. Programación de Robot vía ICSP.

Figura 9.65. Robot con µC en Assembler en funcionamiento.

CAPÍTULO 10
ARDUINO. OTRA FORMA DE HACER ROBÓTICA

10.1 INTRODUCCIÓN A ARDUINO

Una de las tecnologías que más se ha popularizado en los últimos tiempos es Arduino. Desde la experiencia personal, puedo decir que desarrollé el Primer Trayecto **Arduino - Aplicación y programación**, en el área de Formación Profesional, CFP 36 Parque Patricios, CABA, donde el Tercer Módulo – Modulo Senior, está orientado al desarrollo de Robots con Arduino.

Arduino es una plataforma o sistema, que integra software y hardware a nivel de placas y Shields o escudos, facilitando el acceso a aplicaciones muy potentes, como Ethernet, WiFi, Wireless, GSM, Motor, USB Host, Proto, Relays, Impresión 3D, entre tantos disponibles.

Ocurre también que luego de conocer sus recursos, surge esta pregunta ¿Qué diferencia existe entre emplear un Microcontrolador, tal cual se vio en el Capítulo 8, integrando además la electrónica, y el trabajo con Arduino?

La respuesta reside en que Arduino posee placas muy difundidas, como la Uno y Mega, con un montaje de producción industrial. Pero, en definitiva, emplean un microcontrolador de ATMEL, interconectando los pines de sus puertos, a través de los alojamientos hembra provistos en sus extremos. Es decir que se puede realizar con Arduino, aplicaciones similares a las que se han desarrollado

conociendo la electrónica y aprendiendo a programar un microcontrolador, en distintos lenguajes.

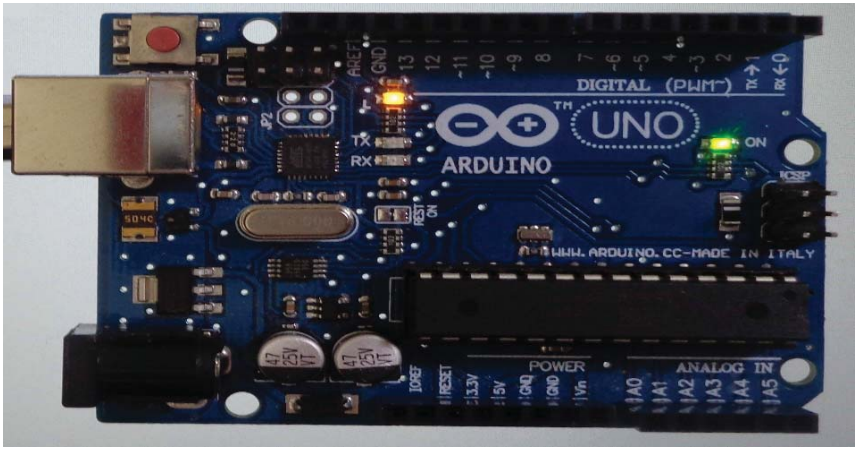


Figura 10.1. Placa Arduino Uno.

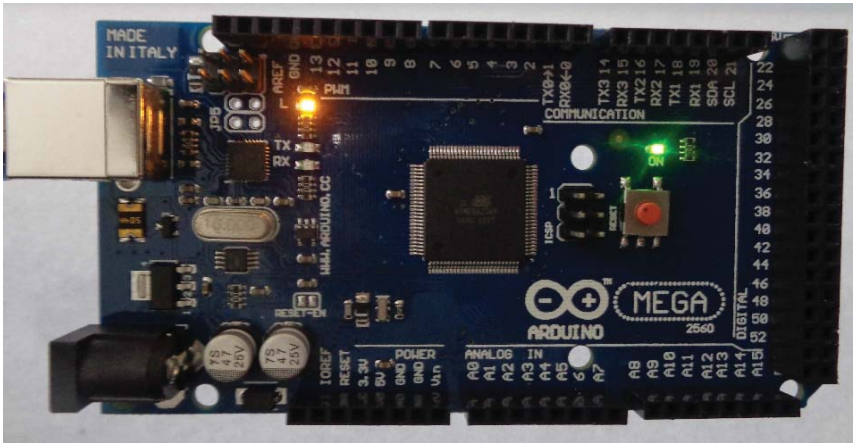


Figura 10.2. Placa Arduino Mega.

Uno de los grandes beneficios que ha introducido Arduino es el de posibilitar un acceso masivo a las aplicaciones de electrónica, de forma simple, y rápida, al aportar soluciones a nivel de software y hardware. Cuenta no solamente con soporte y desarrollo propio, sino con una comunidad a nivel mundial, de programadores, desarrolladores, estudiantes, hobbistas, activos en el crecimiento y divulgación del software – hardware libres, donde inclusive, Arduino tiene disponibles en la Web, los diseños de sus placas para ser descargados, para modificarlos, o hacer uso de ellos.

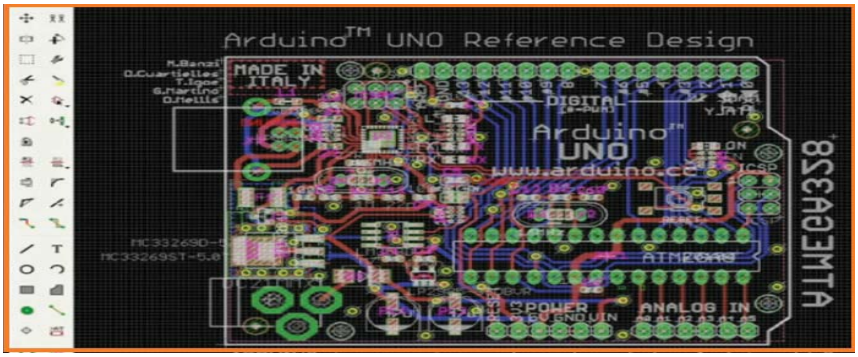


Figura 10.3. Placa Arduino Uno. PCB.

Se ve también en la siguiente figura que Arduino está desarrollado sobre un conjunto de tecnologías, cuyas bases son Processing y Wiring.

Figura 10.4. Componentes de código abierto, integrados en la tecnología Arduino.

Hardware	Software	Docs	Brand
CC-SA-BY	GPL	CC-SA-BY	TM

Figura 10.5. Elementos del sistema y categorías de sus licencias.

Descripción:

COMPONENTE	LICENCIA	ALCANCE
Hardware	CC BY-SA	Atribución-Compartirigual
Software	GPL	Licencia Pública General
Docs	CC BY-SA	Atribución-Compartirigual
Marca	TM	Marca Registrada

Figura 10.6. Licencias y alcance de los componentes del sistema.

CC BY-SA

Esta licencia permite crear sobre una obra existente o retocar, incluso con fines comerciales. Esto se puede hacer siempre y cuando reciba el autor original de la obra los respectivos créditos, y cuando las nuevas creaciones sean licenciadas bajo las mismas condiciones.

GPL

GPL es la sigla que refiere a Licencia Pública General. Es la licencia pública general más empleada y responde a las cuatro libertades del software propuestas por Richard Stallman:

0 – Libertad de usar el programa con cualquier propósito.
1 – Libertad de estudiar cómo funciona el programa y modificarlo.
2 – Libertad de distribuir copias del programa.
3 – Libertad de mejorar el programa y hacer públicas las mejoras.

Arduino website

En el sitio Web de Arduino, se puede encontrar información, novedades y productos.

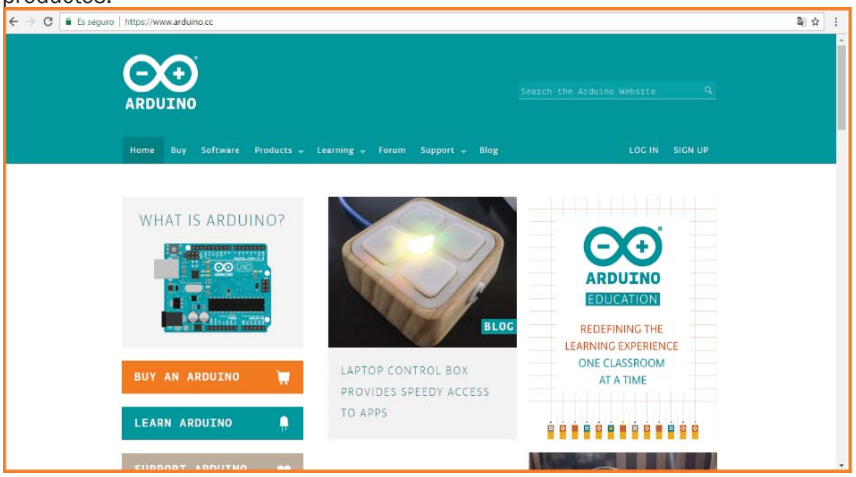


Figura 10.7. Arduino website.

También, en su Wiki Comunitaria en español.

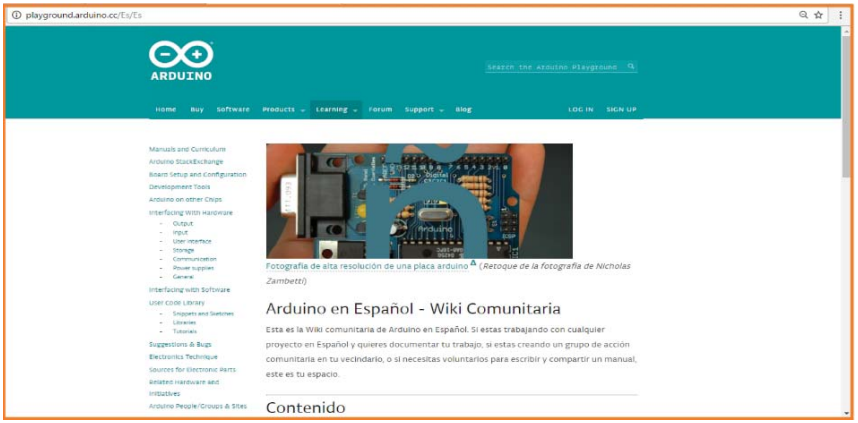


Figura 10.8. Wiki Comunitaria de Arduino en español.

10.2 PROCESSING

Processing es un software flexible, basado en lenguaje Java. Ha sido diseñado para aprender a programar, dentro del contexto de las artes visuales.

Java, a su vez, fue desarrollado por Sun Microsystems, y es un lenguaje de programación orientado a objetos, cuya sintaxis deriva de C y C++.

El IDE Processing puede ser descargado de forma gratuita de su *website*, ya que aplica a Open Source o Código Abierto.

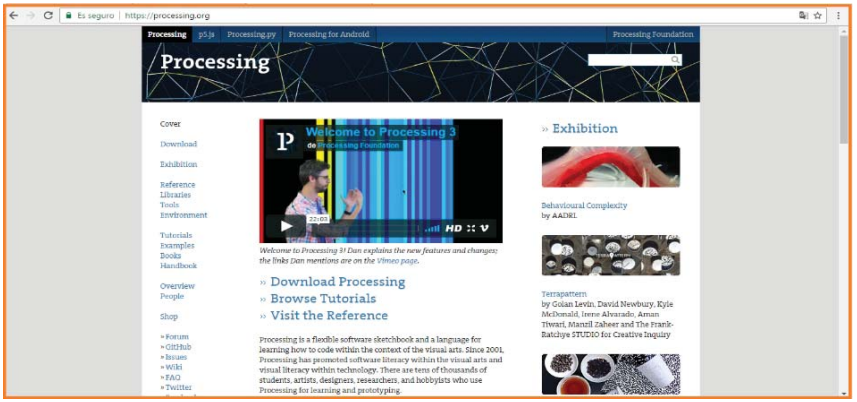


Figura 10.9. Processing website.

Una vez que se descarga y ejecuta el software, se ve la similitud que posee con el IDE Arduino, el cual está basado en Processing.

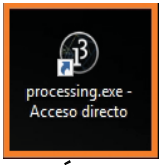


Figura 10.10. Ícono Processing.

Figura 10.11. Processing. Carga.

Figura 10.12. IDE Processing.

Ejemplo de programa:

Desarrollar código de programa en lenguaje Processing, que permita dibujar y mostrar una línea negra, similar a la empleada por nuestro **robot seguidor de línea negra**. La misma deberá estar centrada y ocupar la ¾ parte del alto de pantalla.

Código:

```
for (int i = 0; i < 75; i = i+1) {
line(47, i, 52, i);
```

```
}
```

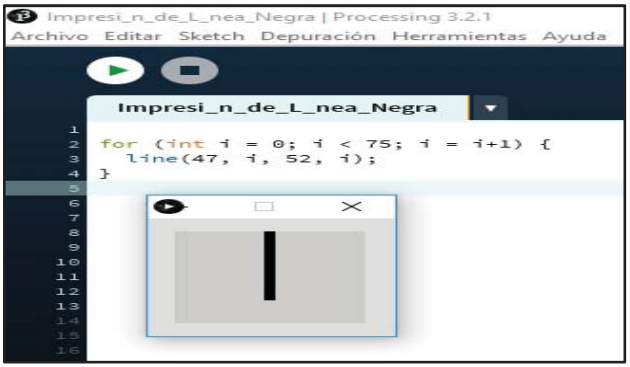


Figura 10.13. Ejemplo de programa en Processing. Código y ejecución.

10.3 WIRING

Respecto del hardware, Wiring es un sistema de programación de código abierto para microcontroladores.

Wiring permite escribir software multiplataforma para controlar dispositivos conectados a una amplia gama de placas de microcontroladores, y crear todo tipo de codificación creativa, objetos interactivos, espacios o experiencias físicas.

Al igual que ocurre con Arduino y Processing, Wiring está formado por una comunidad de expertos, estudiantes, diseñadores, artistas, quienes comparten ideas, conocimientos y su experiencia colectiva, para el aprendizaje, la creación de prototipos, y la producción profesional.

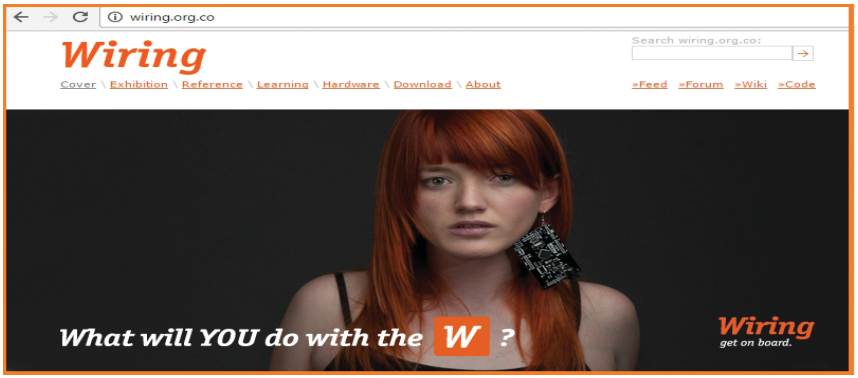


Figura 10.14. Wiring website.

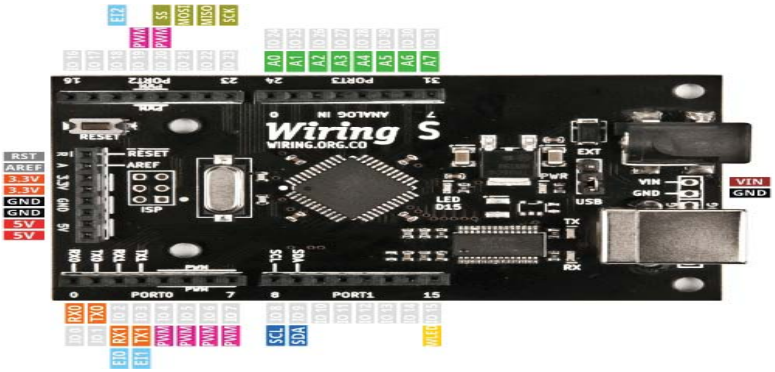


Figura 10.15. Hardware Wiring.

El microcontrolador de la placa puede ser programado en el entorno Wiring.

El hardware Wiring provee ISP y TWI (*Two Wire Interface*), o I2C, que permite conectar más de 128 sensores/actuadores, o una red superior a 127 tarjetas, empleando la librería Wire.

En el sitio Web de Wiring, en la sección descargas, hay disponible mucha información valiosa, relacionada, por ejemplo, con la interface del hardware a nivel de puertos. Tal es el caso de los drivers FTDI (*Future Technology Devices International*), compañía de semiconductores, necesarios para la emulación del puerto serie a USB mediante puertos COM virtuales.

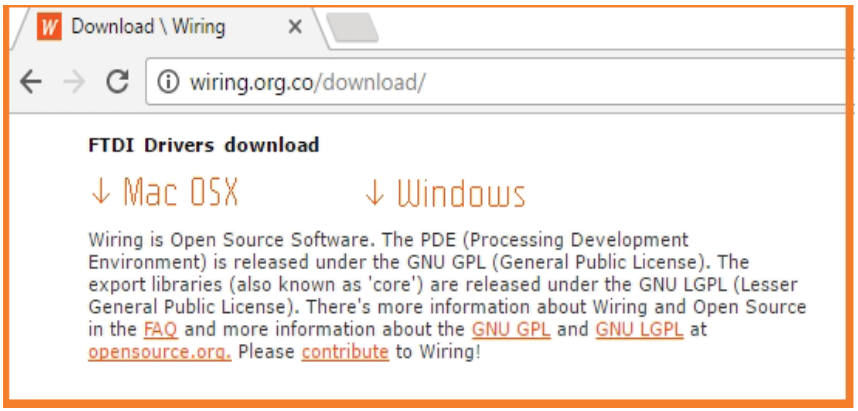


Figura 10.16. Wiring website: sección descargas.

Observe en la Figura 10.16 que en el texto original se destacan las características de *Open Source Software*, (GNU GPL), tratadas en esta obra.

IDE Wiring

El IDE Wiring está basado en el IDE Processing. Se observa en la Figura 10.17 la similitud entre dichos entornos. Esta suele ser una característica, entre los sistemas que aplican a Open Source.

A diferencia de free software, Open Source se focaliza más en la fase práctica de acceso al código fuente, que en cuestiones relacionadas con las libertades del software.

Figura 10.17. IDE Wiring.

```
1 void setup()
2 {
3
4 }
5
6 void loop()
7 {
8
9 }
```

Figura 10.18. Estructura de un programa en Wiring.

CÓDIGO Wiring

Se observa en la Figura 10.18, dado su basamento en Processing, la estructura de un programa en Wiring es similar, como se verá, a la que emplea Arduino.

Ejemplo de programa:

Desarrollar una subrutina para conectar un diodo LED en paralelo con el motor del lado izquierdo o derecho de un Robot, para indicar su activación.

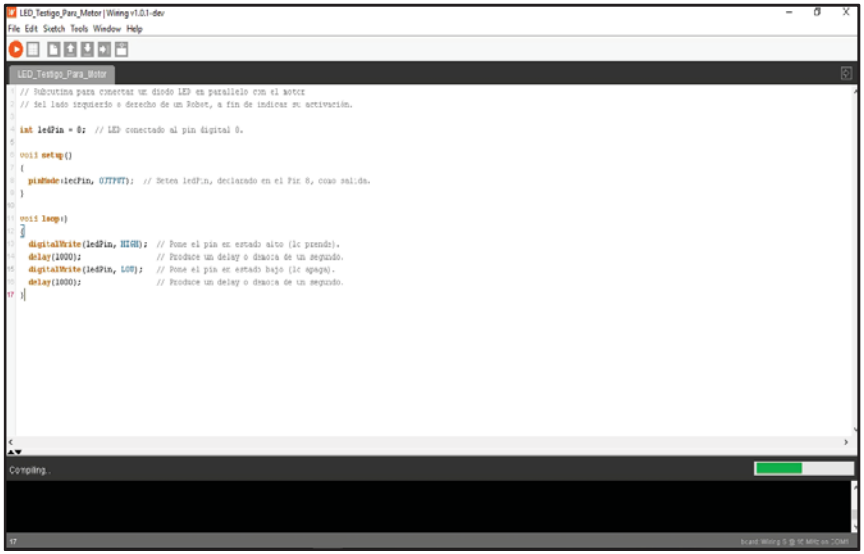


Figura 10.19. Programa LED_Testigo_Para_Motor. Codificación y Compilación.

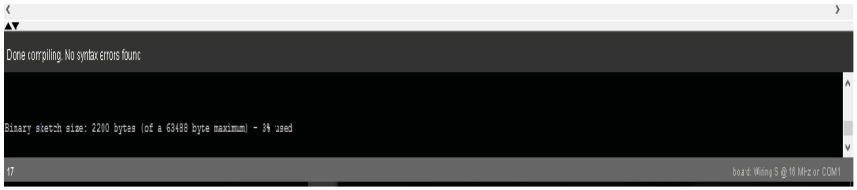


Figura 10.20. Programa LED_Testigo_Para_Motor. Compilación exitosa.

Si la sintaxis no presenta errores, el programa se compila. De esta manera, se genera el archivo ejecutable para descargar a la placa.

Se observa también que los lenguajes base del sistema son C, y C++.

Figura 10.21. Lenguajes de Programación bases del sistema.

10.4 IDE ARDUINO

Se vio en el ítem anterior, que el IDE Arduino, al igual que el IDE Wiring, está basado en Processing.

El IDE Arduino puede ser descargado de su sitio Web.

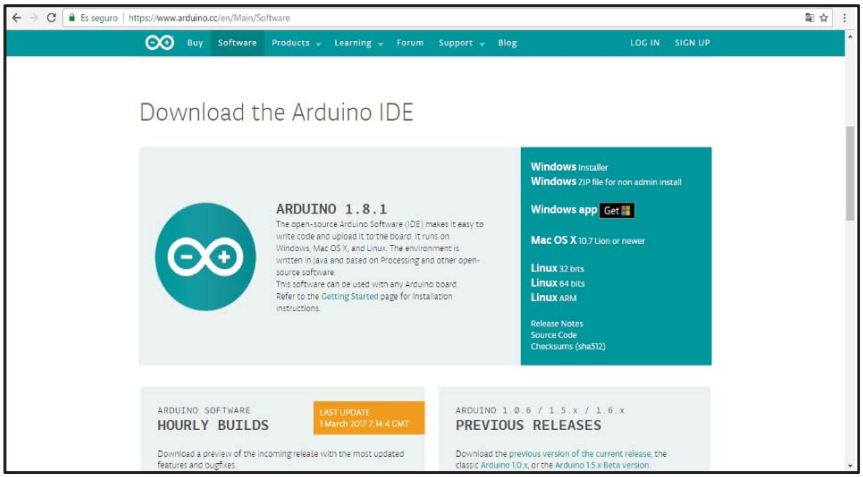


Figura 10.22. IDE Arduino. Descarga.

Una vez descargado el IDE, debe ser ejecutado; se genera también el ícono del programa en el escritorio.

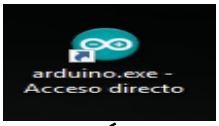


Figura 10.23. Ícono Arduino.

Se observa en la imagen de la Figura 10.22 que las versiones son actualizadas continuamente, introduciendo mejoras. Se recomienda realizarlas.

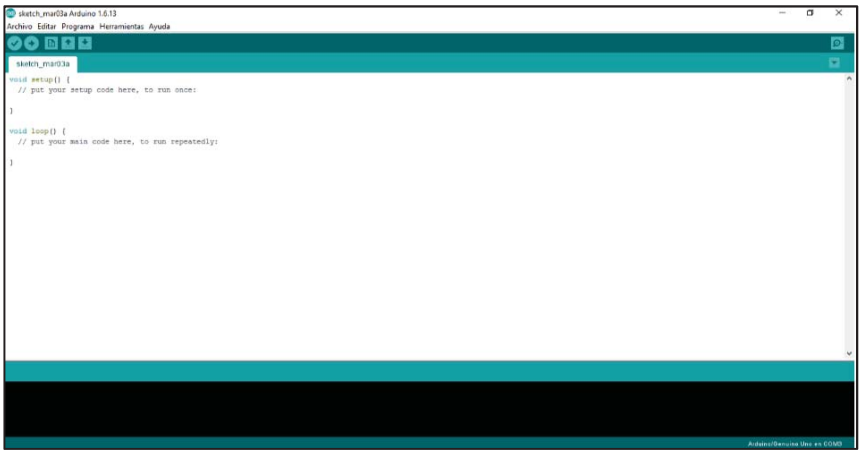


Figura 10.24. IDE Arduino.

Se observa entonces de la imagen anterior, la similitud con la Figura 10.17. IDE Wiring.

10.5 LENGUAJE ARDUINO

Antes de llegar a la placa, el entorno Arduino realiza algunas transformaciones para asegurarse de que el código fuente en C o C++ es correcto. Luego, interviene un compilador avr-gcc, para convertir el código a instrucciones máquina o código objeto. Seguidamente, el código se combina o enlaza con las bibliotecas estándar de Arduino, que proporcionan funciones básicas como `digitalWrite()` o `Serial.print()`.El resultado es un único archivo hexadecimal Intel, que contiene los bytes específicos a ser escritos en la memoria del microcontrolador de la placa en uso. Este archivo entonces se descarga a la placa, transmitido por el USB o conexión serie a través del gestor de arranque o bootloader existente en el microcontrolador, o con hardware de programación externo.

10.6 ESTRUCTURA DE UN PROGRAMA

Como se vio en la Figura 10.24, la estructura de un programa en Arduino es similar a la de un programa en Wiring.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figura 10.25. Estructura de un programa en Arduino. Vista ampliada.

Donde:

`Void setup`→ es empleado para las declaraciones.

La línea de comentario inserta en la estructura básica del programa indica: ingrese aquí su código de configuración para ejecutar una vez.

Debe recordarse del Capítulo 6 que la inclusión de archivos, declaraciones de variables, constantes, definición de constantes simbólicas, se realiza una sola vez, al igual que ahora, en la parte inicial del programa.

`Void loop` →es empleado para las instrucciones del programa que se repiten, siguen o monitorizan un proceso, estado o cambio de una variable.

La línea de comentario inserta en la estructura básica del programa indica: ingrese aquí su código de configuración para ejecutar repetidamente.

10.7 DESARROLLO DE PROGRAMAS

Una vez descargado, presentado el IDE y trabajada la estructura de un programa en Arduino, se está en condición de desarrollar nuestro primer proyecto. Por esto sería interesante tomar a modo de ejemplo la consigna propuesta para el IDE Wiring, de forma de ver la similitud en la sintaxis de su código.

Desarrollar una subrutina para conectar un diodo LED en paralelo con el motor del lado izquierdo o derecho de un Robot, a fin de indicar su activación.

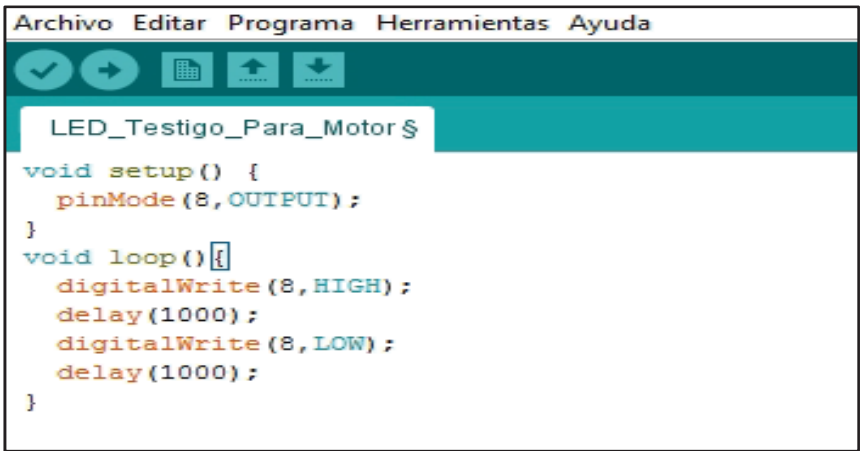


Figura 10.26. Código de programa en Arduino.

Una vez codificado o generado el programa hay dos opciones. La primera es compilar para verificar que no haya errores de sintaxis en el programa (Figura 10.27). La segunda opción, de estar conectada la placa con el hardware correspondiente al proyecto, es seleccionar la opción subir, de modo de cargar el archivo `.hex` (Figura 10.28).

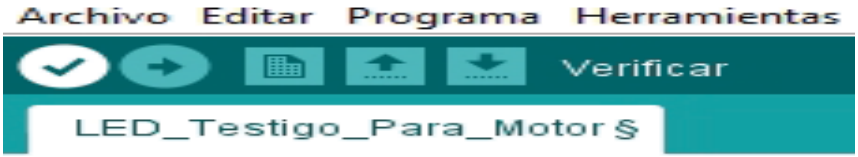


Figura 10.27. Opción `Verificar` programa.

Figura 10.28. Opción `Subir` archivo a la placa.

Figura 10.29. Compilando programa LED_Testigo_Para_Motor.

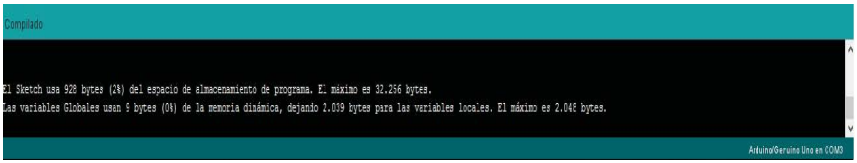


Figura 10.30. Si el programa no posee errores. Se genera el archivo `.hex`.

Se observa entonces, comparando con la Figura 10.19, la similitud entre ambos códigos y sus IDE basados en Processing.

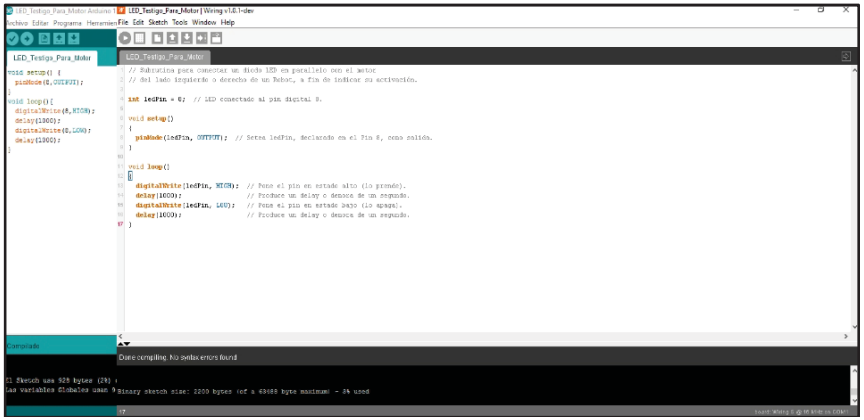


Figura 10.31. Similitud entre el mismo código generado en Arduino y Wiring.

Se simula, antes de descargar el programa a las placas Uno Y Mega, la existencia de un error en la sintaxis del código, a fin de conocer la respuesta del sistema.

Si se observan las líneas de código, se reconoce que se ha borrado la letra D, a la palabra reservada `DigitalWrite`.

```
LED_Testigo_Para_Motor §
void setup() {
  pinMode(8,OUTPUT);
}
void loop(){
  igitalWrite(8,HIGH);
  delay(1000);
  digitalWrite(8,LOW);
  delay(1000);
}
```

Figura 10.32. Simulación de error en la sintaxis del programa.

Si se compila el programa, se obtiene el siguiente mensaje de error.

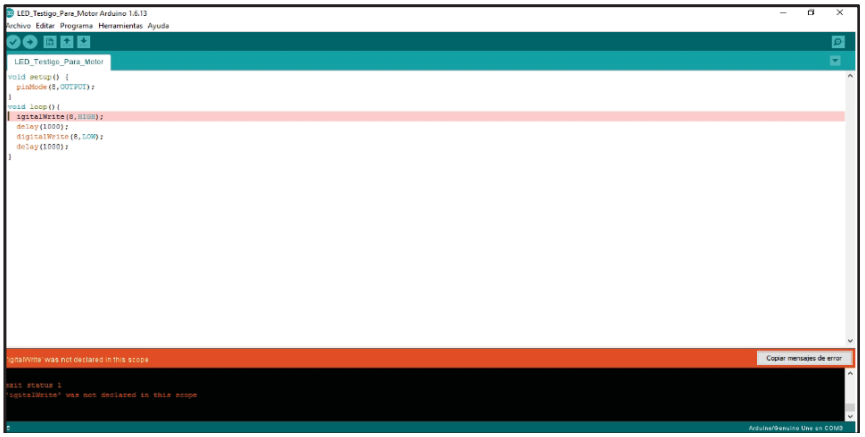


Figura 10.33. Respuesta del sistema ante un error de sintaxis.

En la Figura 10.33, aparece marcada la línea que presenta el error de sintaxis. Al pie de página, se indica lo trabajado respecto a **el código se combina o enlaza con las bibliotecas estándar de Arduino, las cuales proporcionan funciones básicas como** `digitalWrite();` por tanto, el código introducido no es reconocido.

```
'igitalWrite' was not declared in this scope

exit status 1
'igitalWrite' was not declared in this scope

5
```

Figura 10.34. La palabra tipeada no es reconocida.

10.8 PLACAS ARDUINO

Ahora se va a trabajar en la descarga del archivo a la placa. Para esto, se van a emplear los dos modelos de placas expuestos al inicio del capítulo, la Uno y Mega. Pero debe recordarse también la existencia de otros modelos, con diferentes tamaños y prestaciones, a fin de adaptarse a las necesidades particulares de nuestros proyectos.

Para personas que se inician en esta actividad, o para aquellos que desean profundizar la manipulación y el contacto con las actividades de hardware, una buena práctica antes de descargar el programa LED_Testigo_Para_Motor, es realizar la prueba de las placas. Para esto, se puede emplear el diodo LED en SMD, que poseen asociado al pin 13, modificando simplemente la aplicación de la siguiente forma:

```
Testeo_de_placa_con_LED_pin_13 §
void setup() {
  pinMode(13,OUTPUT);
}
void loop(){
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
}
```

Figura 10.35. Programa para testear la placa empleando el LED del pin 13.

Figura 10.36. Compilando y descargando programa de prueba a la placa.

Generalmente, la primera vez que se conecta la placa al sistema, se espera encontrar el modelo ideal o *plug and play*, donde enlazar con el IDE, y ver la ejecución de la aplicación. Pero lo más común, inicialmente, es recibir este aviso de error de enlace.

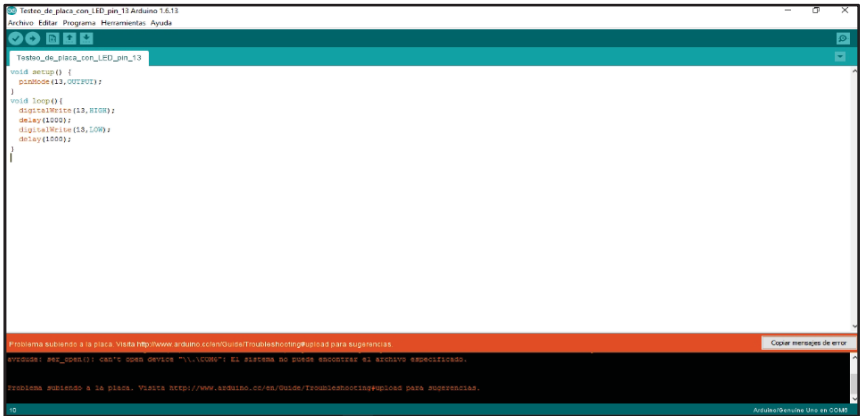


Figura 10.37. Aviso del sistema: `No se puede conectar con la placa`.

Figura 10.38. Error de enlace. Referencia.

Para solucionar este problema, lo primero que se puede hacer es acceder a la solapa `Herramientas` y verificar en la opción `Placa` si coincide con la que se ha conectado al puerto de nuestra PC; en este caso, la Uno.

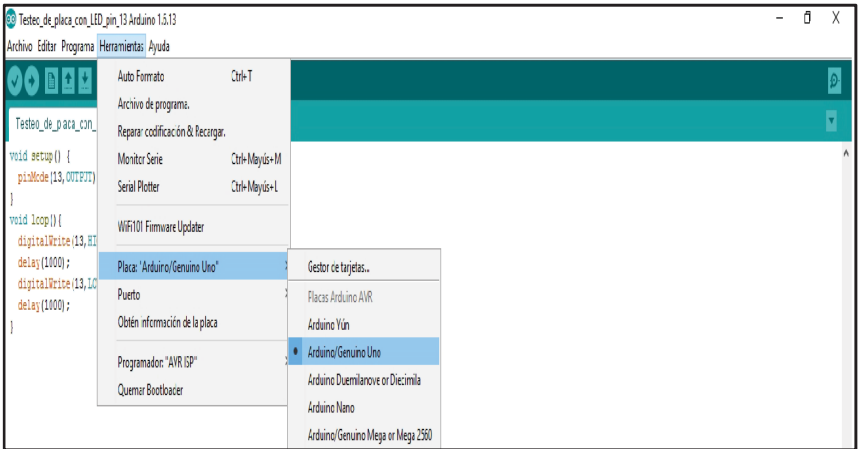


Figura 10.39. Selección de placa a emplear.

Si como ocurre en este caso, la placa configurada en el sistema coincide con la placa a emplear, lo que se debe hacer es chequear la configuración del puerto.

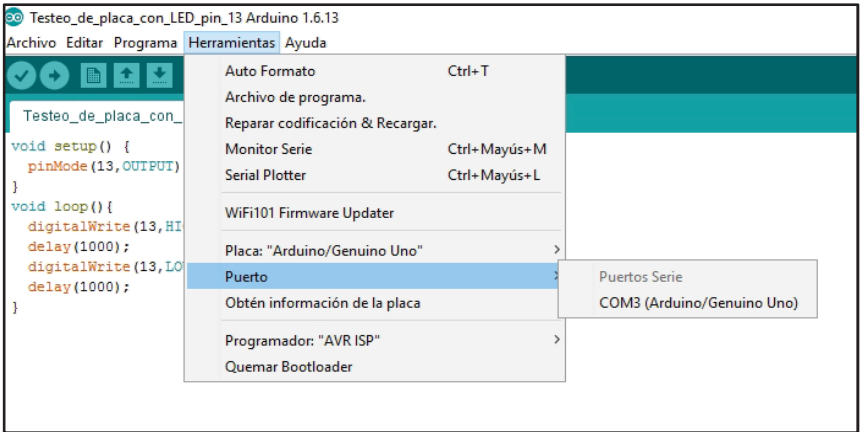


Figura 10.40. Selección de puerto para conexión.

Se observa en la Figura 10.40. que no existe ninguna opción de selección de puerto, seteada o establecida, pero en la línea inferior aparece en negrita el COM3, disponible para esta conexión de placa.

Figura 10.41. Opción de conexión al COM3.

Se obtiene luego el enlace o conexión con la placa.

Figura 10.42. Enlace con la placa y descarga del programa

De esta forma, el programa cargado se ejecuta en la placa.

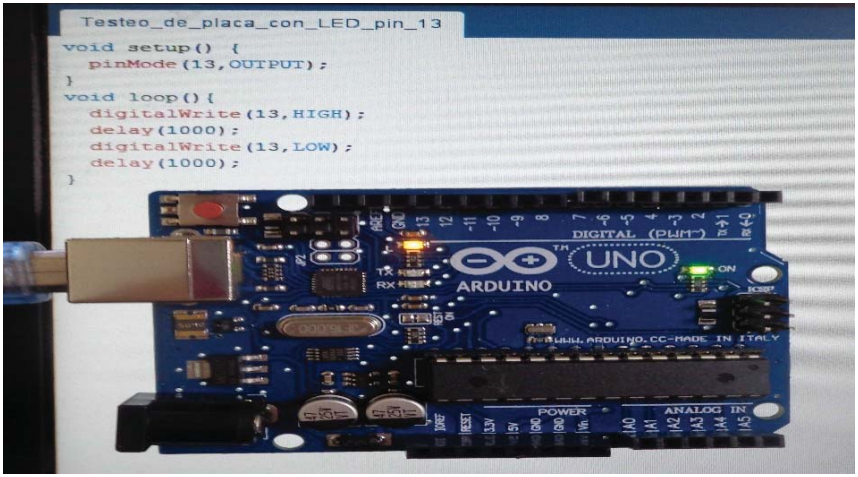


Figura 10.43. Descarga y ejecución de programa en la placa Arduino Uno.

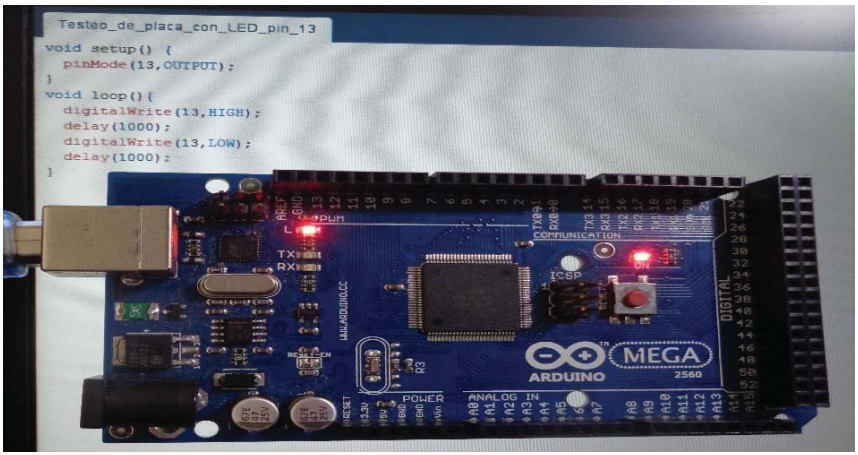


Figura 10.44. Descarga y ejecución de programa. Placa Arduino Mega 2560.

10.8.1 ARDUINO UNO

La placa ARDUINO Uno (Figura 10.43) emplea el Microcontrolador ATMEL, ATmega328P. De los 28 pines, 14 pueden ser configurados como digitales de entrada / salida (6 como PWM), 6 entradas analógicas, cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación, una tira de pines macho para conexión ICSP (Programación Serial en Circuito), botón de reinicio.

Un gran beneficio de esta placa, para quienes ya han dado un paso en la Electrónica, es la posibilidad de emplearla como programador. Luego extraer el microcontrolador, para controlar una aplicación.

Figura 10.45. Extracción del microcontrolador de la placa.

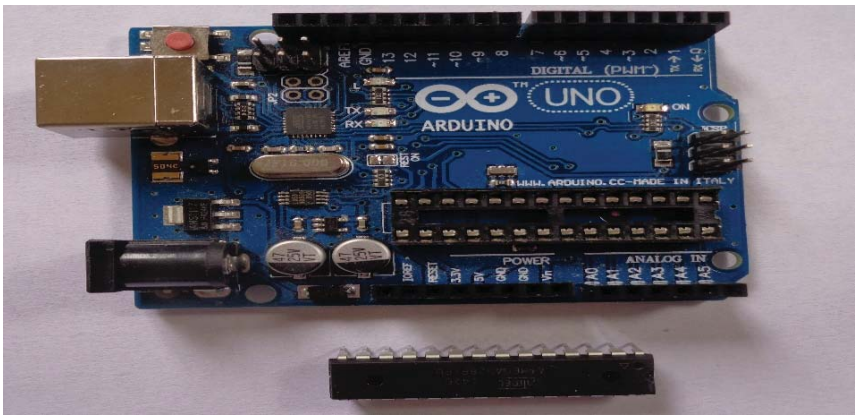


Figura 10.46. El Microcontrolador puede ser trasladado a un circuito externo.

ESPECIFICACIONES TÉCNICAS – PLACA ARDUINO UNO

ESPECIFICACIONES TÉCNICAS

Voltaje de operación	5 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límite)	6-20V
I por pin como entradas/salida	20 mA
I por pin 3.3 V	50 mA

Figura 10.47. Especificaciones técnicas de V y I - Arduino Uno.

10.8.2 ARDUINO MEGA 2560

La placa Arduino Mega 2560 (Figura 10.44) posee un microcontrolador ATMEL, ATmega2560, con 54 pines digitales configurables como entrada/salida. De los 54, 15 pueden ser empleados como salidas PWM, 16 entradas analógicas, 4 UARTs (hardware de puertos serie), cristal oscilador de 16 MHZ, una conexión USB, un tomacorriente, una tira de pines macho para conexión ICSP (Programación Serial En Circuito), botón de *reset*. La placa es compatible con los shields diseñados para la Uno.

ESPECIFICACIONES TÉCNICAS – PLACA ARDUINO MEGA

ESPECIFICACIONES TÉCNICAS

Voltaje de operación	5 V
Voltaje de entrada (recomendado)	7-12 V
Voltaje de entrada (límite)	6-20 V
I por pin como entradas/salida	20 mA
I por pin 3.3V	50 mA

Figura 10.48. Especificaciones técnicas de V y I - Arduino Mega 2560.

10.9 SHIELDS ARDUINO

Los shields o escudos de Arduino brindan una respuesta rápida y práctica para el desarrollo de aplicaciones con distintos niveles de complejidad.

Debe recordarse que las aplicaciones que representan los Shields pueden desarrollarse, aplicando los conocimientos adquiridos en los Capítulos 2, 3, 4, 6, 7 y 8.

Ejemplo: shield módulo relé Arduino.

Figura 10.49 Shield Arduino relé.

Recuérdese que los relés son de gran aplicación en robótica, ya que permiten obtener un efecto similar a la conmutación de un transistor, cada vez que la bobina de entrada recibe un valor de tensión, acorde a su tensión de trabajo.

A este mismo relé, se lo puede implementar también adquiriendo los componentes de forma discreta (por separado), en una casa de electrónica.

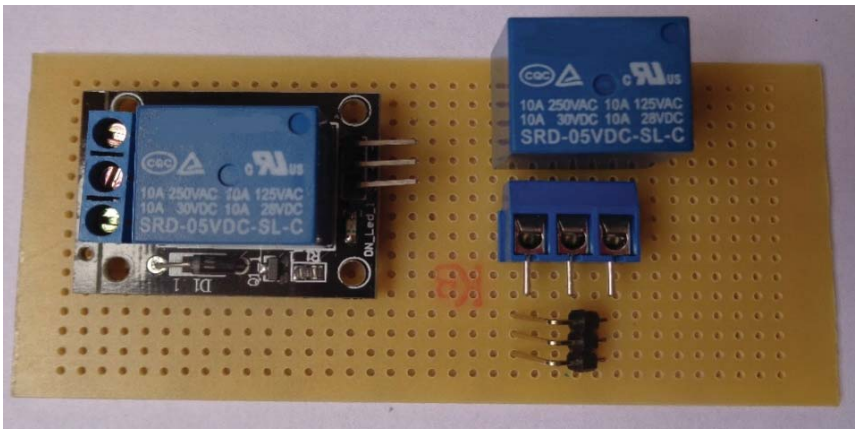


Figura 10.50. Shield Arduino relé. Componentes discretos.

Se puede obtener la lista de componentes a emplear para el montaje del shield módulo relé, observando la Figura 10.50 donde se incluyen:

Shield - módulo Relé Arduino – Componentes.

Relé 5 VCC
Bornera para placa de 3 terminales
Pines para placa a 90 grados
Corte de placa experimental o multipropósito
Cable de *wire wrapping* para conexiones

Figura 10.51. Shield Arduino relé. Listado de componentes

10.10 FRITZING SOFTWARE DE DISEÑO DE PCB

Fritzing no es un software de simulación de circuitos, pero es muy empleado en la actualidad dada su aplicación en Arduino.

Fritzing responde también al paradigma GNU/GPL. Por esto se puede iniciar la descarga de su website, a fin de ejercitar, diseñar e implementar el PCB de un circuito.

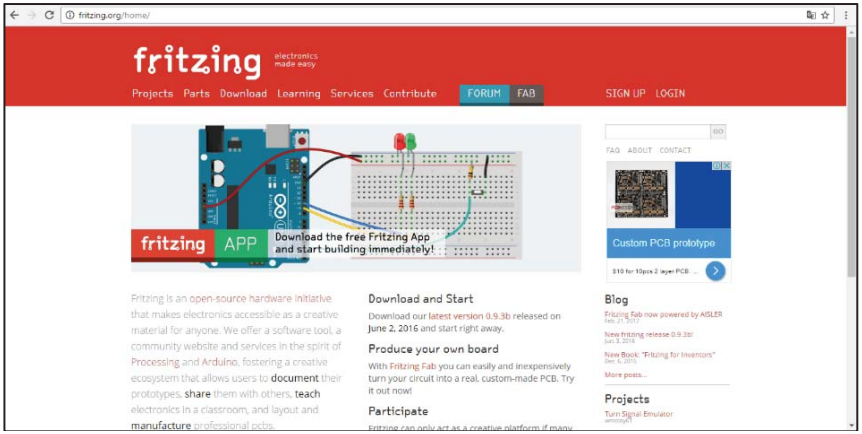


Figura 10.52. Fritzing website.

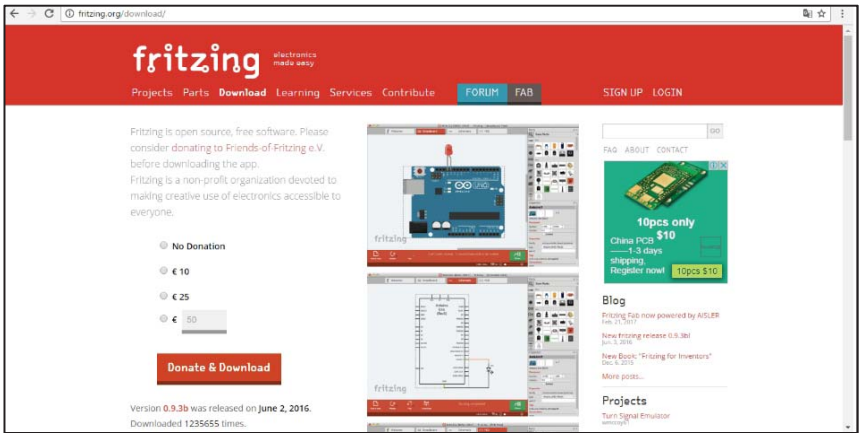


Figura 10.53. Fritzing. Descarga desde su website.

Una vez realizada la descarga del software, se lo instala y se ejecuta.

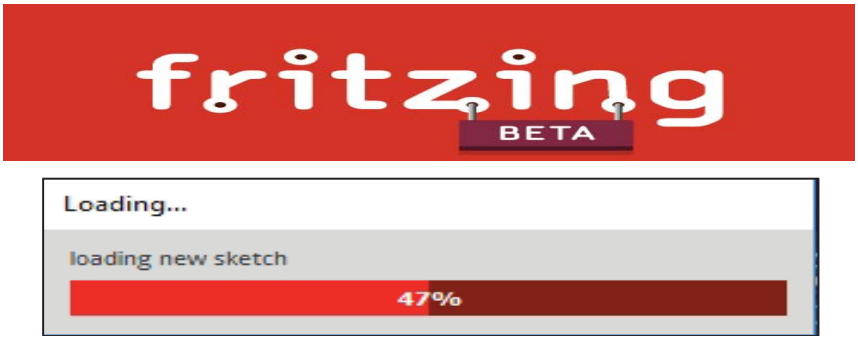


Figura 10.54. Carga del programa.

Recuérdese que, en caso de ya poseer el software, se deben realizar también las actualizaciones presentes, a fin de incorporar nuevas funciones y componentes.

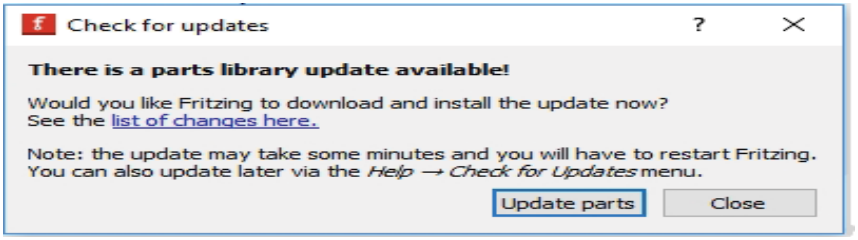


Figura 10.55. Fritzing. Actualización.

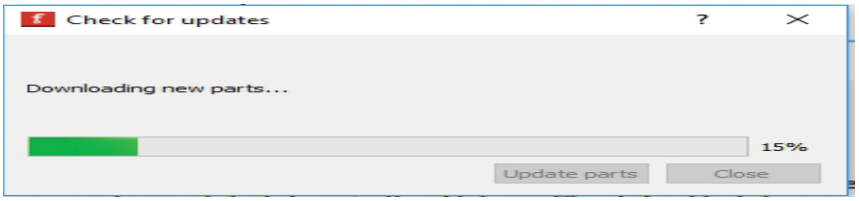


Figura 10.56. Fritzing. Actualización. Proceso.

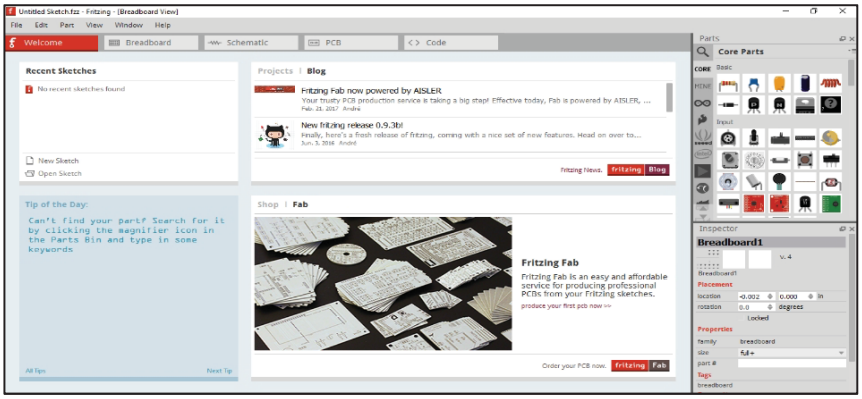


Figura 10.57. Fritzing – Entorno – Bienvenida.

Para iniciar la simulación de circuitos, se puede seleccionar la opción `Breadboard` (protoboard), donde aparece la siguiente pantalla.

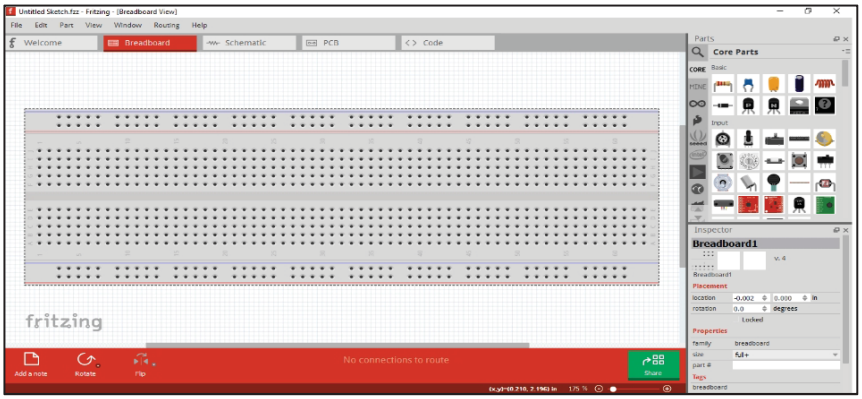


Figura 10.58. Fritzing. Área de trabajo.

Ejemplo:

Implementación de circuito eléctrico. Robot seguidor de luz.

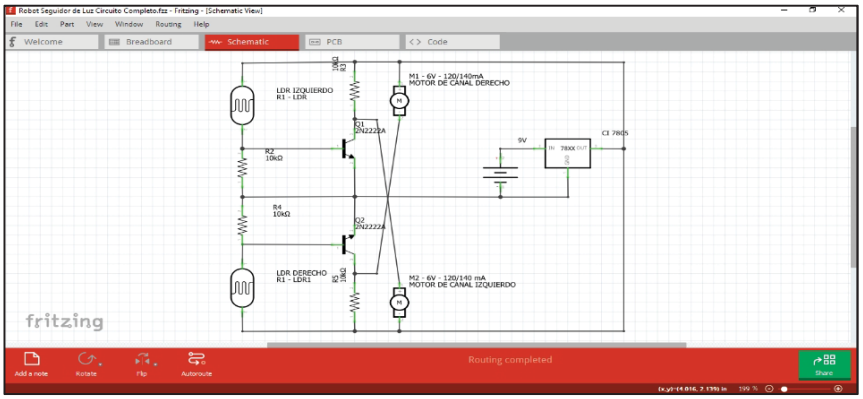


Figura 10.59. Fritzing. Diseño de circuito de robot seguidor de luz.

10.11 APLICACIONES DE ROBÓTICA CON ARDUINO

10.11.1 INTRODUCCIÓN

Las opciones directas para trabajar robótica empleando Arduino son tres:

- Arduino: robot Arduino.
- Robots compatibles con Arduino.
- Desarrollo de sus propios robots, empleando placas Arduino.

A continuación, se desarrollarán estas opciones.

10.11.2 ARDUINO ROBOTS

Se puede ver en su website el Arduino Robot, primer robot oficial sobre ruedas de Arduino.

Este robot emplea dos procesadores: uno para la junta de motor y otro para la junta de control. Ambos están basados en el Microcontrolador ATmega32U4.

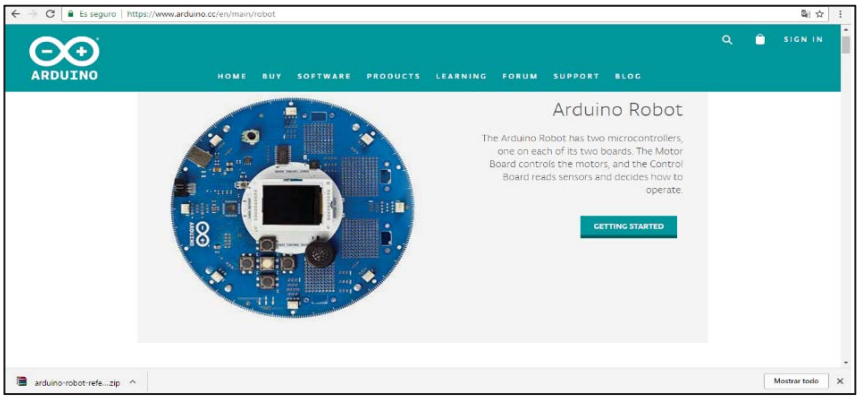


Figura 10.60. Arduino Robot.

Como característica de Arduino, toda la información sobre este robot está disponible en su website.

10.11.3 ROBOTS COMPATIBLES CON ARDUINO

Otra opción interesante para trabajar robótica con Arduino es el empleo de robots de fabricantes que lo desarrollan con tecnología compatible con Arduino. En la mayoría de los casos, los fabricantes ensamblan sus propias placas, basadas en la placa Arduino Uno.

Figura 10.61. Alicia Rodríguez y Ezequiel Schvarzman. Robots compatibles Arduino.

Se deben tener en cuenta a la hora de adquirirlos detalles como la incorporación del set de destornilladores correspondientes y accesorios acordes a la descripción del montaje en la caja.

Figura 10.62. Mirta Arredondo y Víctor Cuba. Robots compatibles Arduino.

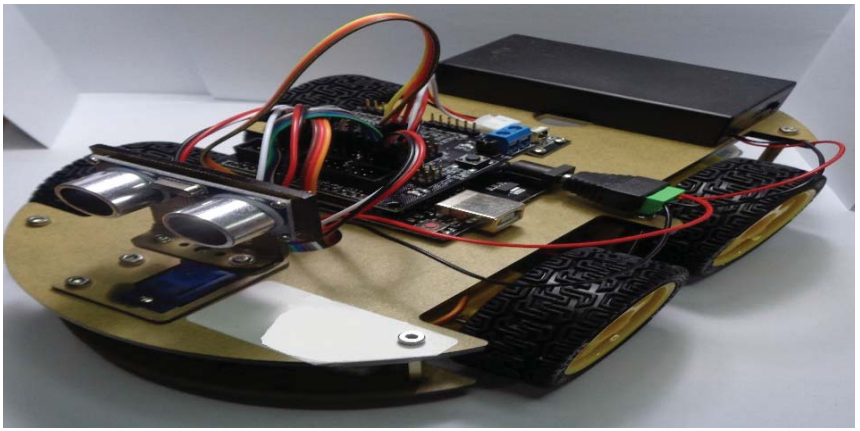


Figura 10.63. Robot compatible Arduino.

10.11.4 DESARROLLO EMPLEANDO ARDUINO

Otra posibilidad de implementar Robótica con Arduino es emplear el chasis desarrollado para el robot seguidor de luz, y realizar un montaje a su elección.

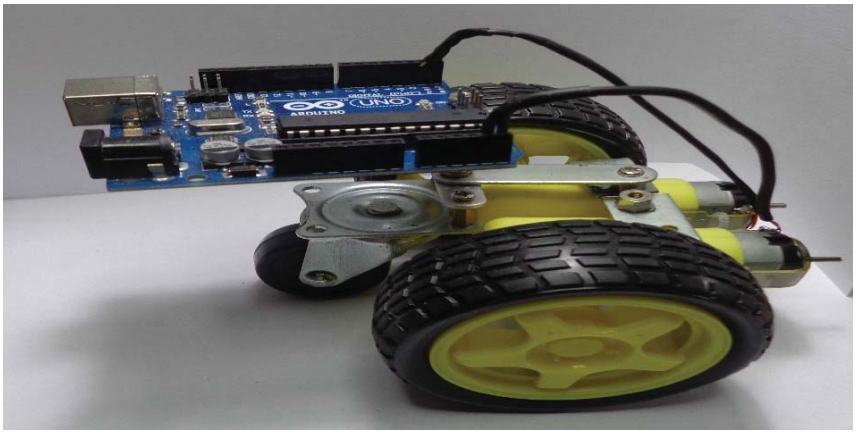


Figura 10.64. Implementación de robots con tecnología Arduino.

De esta forma, se puede también utilizar la placa para controlar las señales provenientes de los sensores, y en lugar de transistores para conmutar, puede responder a estas señales con niveles lógicos.

Se debe tener la precaución de utilizar la entrada de alimentación (posee regulador de tensión), en lugar de la entrada USB, debido a que esta entregará el valor provisto por la fuente de V: es decir, 9 V, y de esta forma, produce la rotura de los motores de 6 V.

Como se ha presentado, son variadas las opciones de implementación combinando tecnología e ingenio.

CAPÍTULO 11
IMPRESORAS 3D - APLICACIÓN EN ROBÓTICA

11.1 INTRODUCCIÓN

El modelado e impresión 3D es junto con Arduino una de las grandes tecnologías encargadas de impulsar una nueva revolución en las formas de diseño, desarrollo y producción.

Su aplicación está presente en todas las áreas del conocimiento y la industria. En la robótica aporta las posibilidades de:

Modelar e imprimir piezas, como las llantas y cubiertas.
Modelar e imprimir chasis, con posibilidad de personalizar nuestros proyectos.

Figura 11.1. Aplicaciones del 3D en robótica.

Arduino, por ejemplo, ha dado un paso más respecto a Shields para la implementación de impresoras 3D, introduciendo en el mercado su primer modelo oficial: MATERIA 101.

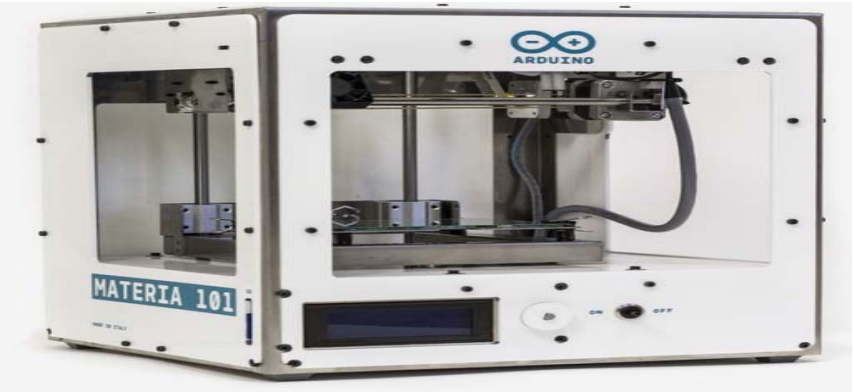


Figura 11.2. Impresora 3D – ARDUINO MATERIA 101 (foto propiedad Arduino).

En la actualidad existen muchos desarrollos de aplicaciones de tipo Open-Source Hardware, que aplican al desarrollo de Impresoras 3D; incluso proyecciones para la tecnología, como las máquinas autorreplicantes de Von Neumann. Estas máquinas pueden ser definidas como robots o autómatas autorreplicantes. Entre los proyectos destaca RepRap una impresora 3D de escritorio, capaz de imprimir piezas de plástico, material con el que están diseñadas muchas de sus partes.

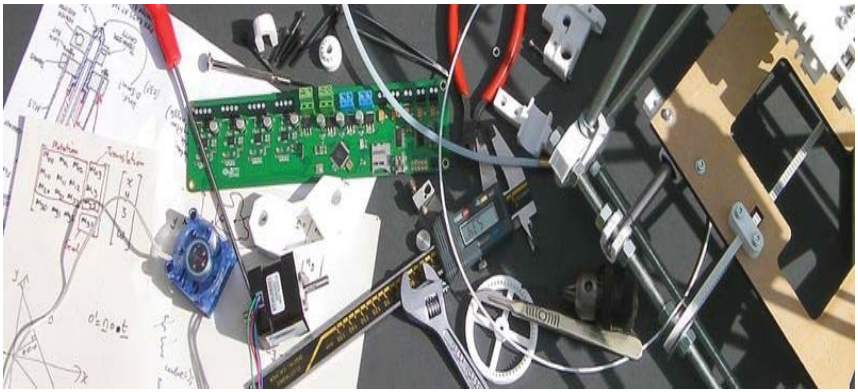


Figura 11.3. Impresora RepRap – Imagen propiedad RepRap.

11.2 TIPOS Y CARACTERÍSTICAS

11.2.1 CARTESIANAS

Este tipo de impresoras reciben su denominación del movimiento en ejes cartesianos, realizado para su trabajo. El movimiento se traduce en tres ejes

ortogonales, donde la parte final del extrusor, llamada HotEnd, que funde el filamento y lo deposita sobre la zona de impresión, se desplaza en las coordenadas X, Y, Z.

En este tipo de impresoras, la cama (zona para impresión) puede levantarse o bajar, dependiendo del modelo.

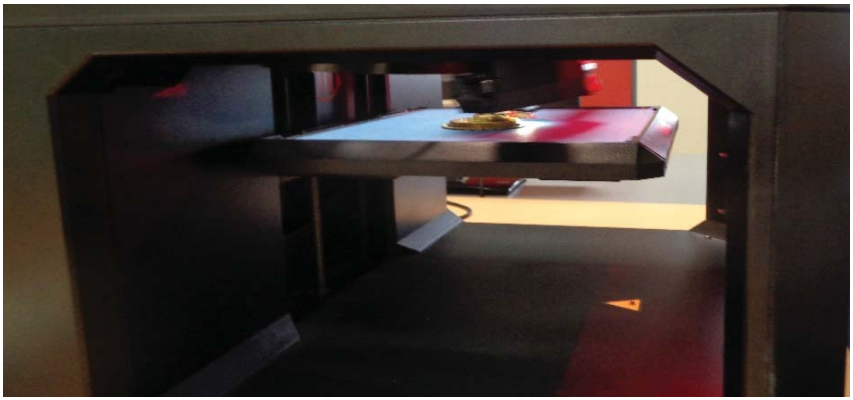


Figura 11.4. Impresora 3D Cartesiana de 2 ejes. Cama levantada en impresión.

La cama caliente es una placa que va montada sobre la cama. Es necesaria cuando se emplean distintos materiales como PCB, ya que la diferencia de temperatura entre el HotEnd (extrusor) y la superficie de la cama provoca que el material se despegue o se combe. Sin embargo, en el trabajo con material tipo PLA, no se requiere cama caliente.

11.2.2 DELTA

Este tipo de impresoras están basadas en el robot paralelo.

Por lo general, su cama de impresión es circular y fija. En este caso, el extrusor está sostenido desde arriba por tres brazos articulados, que se deslizan verticalmente sobre sus guías o ejes. De esta manera, se logra el posicionamiento deseado en el espacio de impresión.

Figura 11.5. Impresora 3D – Delta.

Ventajas de este tipo de impresoras: velocidad de impresión, definición, redireccionamiento, facilidad de ensamble de piezas y montaje, menor tamaño, bajo coste.

11.3 ELECTRÓNICA - HARDWARE

Una particularidad es que la electrónica de base es común para la mayoría de las impresoras 3D, compuestas por la placa Arduino trabajada en el apartado 10.8.2 (Arduino Mega 2560), Shield Ramps 1.4, Drivers Pololu; y para el software residente en la placa, Marlin.

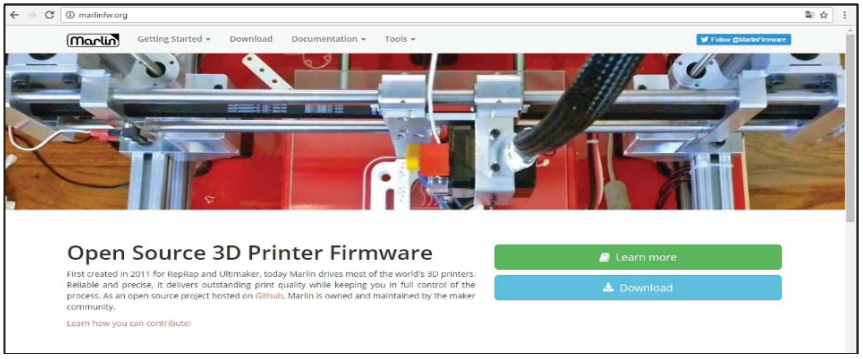


Figura 11.6. Marlin website.

Marlin firmware reside en la placa principal de la impresora 3D y se ejecuta cada vez que la máquina está encendida. Su función es gestionar todas las

actividades en tiempo real de la máquina. Coordina los calentadores, motores paso a paso, sensores, luces, pantalla LCD, botones, y todo lo que necesita hacer como parte del proceso de impresión 3D.

11.4 TIPOS DE ARCHIVOS DE TRABAJO

A modo de clarificar el proceso, se pueden diferenciar los siguientes 4 pasos:

PASO 1	PASO 2	PASO 3	PASO 4
MODELADO	REVISIÓN	G-code	IMRESIÓN
Dibujo 3D – CAD **En AutoCAD se genera la extensión:** `.dwg`	Se exporta a STL: .stl	Generación de G-code	G-code necesario para imprimir

Figura 11.7. Pasos para el modelado e impresión en 3D.

Como se observa en la tabla, el primer paso es modelar la pieza en un programa de CAD. Luego se exporta a STL, y se genera el G code. Por último, se pasa con éste a la fase de impresión.

ARCHIVO STL

Una vez realizado el modelado de la pieza en una herramienta CAD, se debe exportar a STL, a partir del cual se genera en el paso descrito como 3, el G-code necesario para imprimir el archivo, de acuerdo con las características de la impresora 3D.

STL es un acrónimo de STereoLitography. El nombre de estos archivos termina con la `extensión .stl`; en el ejemplo, el archivo generado sería `llanta.stl`.

Este formato fue desarrollado para el trabajo con impresoras 3D; es un estándar de archivo abierto ampliamente utilizado en la fabricación asistida por ordenador (CAM) y en el prototipado rápido. El formato de archivo STL es bueno para imprimir partes, porque describe la geometría de la superficie de una pieza. En la práctica, se sabe que los archivos STL pueden ser importados a herramientas CAD (*Computer Aided Design* – Diseño Asistido por Ordenador), pero resulta muy difícil su modificación.

Los pasos para exportar el archivo STL desde AutoCAD son:

`Menú → Exportar → Otros Formatos → Salvar entrada` (nombre de archivo) y en tipo de archivo `litography (*.stl) → Guardar`.

Por lo general, en el acceso a la primera impresora, es común descargar el STL de algún objeto de interés, en sitios como Thingiverse.

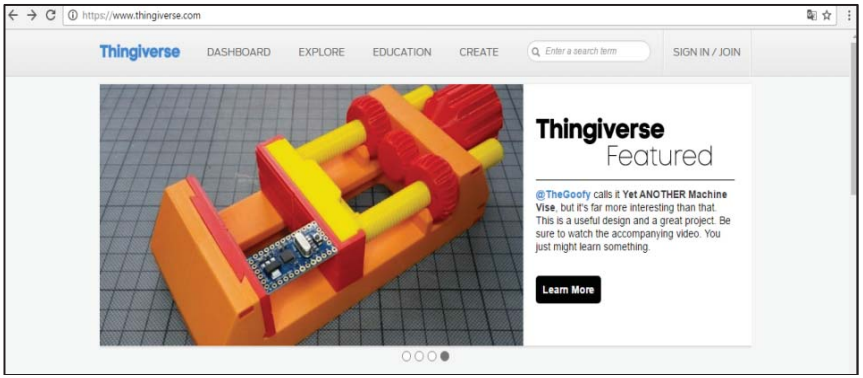


Figura 11.8. Thingiverse website.

G-code

Se vio también en la Figura 11.7 que, como tercer paso, se debe generar el archivo G-code, a partir del STL.

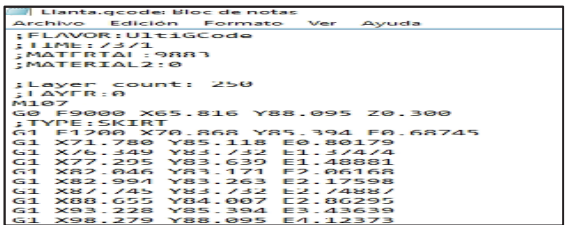


Figura 11.9. Parte superior del G-code para el archivo `Llanta.gcode`.

De forma simple se puede definir que el G-code es a los cabezales de la impresora 3D, lo que el binario a los procesadores.

El G-code es entonces un lenguaje de programación, empleado en el control numérico computarizado o CNC, para controlar tornos, fresadoras, e impresoras 3D.

Cada instrucción transmite a los cabezales, información sobre coordenadas, velocidad, trayectoria. Es decir que en el código G, se introducen todos los parámetros que se necesitan para poder imprimir una pieza, como: tipo de material, temperatura del pico, diámetro de extrusor, altura de capas (resolución).

Esto se lleva a cabo a través de programas como Cura.

11.5 SOFTWARE EMPLEADO EN IMPRESIÓN 3D

El software para impresión 3D responde a las categorías ya trabajadas, de propietario o copyright, y libre bajo el paradigma GNU/GPL.

Como software propietario, uno de los más empleados es Autocad.

En la categoría Open Source, se emplea blender, KISSlicer, Cura, Slic3r.

blender

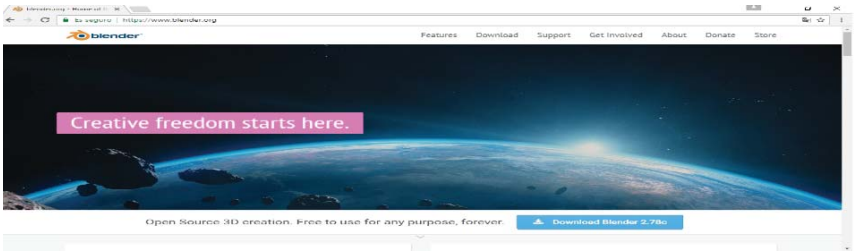


Figura 11.10. blender website.

Blender es una suite de creación 3D gratuita y de código abierto, empleada para modelado, animación, simulación, renderizado, edición de vídeo, y creación de juegos.

KISSlicer

Figura 11.11. KISSlicer website.

KISSlicer es un programa de fácil empleo y multiplataforma, que toma archivos 3D (STL) y genera información de ruta (código G) para una impresora 3D.

La versión gratuita está diseñada para impresoras de un solo cabezal; la versión PRO está diseñada para el trabajo en impresión con múltiples cabezales.

Cura

Cura nos permite preparar el modelo para la impresión 3D, simplificando la actividad inicial; esto permite obtener buenos resultados. También, ofrece a los expertos más de 200 ajustes para adaptarse a sus necesidades.

Figura 11.12. Cura website.

Slic3r

Slic3r es un motor de rebanado 3D de software libre para impresoras 3D y genera código G de archivos CAD 3D (STL u OBJ). Una vez terminado, se envía a la impresora 3D un archivo de código G apropiado para la producción de la pieza u objeto modelado en 3D, para la fabricación de un objeto físico

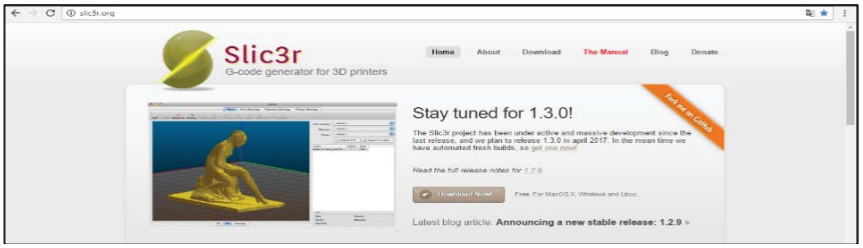


Figura 11.13. Slic3r website.

11.6 DISEÑO E IMPRESIÓN DE COMPONENTES Y CHASIS

En el trabajo en robótica, resulta frecuente encontrar limitación respecto a componentes y accesorios para el diseño.

Por el contrario, es común poseer ruedas y motores, que se obtienen del desguace de equipos y dispositivos. Sin embargo, estos no se pueden emplear, ya que los diámetros de sus ejes no coinciden con las ruedas.

Ante esta situación, se recurre al mercado para obtener aquellos modelos disponibles, al margen de su calidad, rendimiento, o características eléctricas y de potencia.

La impresión 3D posibilita diseñar o modelar piezas que se adapten a las necesidades y requerimientos de nuestros proyectos. De esta forma, se pueden adquirir motores de mayor potencia de trabajo y también desarrollar nuestras ruedas.

Es preciso recordar que no es lo mismo, emplear un motor para un robot seguidor de luz o línea, que para una aplicación de sumo a nivel competencia.

EJEMPLO DE APLICACIÓN - `llanta – modelado`.

Se tomará como base los citados ejemplos, respecto a la incompatibilidad de diámetros entre ruedas y motores, y se describirá un proceso para modelar una llanta.

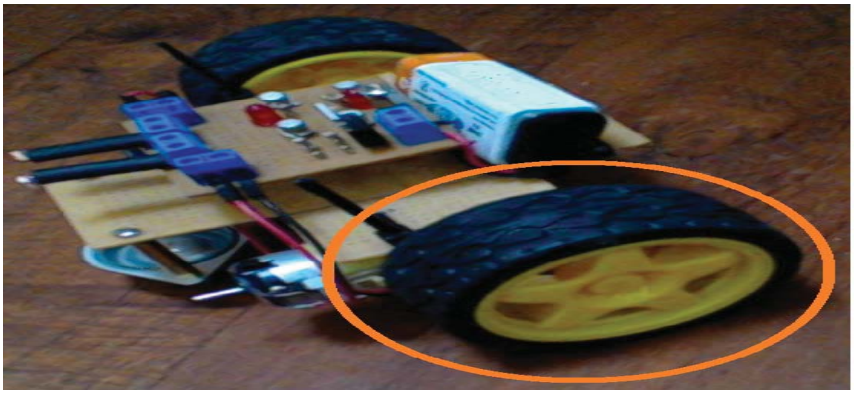


Figura 11.14. Rueda empleada en robot seguidor de luz.

Figura 11.15. Vistas de llanta en la rueda empleada para modelar.

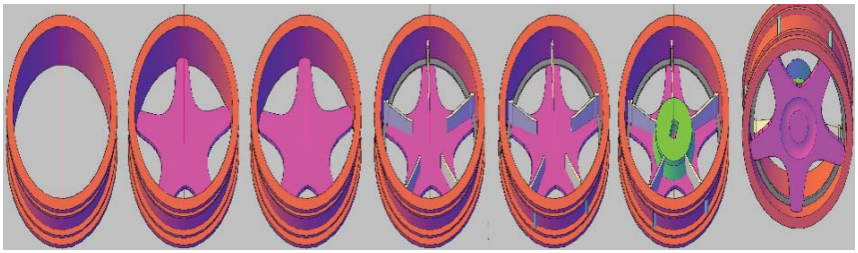


Figura 11.16. Modelado de llanta. Se proyecta imprimir en dos partes.

MATERIAL EMPLEADO PARA IMPRIMIR

El material del filamento empleado es plástico, entre ellos ABS, y PLA.

Para este trabajo de diseño, se emplea PLA o ácido poliláctico. PLA es un polímero (poliéster termoplástico) biodegradable, proveniente de vegetales como el maíz, el trigo, y la caña de azúcar entre otros.

FASE DE DISEÑO

Se diseña en AutoCAD el modelo de la pieza.

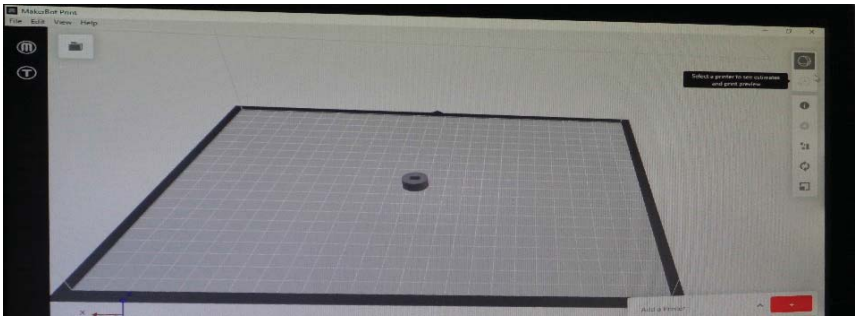


Figura 11.17. Fase de Diseño en AutoCAD.

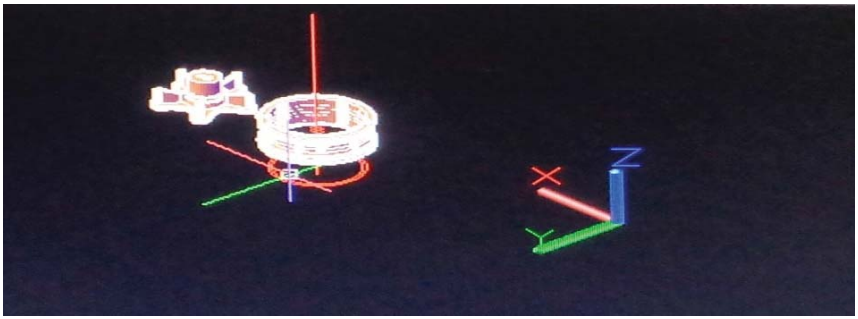


Figura 11.18. Fase de Diseño – llanta en dos partes.

Como se expuso al inicio del capítulo, uno de los beneficios para la aplicación del modelado e impresión 3D en robótica, es la posibilidad de adaptar los diámetros de motores disponibles, a nuestras implementaciones de llantas, (Figura 11.19.)

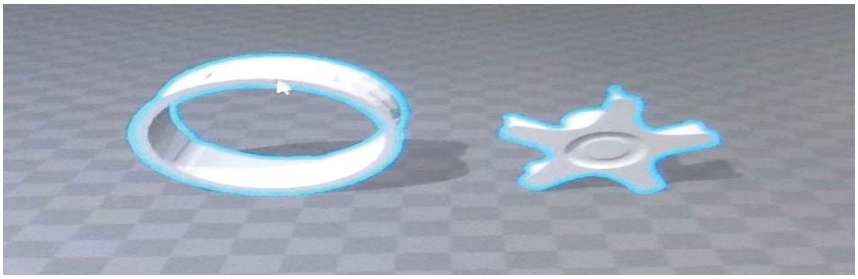


Figura 11.19. Llanta vista frontal.

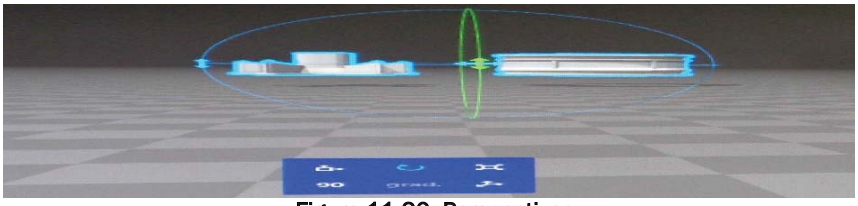


Figura 11.20. Perspectivas.

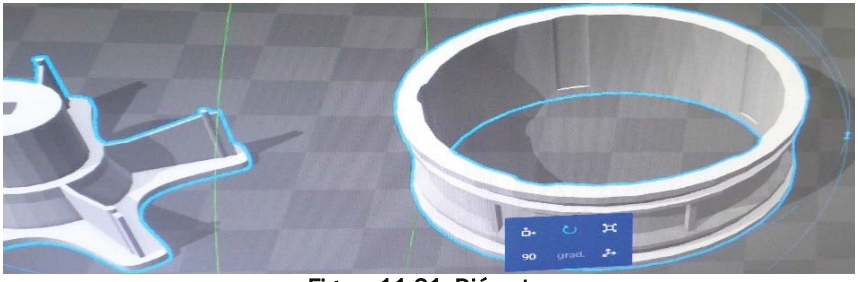


Figura 11.21. Diámetros.

Figura 11.22. Llanta. Parte interior.

Figura 11.23. Texturas. Parte interior.

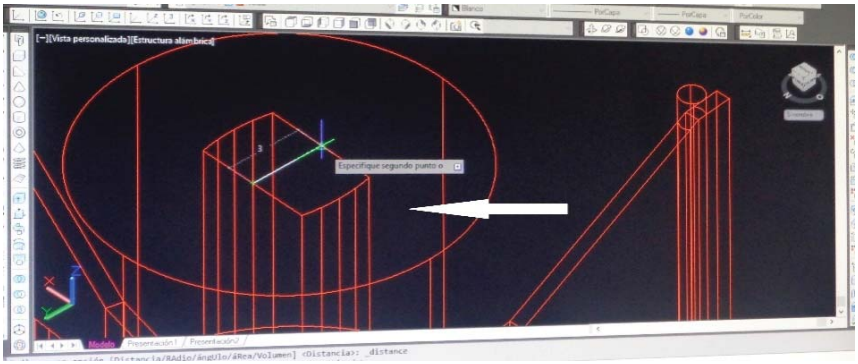


Figura 11.24. Diseño: Cavidad de encastre para eje de motor.

FASE DE IMPRESIÓN

Figura 11.25. Configuración de parámetros en impresora 3D.

Figura 11.26. Extrusor. Parámetros iniciales.

Figura 11.27. Regulación de extrusor a 215 grados.

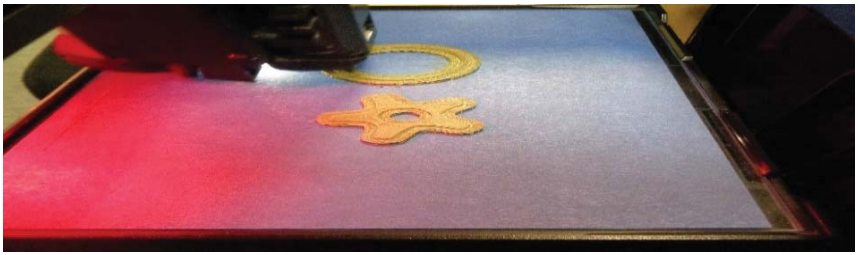


Figura 11.28. Inicio de impresión.

TIEMPO DE IMPRESIÓN

El tiempo de impresión dependerá del tamaño y detalles del modelo. Para el ejemplo de diseño de llanta está indicado en 02:33:09 (tiempo transcurrido + tiempo restante). Tamaño de archivo: 4,297 KB.

Figura 11.29. Indicación en display de tiempo transcurrido y restante.

CABEZALES O JET PRINTERS

El número de cabezales de la impresora determinarán, por ejemplo, la cantidad de colores que se puedan emplear para estos proyectos.

Es posible también controlar por software la tonalidad de cada color.

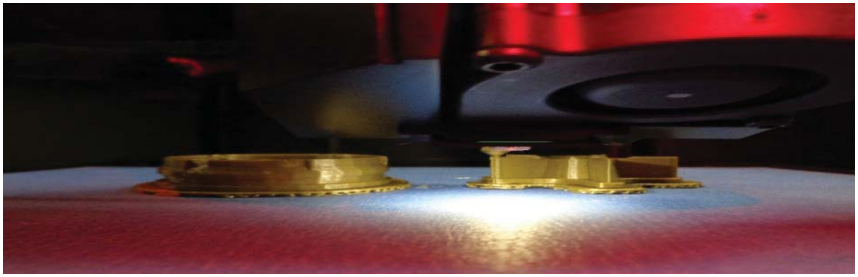


Figura 11.30. Cabezal de impresión.

Figura 11.31. Fin de proceso.

Figura 11.32. Llanta impresión en 2 partes.

Figura 11.33. Llanta. Vista frontal.

Figura 11.34. Llanta. Vista trasera.

TAMAÑO DE IMPRESIÓN

Si bien para este proyecto se trabajó con una pieza pequeña, se debe tener en cuenta, como referencia, la limitación de tamaño para impresión que se indica en función de los ejes x, y, z. Ejemplo 150 x 150 x 150 mm.

OTRAS PIEZAS O PARTES TRABAJADAS

Del mismo modo que se ha modelado y trabajado en la llanta de rueda, se pueden obtener también todo tipo de piezas para diseñar en robótica, tales como: chasis, contenedores de batería, adaptadores para diámetros de ruedas y un sinfín de partes, en función de la necesidad y la creatividad.

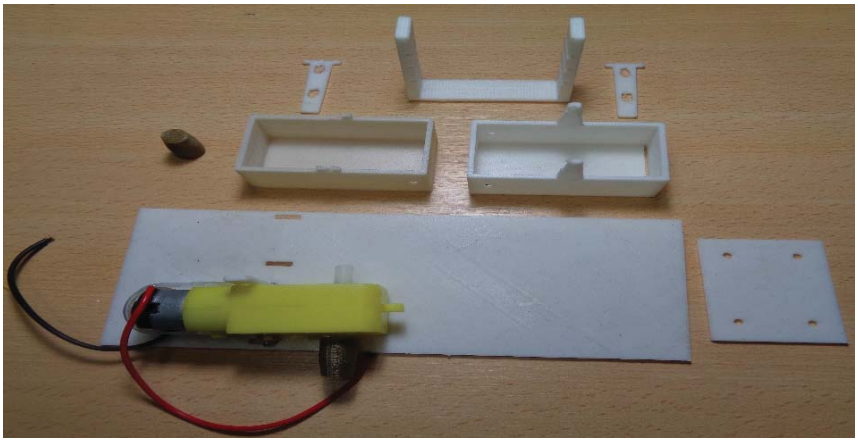


Figura 11.35. Piezas modeladas para el diseño de robots.

CAPÍTULO 12
ROBÓTICA AVANZADA - FPGA

12.1 FPGA – INTRODUCCIÓN

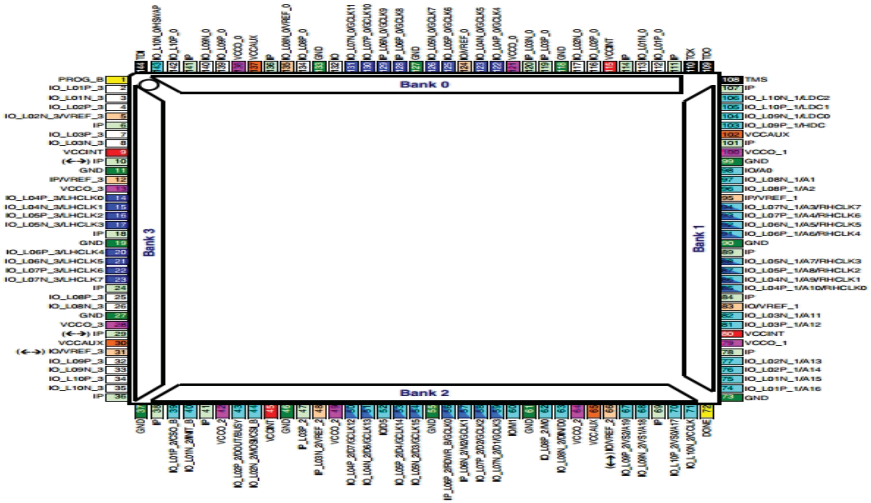


Figura 12.1. Spartan 3E – 100. *Pin out*.

En el Capítulo 7 se expuso que:

No nos hemos de explayar en ítems como el ***fan out*** de 10, o cantidad de dispositivos que se pueden cargar a la salida de una compuerta lógica, ya que, en una instancia de diseño, emplearíamos un Microcontrolador o una FPGA, antes de emplear tantos CI de forma discreta o por separado.

Luego, en el Capítulo 8, se inicia, nuestro trabajo con microcontroladores. Y se cierra este último (capítulo), con la presentación de los dispositivos llamados FPGA, con los que, dependiendo del modelo de placa (o integrado) a disposición, se cuenta con cientos de miles de compuertas lógicas, para modelar un CI, que responda a las necesidades y características de nuestro proyecto.

FPGA es un acrónimo de *Field Programmable Gate Array* o *Array* de Compuertas Programables en Campo.

Estos dispositivos son una gran alternativa para diseñar y aplicar robótica en una instancia superior.

A modo de introducción y ejemplo, vamos a diseñar o modelar, uno de los primeros CI digitales con los que hemos trabajado en el Capítulo 7, una compuerta AND (3 entradas para el proyecto).

12.2 FPGA – PLACA A EMPLEAR

Para nuestro trabajo de modelado en este capítulo, vamos a emplear la placa Basys 2 de Digilent.

La placa Basys2 es una plataforma de diseño e implementación de circuitos que cualquier persona puede utilizar para adquirir experiencia en la construcción de circuitos digitales reales. Construido alrededor de una matriz de puerta programable de campo Xilinx Spartan-3E y un controlador USB AT90USB2 de Atmel, la placa Basys 2 proporciona un hardware completo y listo para usar, adecuado para alojar circuitos que van desde dispositivos lógicos básicos hasta controladores complejos. Se incluye una gran colección de dispositivos de E/S en la placa y todos los circuitos de soporte de FPGA necesarios. De esta manera, se pueden crear innumerables diseños, sin necesidad de otros componentes.

12.3 IDE - ISE

ISE Design Suite integra las herramientas necesarias para nuestro proceso de diseño, en una interfaz gráfica fácil de usar, para lógica programable. Al estar orientado a los dispositivos de Xilinx, proporciona un entorno completo para el desarrollo óptimo de aplicaciones.

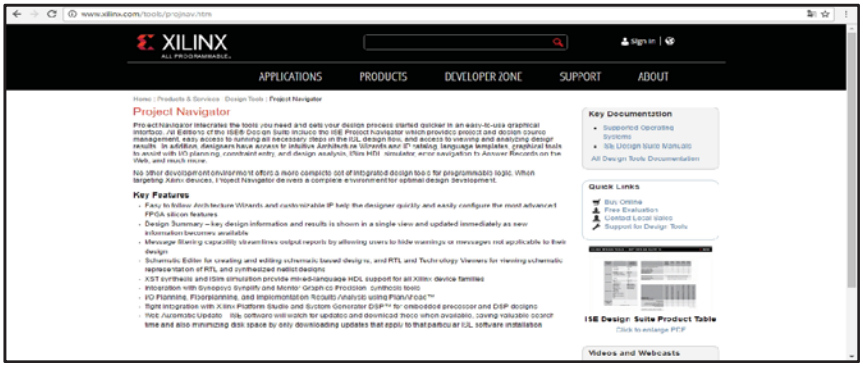


Figura 12.2. Xilinx link y Project Navigator.

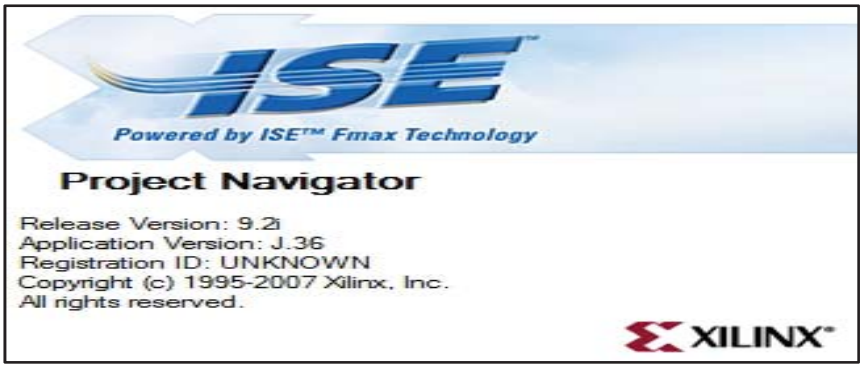


Figura 12.3. ISE Project Navigator.

12.4 DESCARGA DEL PROGRAMA

Al descargar el programa, es importante que se realice la ejecución desde el directorio `C\`.

12.5 CREACIÓN DE UN PROYECTO

Luego, dentro de la carpeta `XILINX92i` en `C:\` se crea la carpeta `project`, en la cual se guardan los proyectos.

Primer paso:

`Se debe seleccionar – New Project →`

`Será solicitado un nombre, ejemplo Gate3In → Next.`

```
C:\Xilinx92i\projects\Gate3In
```

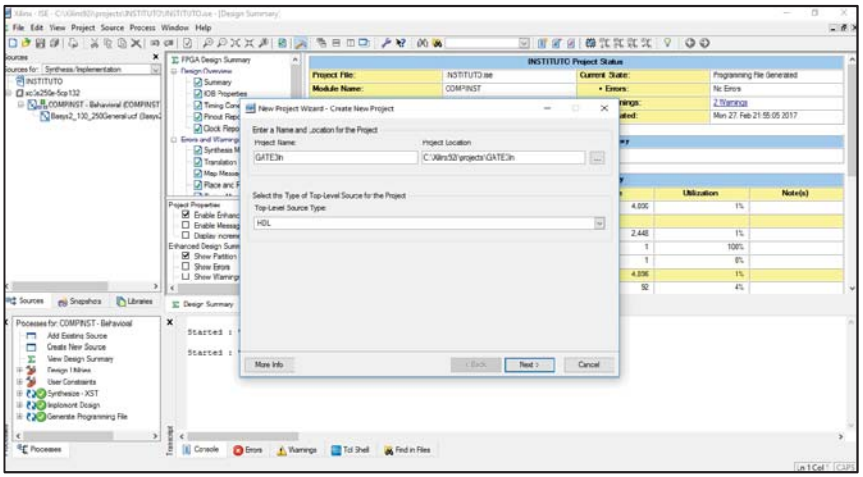


Figura 12.4. Creación de proyecto.

Se observa en la Figura 12.4 la creación de un nuevo proyecto y sobre el margen superior izquierdo, la lista de proyectos existentes.

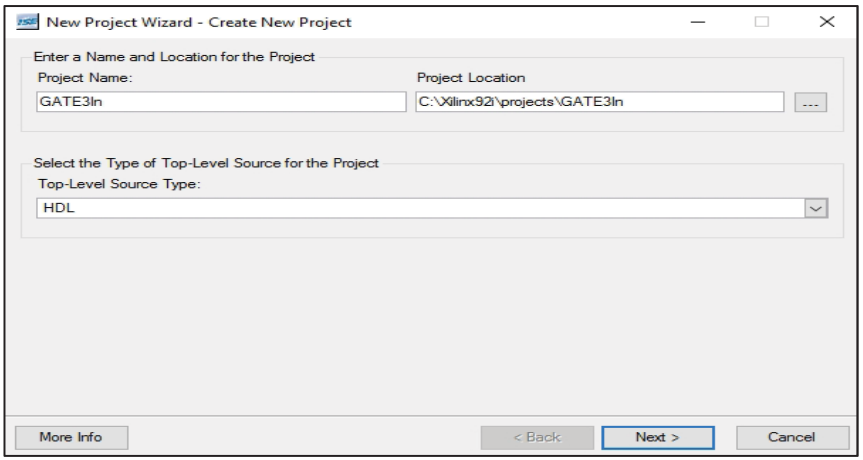


Figura 12.5. Creación de proyecto. Vista ampliada.

→ Next

Segundo paso:

Aplicar en el formulario la configuración seleccionada, de acuerdo con el modelo de FPGA a emplear.

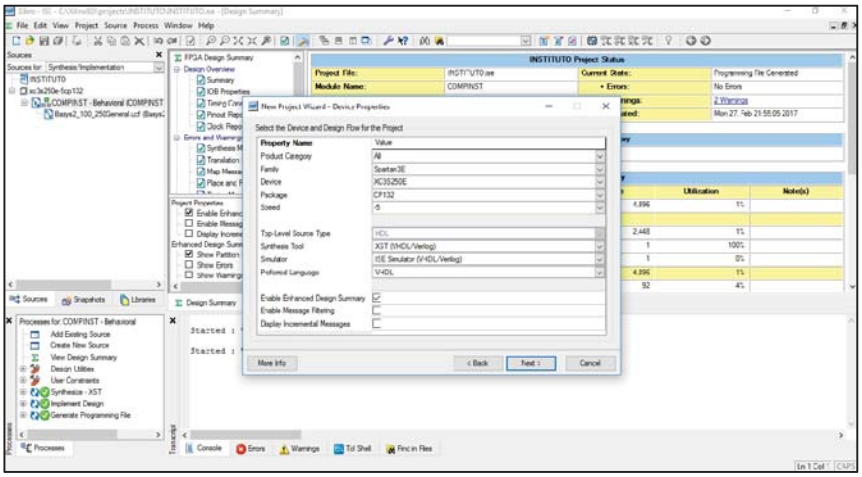


Figura 12.6. Características de la FPGA a emplear.

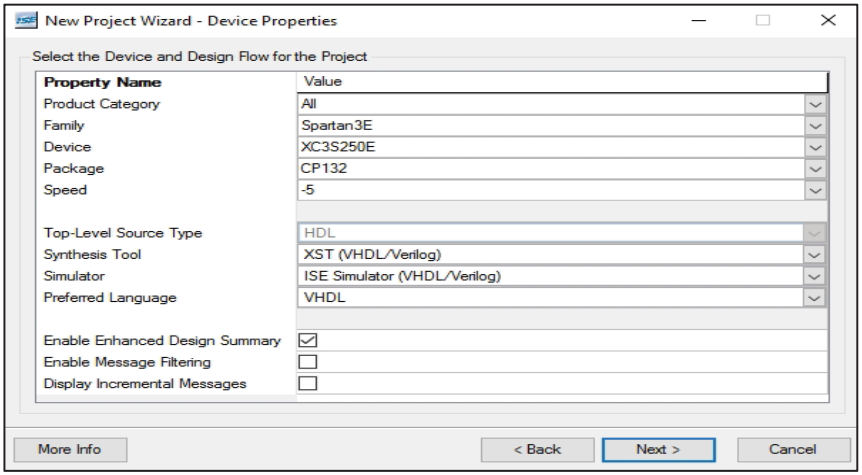


Figura 12.7. Características de la FPGA a emplear. Vista ampliada.

→ Next

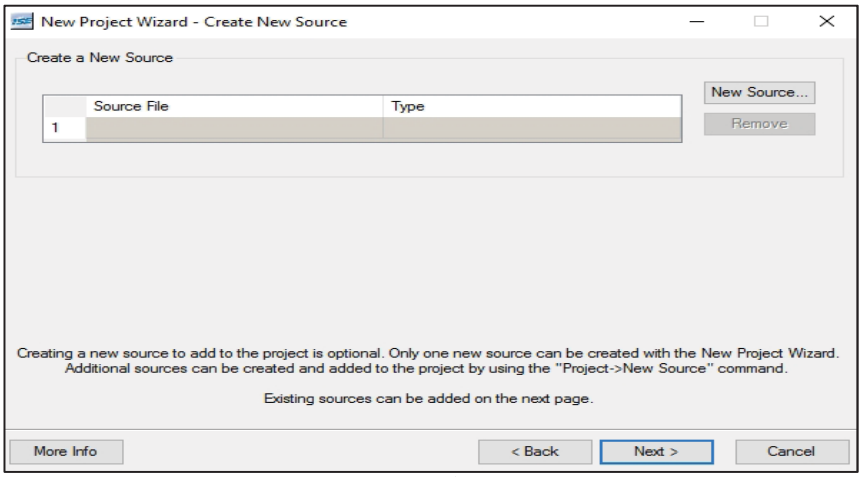


Figura 12.8. Creación de proyecto.

→ Next

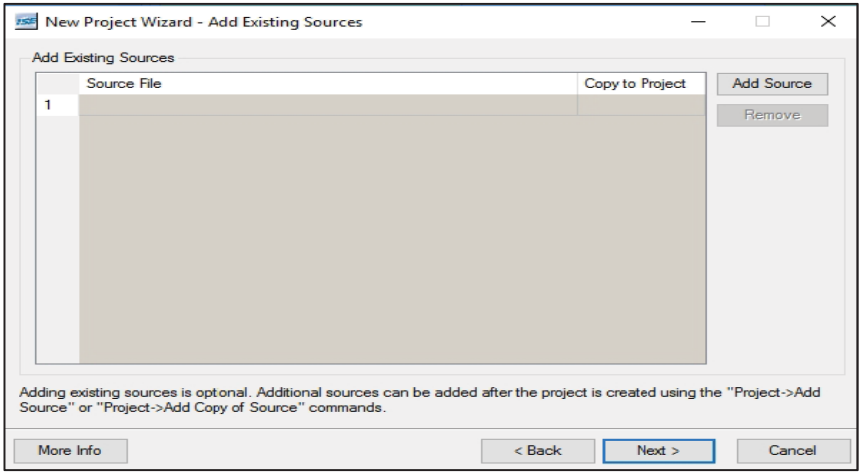


Figura 12.9. Creación de proyecto. Proceso.

→ Next

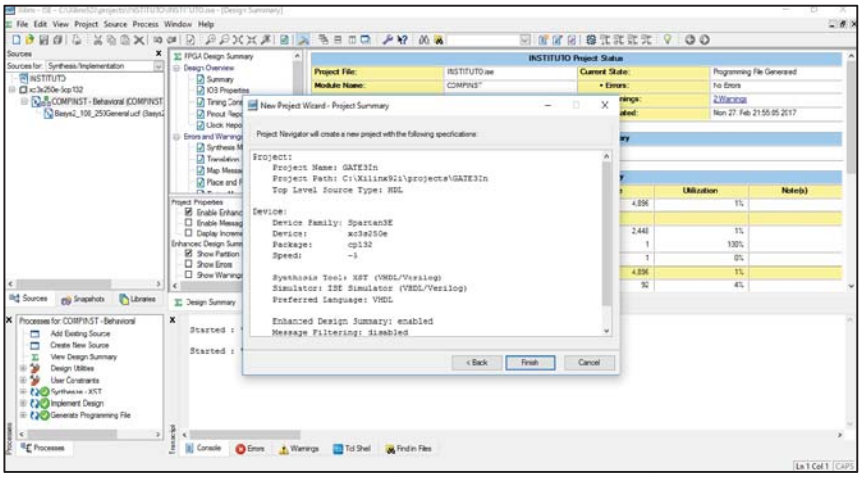


Figura 12.10. Creación de proyecto: finalización.

→ `Finish`

Seguidamente, del lado izquierdo de la pantalla, aparece el componente elegido. Al posicionarse en él, con el botón derecho se selecciona la opción `NEW SOURCE`.

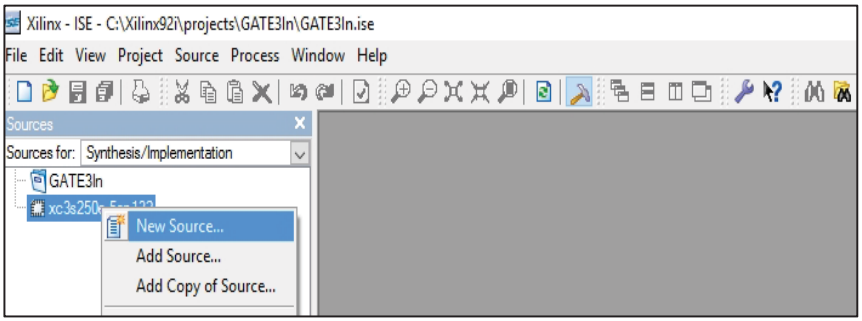


Figura 12.11. Creación de Módulo VHDL.

Se selecciona →

`VHDL MODULE`

Y solicita un →`FILE NAME:`

Ejemplo → `TresInGate`

→ Next

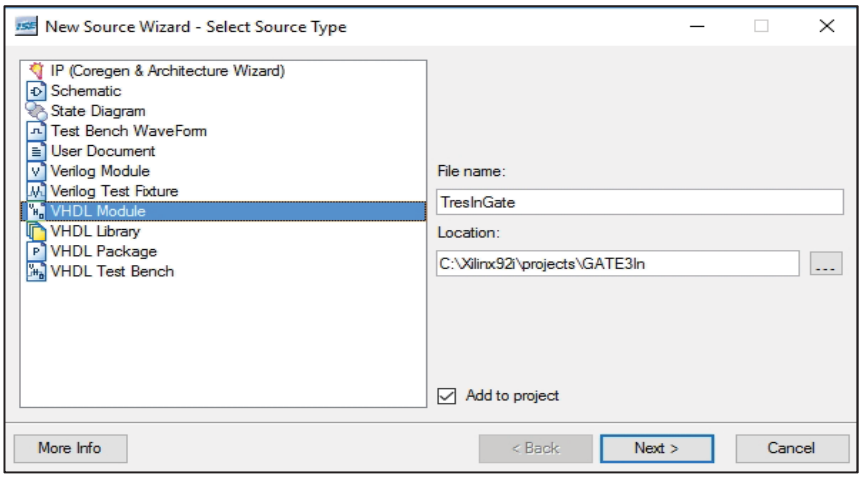


Figura 12.12. Se le asigna un nombre al módulo VHDL.

→ Next

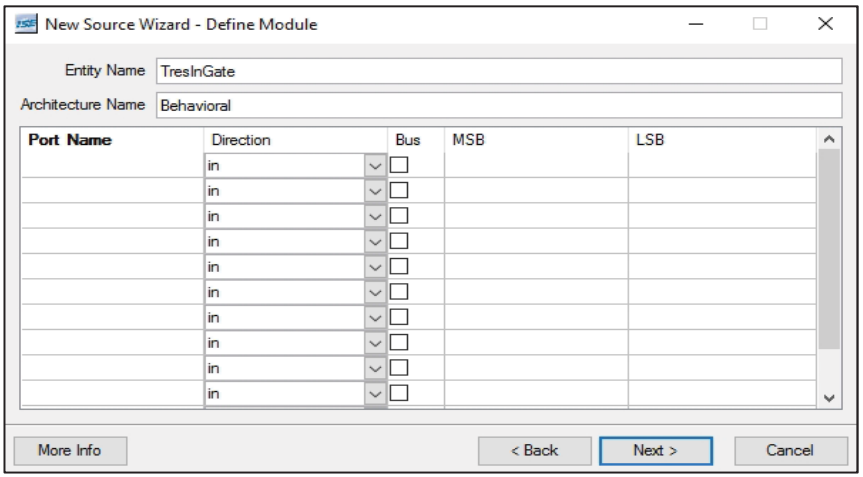


Figura 12.13. Se puede describir el componente a modelar o avanzar.

→ Next

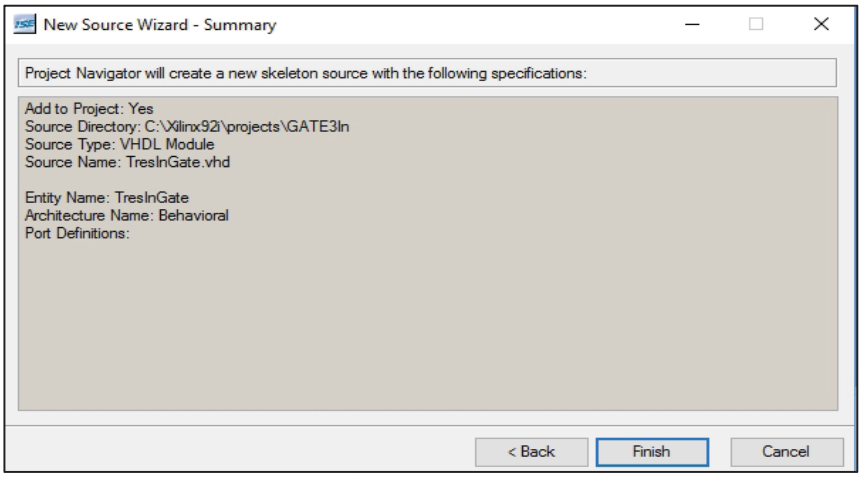


Figura 12.14. Generación de esqueleto fuente o componente.

→ `Finish`

Luego, aparece la siguiente pantalla; se cierra.

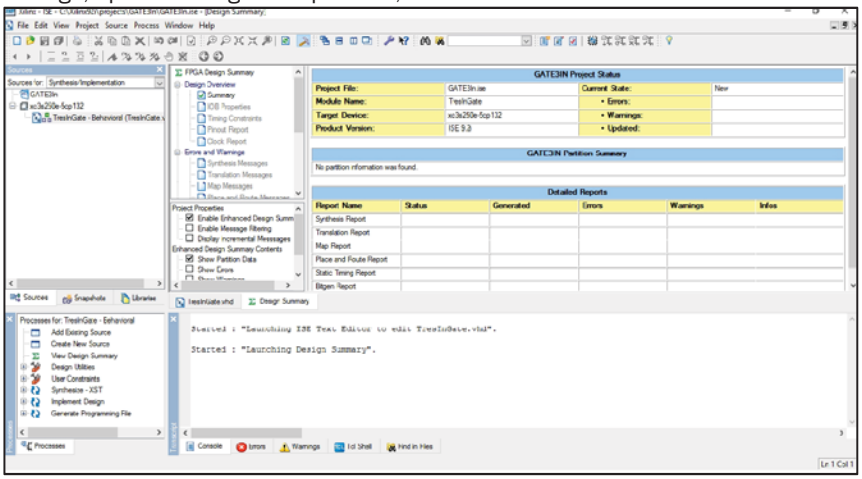


Figura 12.15. Resumen de diseño.

12.5.1 VHDL – LENGUAJE

Un lenguaje de descripción de hardware o HDL (*Hardware Description Language*), es similar a cualquier lenguaje de programación, con la diferencia de ser empleado para describir hardware (no programas de ejecución en la PC).

El lenguaje de programación empleado en este capítulo es VHDL. Su nombre proviene de dos acrónimos, y significa (*Very High Speed Integrated Circuit Hardware Description Language*) Lenguaje de Descripción de Hardware de Circuitos Integrados de Muy Alta Velocidad.

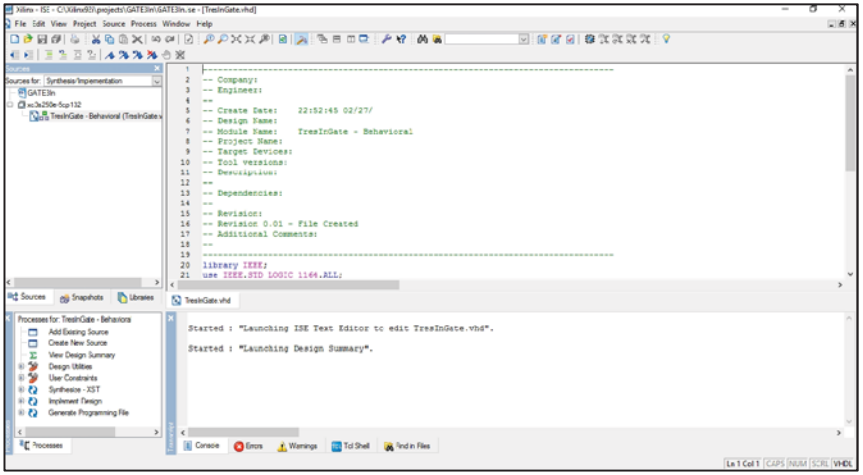


Figura 12.16. Generación de archivo `.vhd`.

Todo aquello que aparece en color verde son líneas de comentario. Se observa también que la extensión del nombre de archivo es: `.vdh`

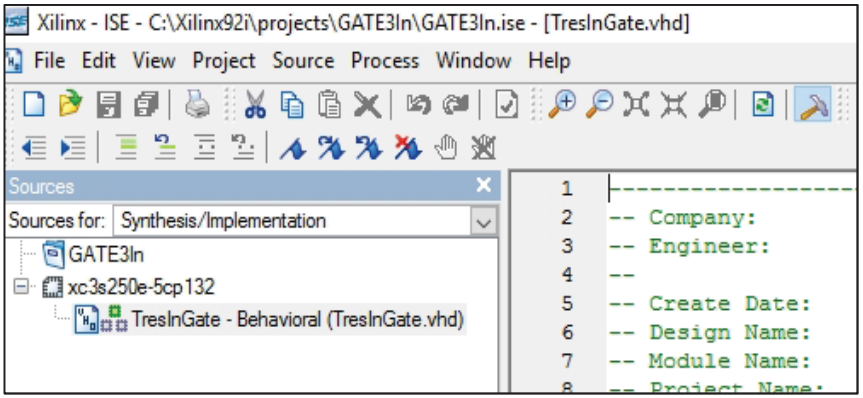


Figura 12.17. Componente `TresInGate`.

La palabra ***Behavioral***, que aparece marcada en gris junto al nombre y a la extensión del proyecto, se traduce como comportamiento.

Se borrarán las líneas verdes de comentarios para simplificar la lectura del código.

Se obtiene la siguiente estructura:

12.5.2 ESTRUCTURA DE UN PROGRAMA

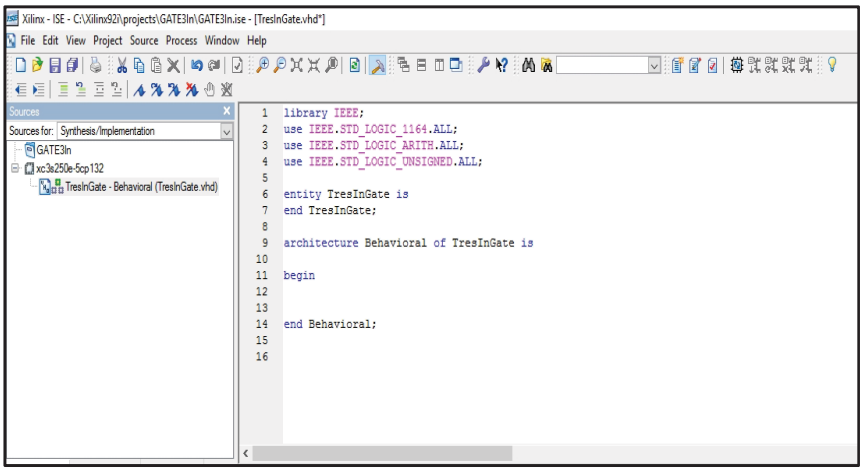


Figura 12.18. Imagen. Estructura del programa.

Líneas 1 a la 4: inclusión de librerías y declaraciones.

Líneas 6 y 7: representan la entidad del componente.

Líneas 9 a 14: representan la arquitectura - comportamiento del componente.

Seguidamente, se programa para definir la entidad del componente. Se observa la definición del empleo de tres pines del puerto del dispositivo como entrada, denominados “a”, “b”, “c” y un pin configurado como salida “s”.

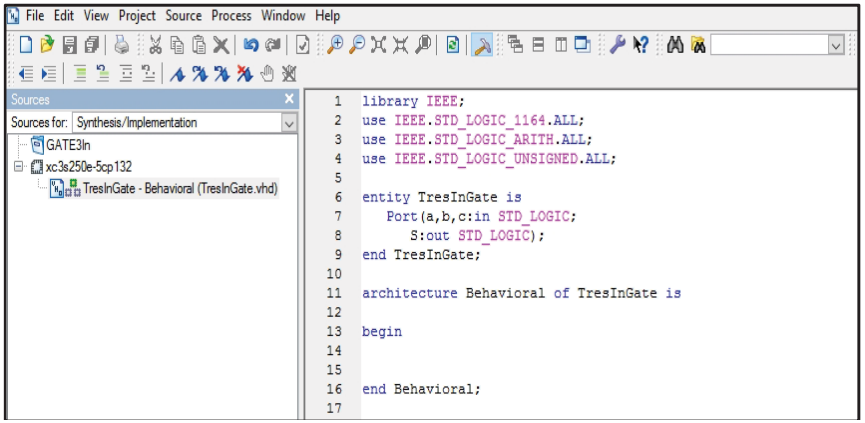


Figura 12.19. Definición de entidad.

12.5.3 COMPORTAMIENTO LÓGICO DEL COMPONENTE

Ahora, se debe programar el comportamiento lógico que desarrollará el componente desde su estructura. Hasta el momento, se ha definido el modelado de una compuerta o Gate de tres entradas. Sin embargo, no se ha definido su tipo, por lo tanto, es preciso diseñar una AND, para que sea más notoria, en la simulación, la necesidad de que sus entradas valgan 1, para obtener un estado alto en su salida.

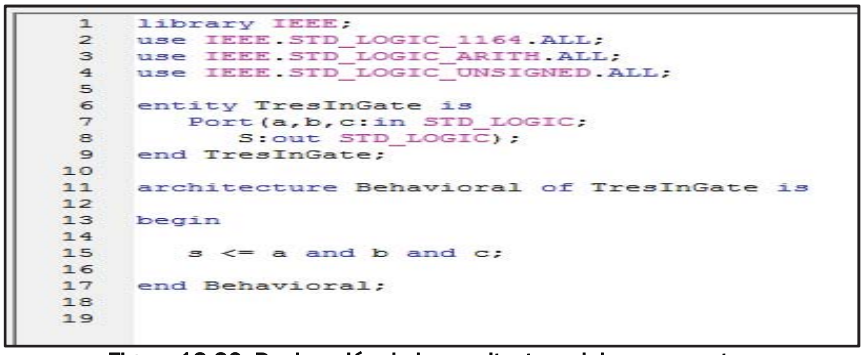


Figura 12.20. Declaración de la arquitectura del componente.

12.5.4 SINTAXIS Y SINTETIZACIÓN DEL PROGRAMA

El código ingresado:

```
s <= a and b and c;
```

se lee, corresponde a “s” la función and entre “a” y “b” y “c”.

Luego de realizado el código, se debe `sintetizar` el programa. Margen inferior izquierdo de la pantalla, con doble clic.

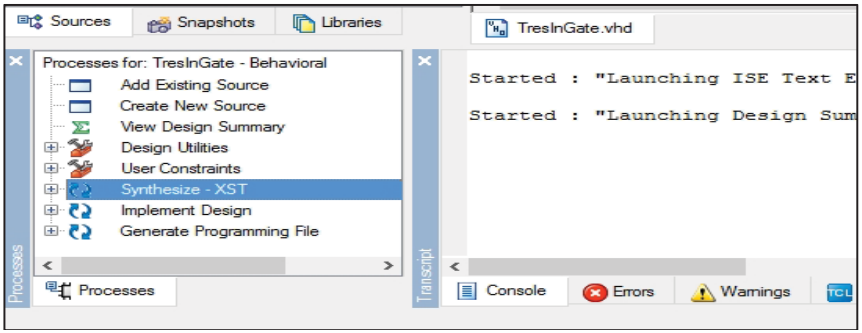


Figura 12.21. Sintetización del programa.

En caso de no existir error de sintaxis, se obtiene la siguiente leyenda:

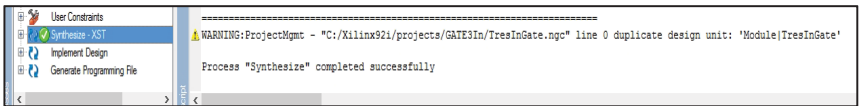


Figura 12.22. Proceso de sintetización exitoso.

Es preciso recordar que los *WARNING* son avisos. En caso de advertencia de un *WARNING*, en función de su descripción puede ser ignorado.

Otra aplicación interesante dentro de las opciones de `Synthesize-XST` es la posibilidad de seleccionar → `View RTL Schematic`, a fin de visualizar el esquemático del CI modelado.

12.5.5 VISTA DE ESQUEMÁTICO

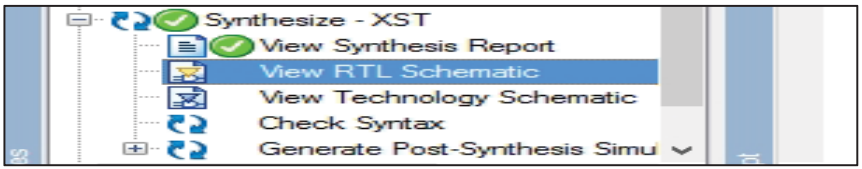


Figura 12.23. Opción RTL para visualizar esquemático.

Se obtiene

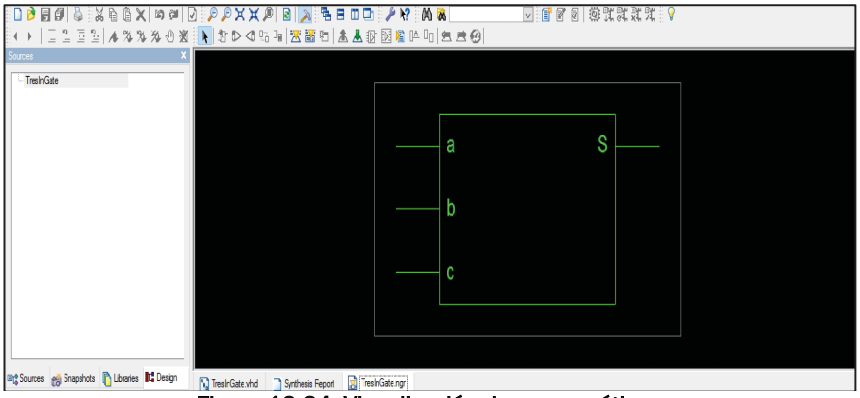


Figura 12.24. Visualización de esquemático.

Se observa el modelado de un dispositivo de tres entradas y una salida.

12.5.6 SIMULACIÓN DEL PROYECTO

Es posible también generar la Simulación del proyecto. Para ello, se debe volver a posicionar sobre el componente, y presionar nuevamente la opción `New Source`, pero esta vez seleccionar `Test Bench WaveForm`. Se le debe asignar un nombre al archivo, por ejemplo, `Comp3InSimulacion`.

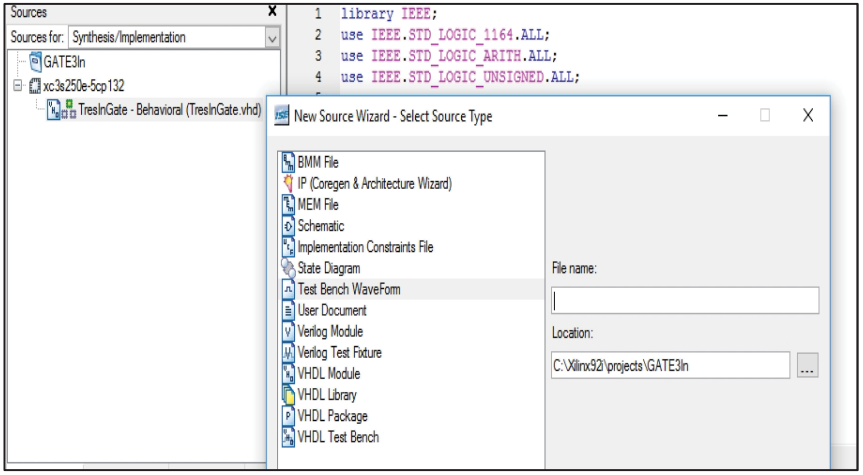


Figura 12.25. Generación de simulación.

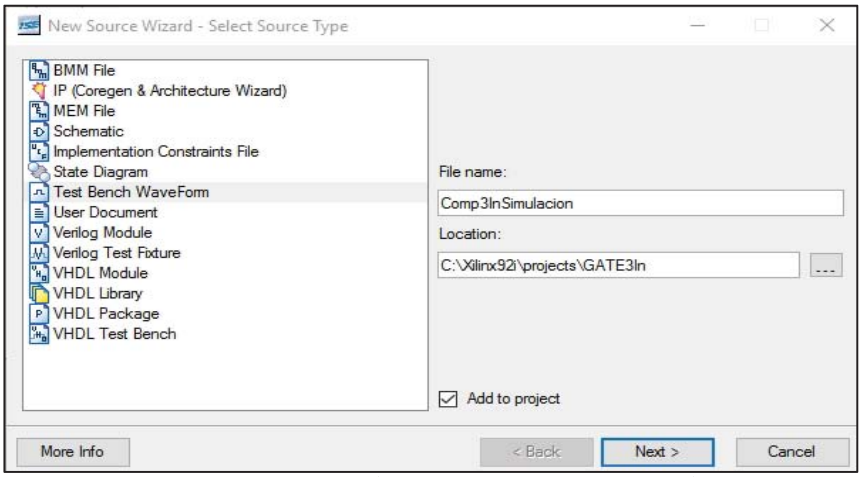


Figura 12.26. Asignación de nombre para la simulación.

Se observa también que la simulación generada queda almacenada en la carpeta del proyecto en curso.

→ Next

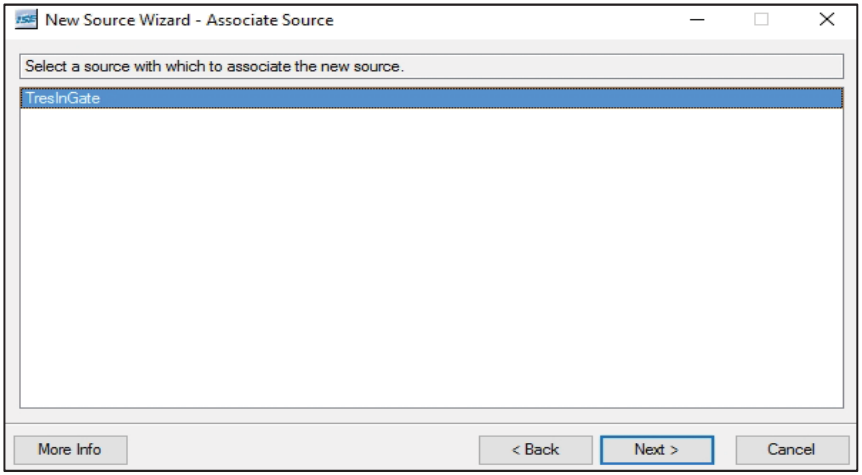


Figura 12.27. Se anexa la simulación al proyecto.

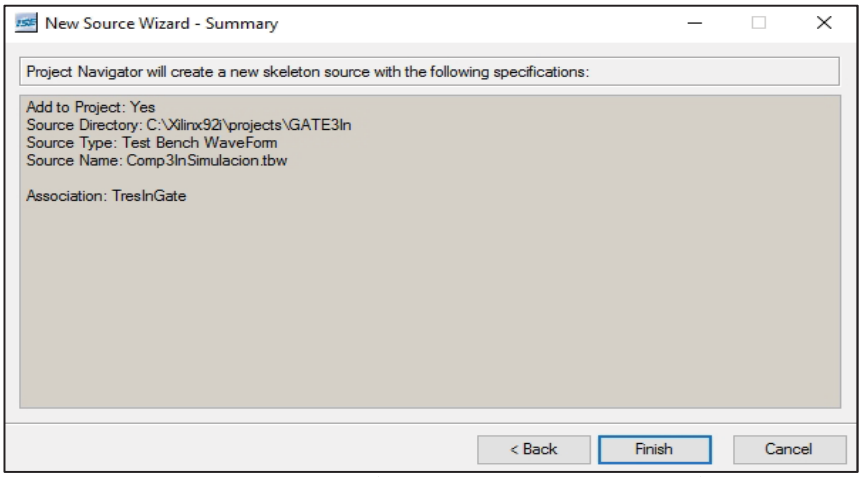


Figura 12.28. Generación del esqueleto de la simulación.

→ `Finish.`

Luego, se genera el siguiente formulario:

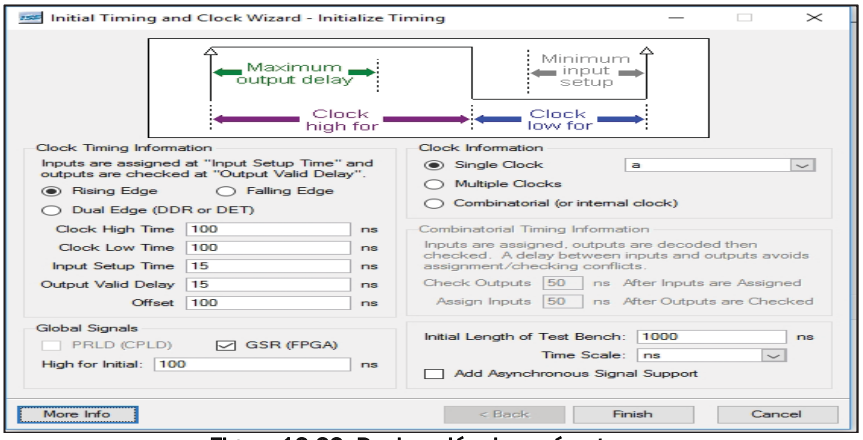


Figura 12.29. Declaración de parámetros.

Aquí se deben marcar las siguientes opciones:

Y setear un valor de **2000 ns** para visualizar mejor la simulación.

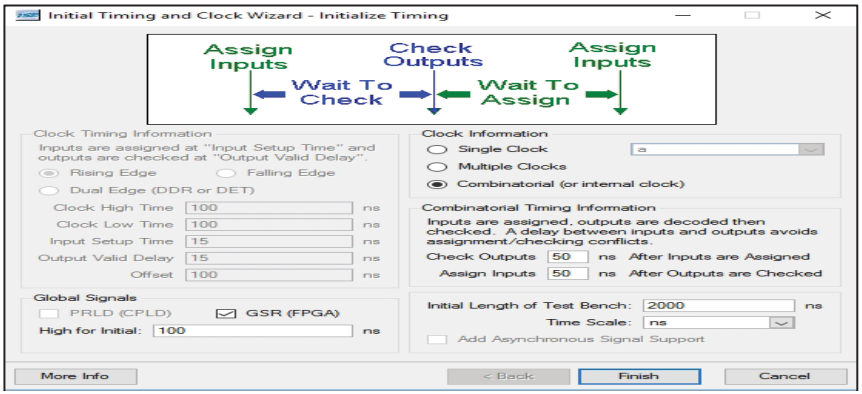


Figura 12.30. Fin de declaraciones.

→ `Finish.`

Se genera, la imagen

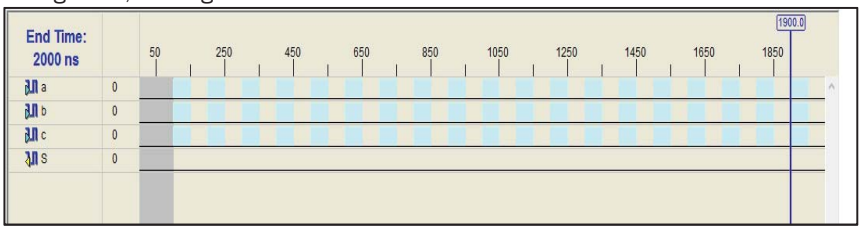


Figura 12.31. Diagrama temporal.

Se observan los diagramas temporales trabajados en el Capítulo 7: una fila para representar a cada una de las tres entradas, y la correspondiente a la salida **s**. El tiempo en ns se visualiza en la parte superior.

En esta instancia, se debe generar una forma de onda, para obtener posteriormente los valores de simulación de la tabla de verdad de la compuerta AND, a medida que se corre la marca.

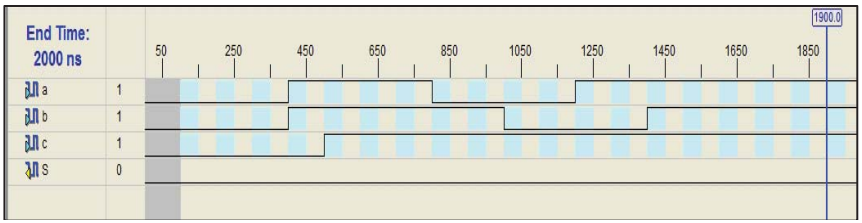


Figura 12.32. Generación de una forma de onda para la simulación.

Los valores o estados (alto/bajo), se generan haciendo un clic en cada punto que se pretende poner en alto; del modo contrario, baja al cliquear nuevamente. Se salva el proyecto y se cierra esta ventana.

Luego, para trabajar sobre el proyecto, se vuelve a la solapa superior, y en `Sources for`, se selecciona la opción:

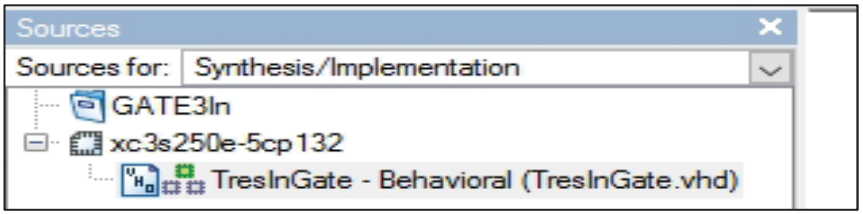


Figura 12.33. Acceso a la simulación.

Behavioral Simulation

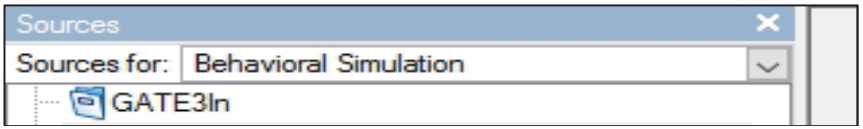


Figura 12.34. Cambiamos a la opción `Behavioral Simulation`.

Se observa en la imagen siguiente, que al hacer clic sobre el componente aparece el archivo correspondiente a la simulación, con la extensión `.tbw`.

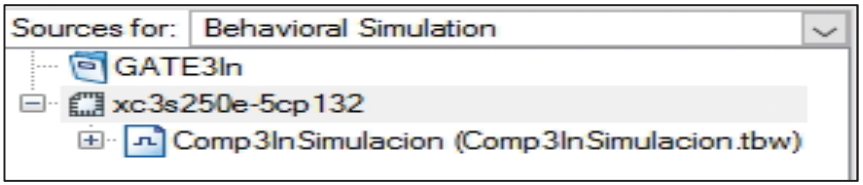


Figura 12.35. Imagen. Se visualiza el archivo de simulación.

Luego, se debe cliquear sobre el archivo de simulación:

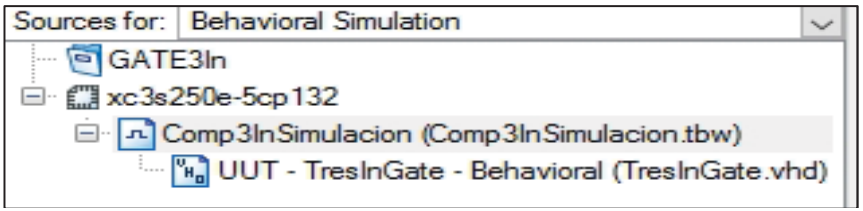


Figura 12.36. Cliqueamos sobre el archivo de simulación.

Y en el cuadro inferior izquierdo en `Processes`:

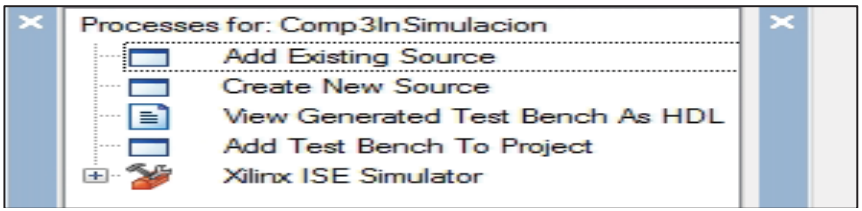
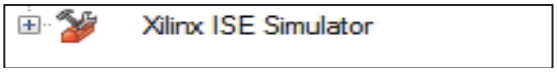


Figura 11.37. Cuadro inferior opción `Processes`.

Expandir la opción



Se obtiene

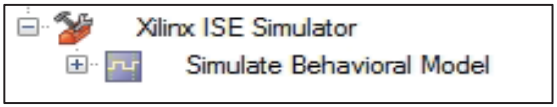


Figura 12.38. `Simulate Behavioral Model` opción.

Se hace doble clic y se obtiene la simulación de la compuerta, con los valores de simulación seleccionados anteriormente.

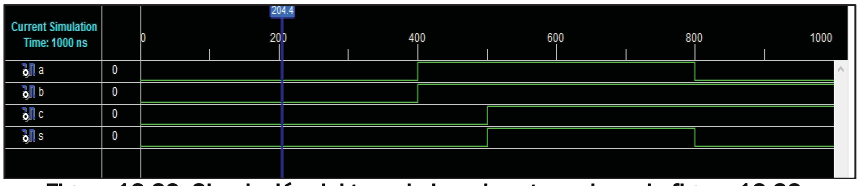


Figura 12.39. Simulación del tren de impulsos trazado en la figura 12.32.

Se observa que, puesta la marca en 200 ms, los tres valores de entrada tienen un valor de 0 lógico, y se obtiene un estado bajo a la salida.

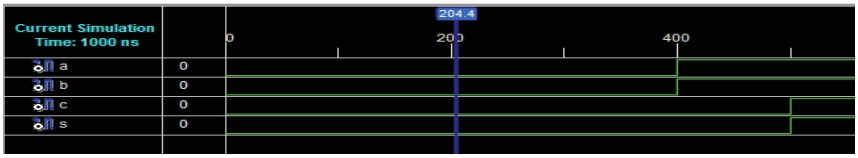


Figura 12.40. Simulación. Tabla de verdad. Compuerta AND.

Cuando todos los valores de entrada están a nivel lógico 1, la salida cambia al estado lógico alto.

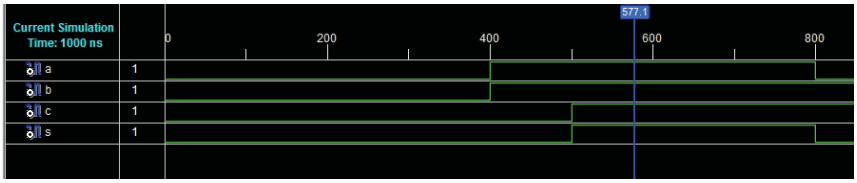


Figura 12.41. Verificación de la tabla de verdad.

De esta forma se obtiene la verificación del funcionamiento correcto del dispositivo modelado.

Una vez realizados el modelado y la prueba de funcionamiento del dispositivo, se debe generar el archivo para descargar a la ROM de la placa.

12.5.7 GENERACIÓN DE ARCHIVO EJECUTABLE

La extensión del archivo ejecutable es `.bit` por lo que para el ejemplo, se obtendrá `Gate3In.bit`.

Previamente, se debe contar también con un archivo llamado `UCF` (*use construyed file*), que se descarga del website de Digilent.

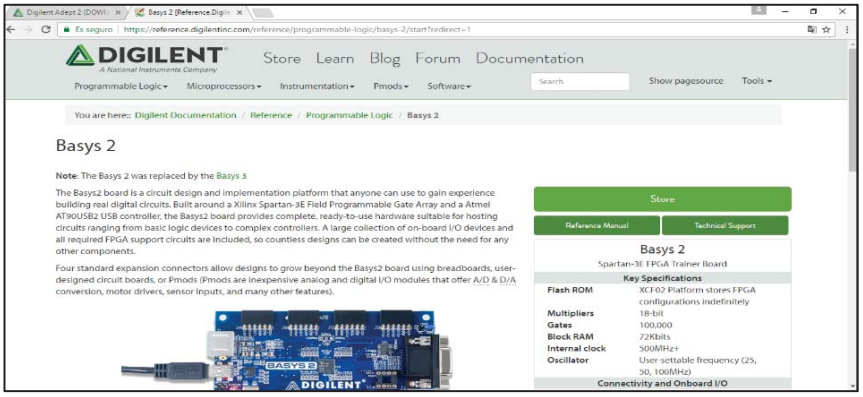


Figura 12.42. Digilent website.

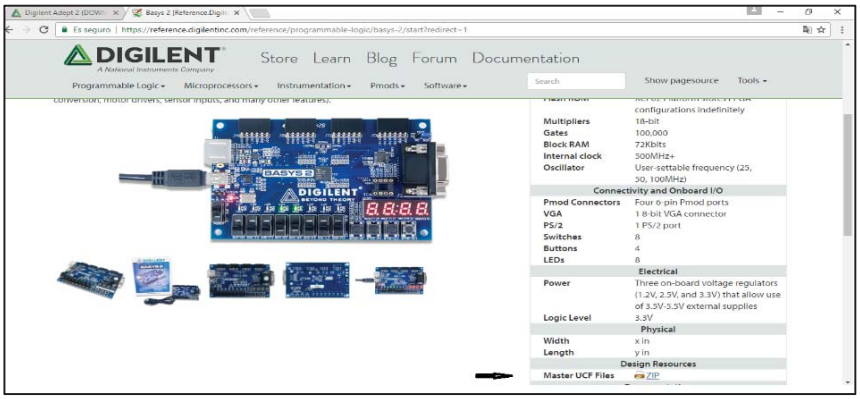
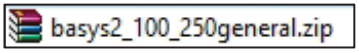


Figura 12.43. Descarga de archivo UCF.

Una vez descargado el ZIP Master UCF Files:



Se descomprime el archivo para acceder a él:

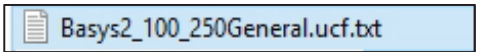
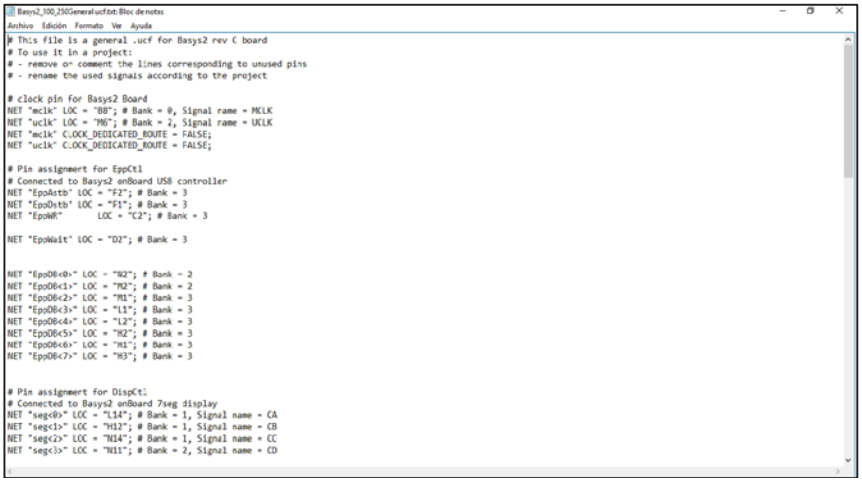


Figura 12.44. Archivo UCF.

El archivo contiene la descripción del Hardware de la placa, así como los archivos `include` para los microcontroladores.



```
# This file is a general .ucf for Basys2 rev C board
# To use it in a project:
# - remove or comment the lines corresponding to unused pins
# - rename the used signals according to the project

# clock pin for Basys2 Board
NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
NET "uclk" LOC = "M6"; # Bank = 2, Signal name = UCLK
NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;
NET "uclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for EppCtl
# Connected to Basys2 onBoard USB controller
NET "EppAstb" LOC = "F2"; # Bank = 3
NET "EppDstb" LOC = "F1"; # Bank = 3
NET "EppWR"       LOC = "C2"; # Bank = 3

NET "EppWait" LOC = "D2"; # Bank = 3


NET "EppDB<0>" LOC = "N2"; # Bank = 2
NET "EppDB<1>" LOC = "M2"; # Bank = 2
NET "EppDB<2>" LOC = "M1"; # Bank = 3
NET "EppDB<3>" LOC = "L1"; # Bank = 3
NET "EppDB<4>" LOC = "L2"; # Bank = 3
NET "EppDB<5>" LOC = "H2"; # Bank = 3
NET "EppDB<6>" LOC = "H1"; # Bank = 3
NET "EppDB<7>" LOC = "H3"; # Bank = 3


# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
NET "seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
NET "seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
NET "seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
```

Figura 12.45. Archivo UCF Contenido.

Otra opción para trabajar con el archivo UCF es modificarlo en función de los pines necesarios para el proyecto. Para esto, se necesita contar también con el `PDF` descriptivo de la placa.

En este caso, la placa de entrenamiento a emplear es la Digilent Basys 2.

Figura 12.46. Placa Digilent Basys 2.

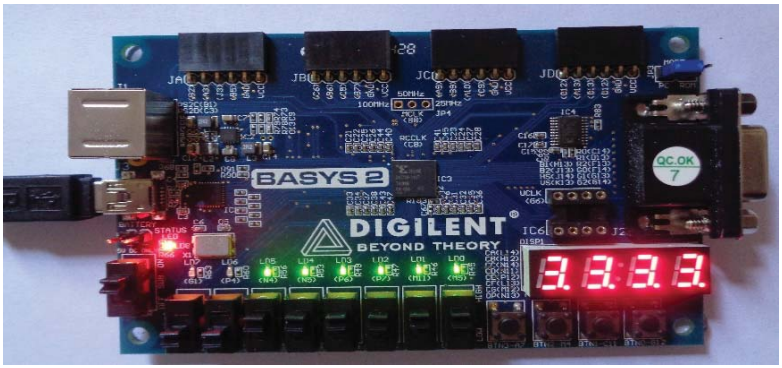


Figura 11.47. Placa Digilent Basys 2 – Test.

La documentación necesaria para trabajar se puede descargar también del website de Digilent:

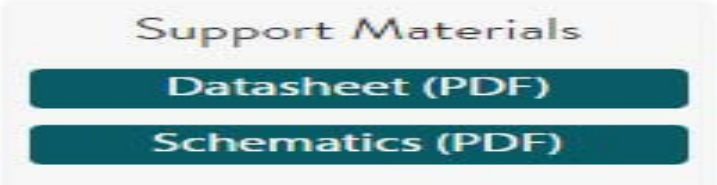


Figura 12.48. Digilent website. *Data sheet* disponible.

Si se observa con atención la imagen de la placa, se reconoce, en la parte inferior, la disposición de tres pulsadores y un diodo LED en SMD, que se pueden emplear para simular el comportamiento de la compuerta AND de tres entradas.

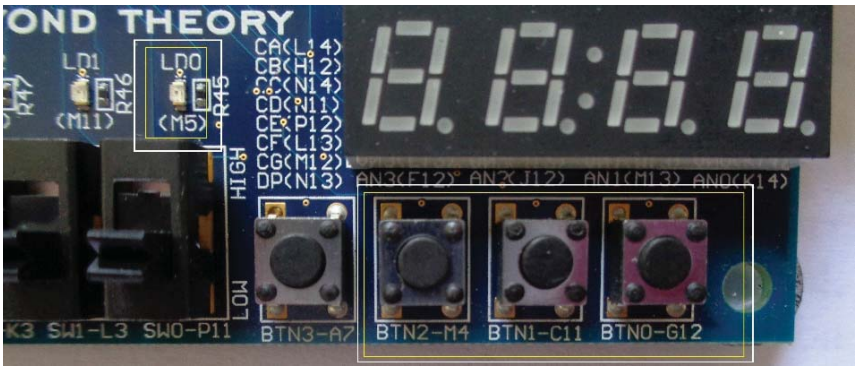


Figura 12.49. Pulsadores y LED SMD empleados para la simulación.

La denominación para los pulsadores es G12, C11, M4 y para el diodo LED M5.

Se pueden trasladar estos datos al archivo `UCF`, y, de esta manera, eliminar la información que no se va a emplear para este proyecto.

Figura 12.50. Archivo `UCF`, declaración de hardware a emplear.

Es importante observar en la imagen anterior que se deben mantener las iniciales o nombres empleados cuando en el programa se define la identidad (a, b, c, s). También es preciso recordar que el `UCF` generado debe guardarse en el disco C, dentro de la carpeta de proyecto en curso.

Ahora si están dadas las condiciones para generar el archivo `Gate3In.bit` para descargarlo luego en la placa y comprobar su funcionamiento. Para ello se debe volver al proyecto, y seleccionar en `Sources for` la opción `Synthesis/Implementation`.

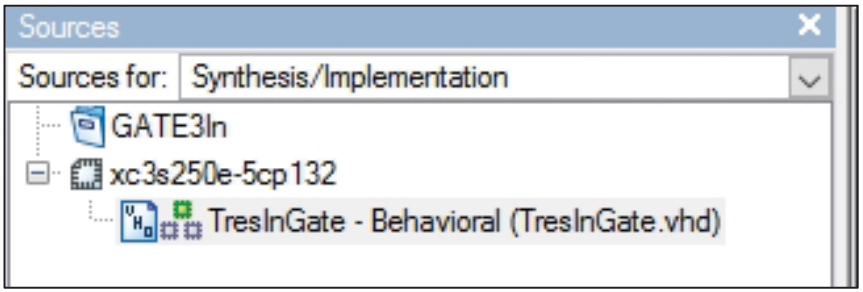


Figura 12.51. Generación de archivo a descargar a la placa.

Es preciso volver sobre el componente, y con el botón derecho, seleccionar la opción `Add Source`...

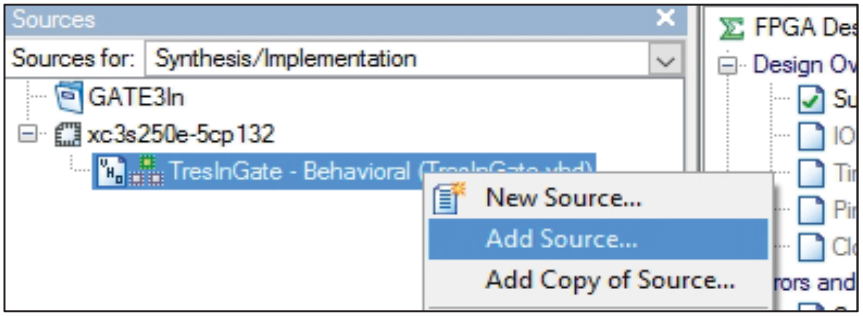


Figura 12.52. Adherimos el UCF al proyecto.

Y en la carpeta del proyecto en curso, se debe seleccionar el archivo `Basys2_100_250General.ucf`.

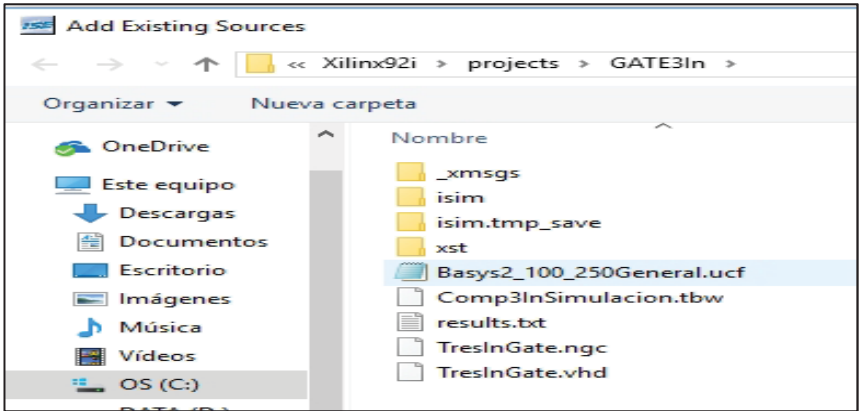


Figura 12.53. Archivo UCF en la carpeta del proyecto.

Al hacer `doble clic` aparece el formulario:

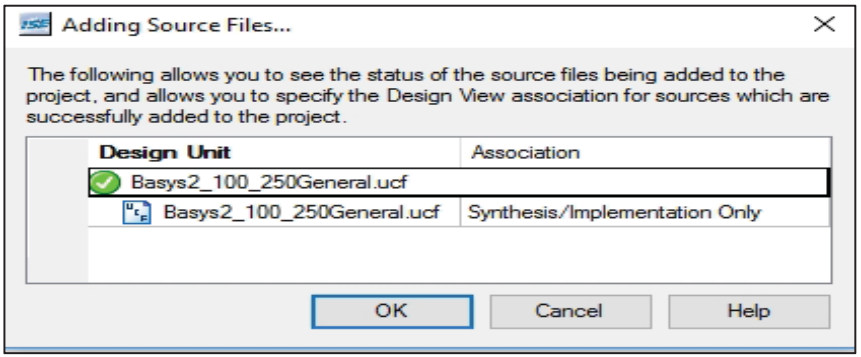


Figura 12.54. Generación de archivo `.bit`

Selección→ `Ok`,

Y en el margen inferior izquierdo de la pantalla, se selecciona la opción

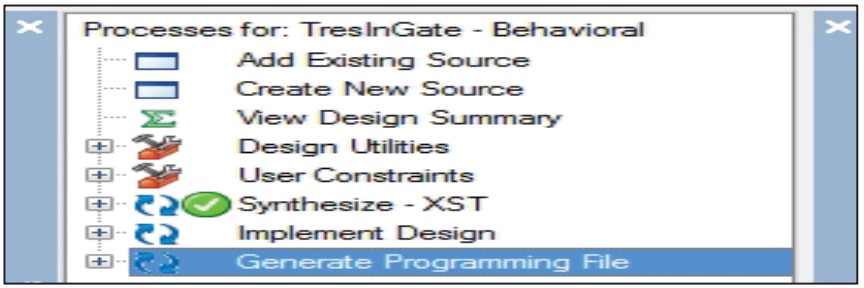


Figura 12.55. Generación de archivo.

Se hace `doble clic`, y si todo es correcto, aparece en pantalla la siguiente imagen:

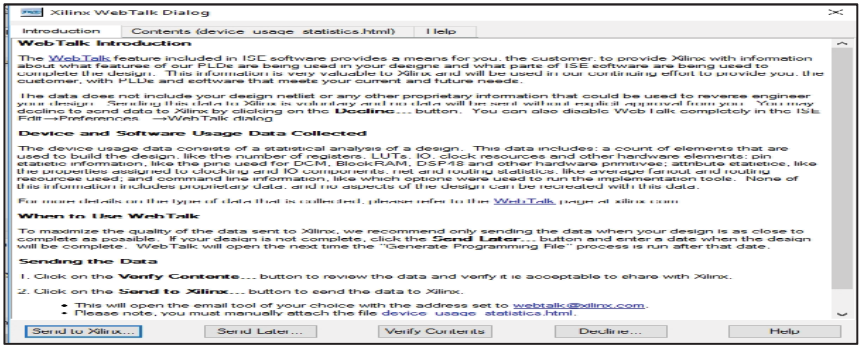


Figura 12.56. Archivo generado.

Se la cierra y se observa en la parte inferior de la pantalla, en el cuadro de diálogos, la leyenda `Generación de archivo de programa generado satisfactoriamente`.

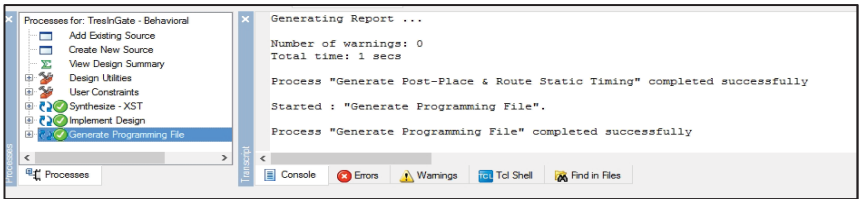


Figura 12.57. Leyenda de comprobación.

Ahora en la carpeta del proyecto, se encuentra el archivo con la extensión `.bit`.

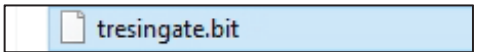


Figura 12.58. Archivo `tresingate` en carpeta de proyecto.

Se lo puede dejar allí, o bien, llevarlo al escritorio.

12.5.7 CONEXIÓN DE LA PLACA BASYS 2 A LA PC

Para poder interfacear nuestra placa con la PC, se debe, como último paso, descargar el software Adept del website de Digilent.

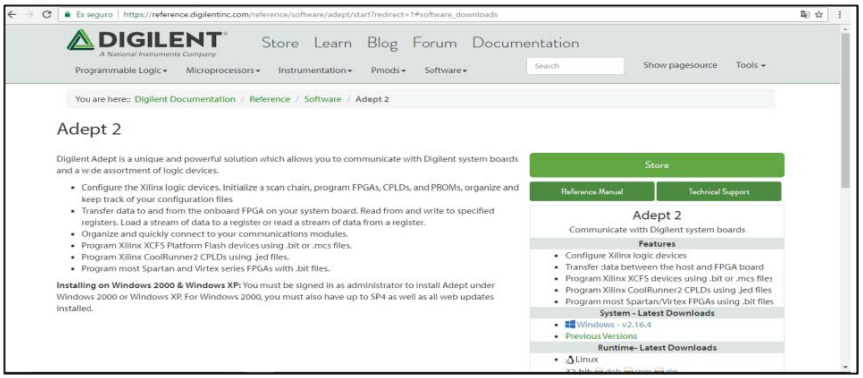


Figura 12.59. Digilent website.

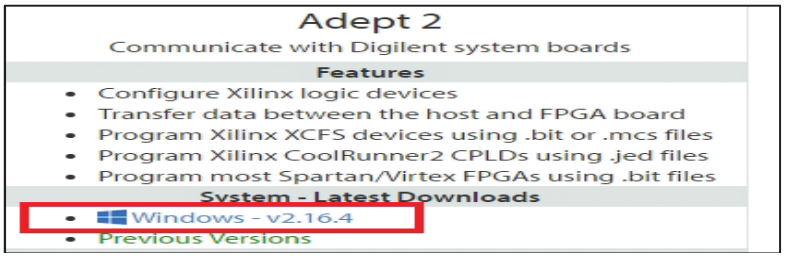


Figura 12.60. Descarga de archivo `Adept 2`.

Una vez descargado y ejecutado el software, aparece el ícono del programa en el escritorio. Se lo abre y aparece la siguiente ventana:

Figura 12.61. Icono Adept.

Figura 12.62. Opciones de interface.

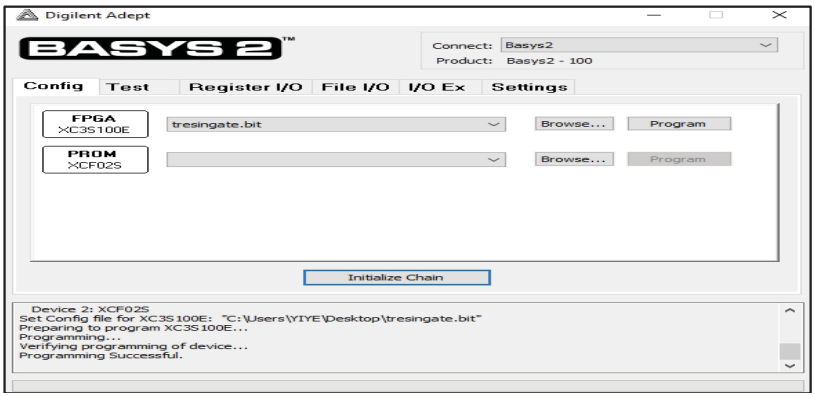


Figura 12.63. Descarga de archivo a la placa.

Se observa en la imagen anterior que a la derecha del botón `FPGA`, está cargado el archivo `tresIngate.bit`. Luego, simplemente se selecciona `programar`.

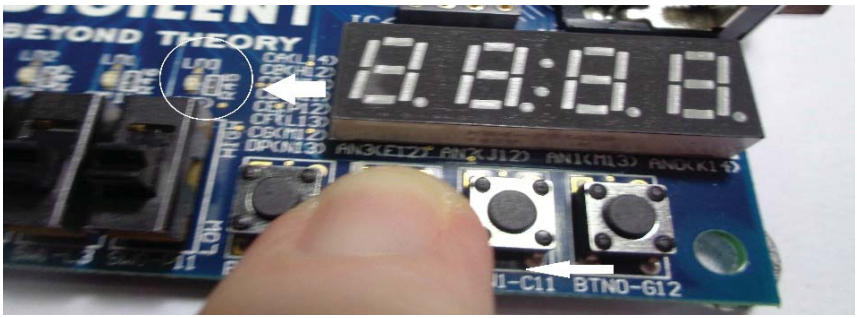


Figura 12.64. Simulación. Tabla de verdad para la compuerta AND modelada.

En la imagen se muestra de acuerdo con los pulsadores configurados para el diseño. Que en cumplimiento de la tabla de verdad de la Compuerta AND, todas las entradas deben estar a nivel lógico alto para obtener un nivel lógico alto a la salida, tal cual se observa en la Figura 12.65.

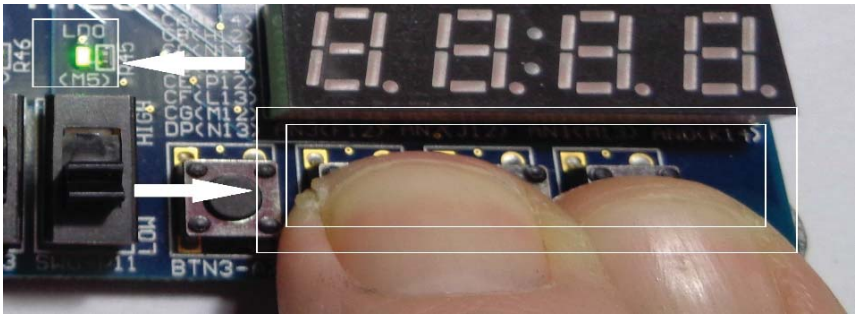


Figura 12.65. Con las tres entradas a nivel alto LD0 enciende.