



**UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL**

CARRERA DE INGENIERÍA ELECTRÓNICA

**Trabajo de titulación previo a la obtención del título de:
INGENIERO ELECTRÓNICO**

Título:

**“DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL
CONTROLADO POR UN PLC DELTA PARA APLICACIONES
DIDÁCTICAS DE AUTOMATIZACIÓN Y CONTROL”.**

Autor:

ÁNGEL FABIÁN MANZANO GALARZA

Tutor:

ING. BYRON LIMA CEDILLO, MSc.

OCTUBRE 2018

GUAYAQUIL –ECUADOR

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA

Yo, ÁNGEL FABIÁN MANZANO GALARZA autorizo a la **Universidad Politécnica Salesiana** la publicación total o parcial de este trabajo de titulación y su reproducción sin fines de lucro.

Además, se declara que los conceptos y análisis desarrollados y conclusiones del presente trabajo son de exclusiva responsabilidad del autor.

Ángel Fabián Manzano Galarza

C.I: 0202045951

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR

Yo, **ÁNGEL FABIÁN MANZANO GALARZA**, con documento de identificación N° 0202045951, manifestó mi voluntad y cedo a la **UNIVERSIDAD POLITÉCNICA SALESIANA** la titularidad sobre los derechos patrimoniales en virtud de que soy autor del trabajo de grado titulado: **“DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL CONTROLADO POR UN PLC DELTA PARA APLICACIONES DIDÁCTICAS DE AUTOMATIZACIÓN Y CONTROL”** mismo que ha sido desarrollado para optar por el título de **INGENIERO ELECTRÓNICO**, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos antes cedidos.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscrito este documento en el momento que se realiza la entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Ángel Fabián Manzano Galarza

C.I: 0202045951

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **ING. BYRON XAVIER LIMA CEDILLO, MSc.** Director del proyecto de titulación denominado: **“DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL CONTROLADO POR UN PLC DELTA PARA APLICACIONES DIDÁCTICAS DE AUTOMATIZACIÓN Y CONTROL”** realizado por el estudiante **ÁNGEL FABIÁN MANZANO GALARZA**, certifico que ha sido orientado y revisado durante su desarrollo, por cuanto se aprueba la presentación del mismo ante las autoridades pertinentes.

Guayaquil, octubre del 2018

Ing. Byron Xavier Lima Cedillo, MSc.

DEDICATORIA

Dedico este trabajo de titulación a mi familia por apoyarme incondicionalmente en todo sentido y en todo momento.

A Dios por colmarme de salud y fuerzas suficientes para superar los retos presentado a lo largo de mi periodo estudiantil.

A todos y cada uno de los docentes por su tiempo y su paciencia dedicada en mi vida académica.

Ángel Fabián Manzano Galarza

AGRADECIMIENTO

Mi gratitud, principalmente está dirigida a todos y cada uno de mis familiares por apoyarme y motivarme incondicionalmente, por demostrándome cada día lo importante que es la vida, y mucho más a mis padres por enseñarme la fortaleza de carácter.

A mi hermana Pilar por ser mi apoyo fundamental e incondicional, a quien estimo y valoró mucho.

Igualmente, a Dios por haberme colmado de salud, fuerza e inteligencia permitiéndome llegar al final de mis estudios.

A todos y cada uno de los docentes, al Tutor y amigo Ing. Byron con quien pude compartir grandes experiencias en el Club de Robótica de esta universidad, lugar donde nació mi frase de vida “¿y qué hacemos nos sentamos a llorar?”.

A todos mis allegados que de una u otra forma han colaborado con un granito de arena, a quienes agradezco de forma sincera su valiosa colaboración.

Ángel Fabián Manzano Galarza

RESUMEN

AÑO	ALUMNO	DIRECTOR DE PROYECTO TÉCNICO	TEMA DE PROYECTO TÉCNICO
2018	ANGEL FABIÁN MANZANO GALARZA	ING. BYRON XAVIER LIMA CEDILLO, MSc.	“DISEÑO E IMPLEMENTACIÓN DE UN ROBOT MÓVIL CONTROLADO POR UN PLC DELTA PARA APLICACIONES DIDÁCTICAS DE AUTOMATIZACIÓN Y CONTROL”.

El presente proyecto está orientado al desarrollo e implementación de un robot móvil prototipo con enlace inalámbrico a una maleta didáctica, para implementación de prácticas de automatización y control.

El robot móvil prototipo cuenta con un módulo Arduino, un convertidor de nivel TTL a RS-232, un módulo inalámbrico RS-232/Bluetooth, mandos locales, un sistema diferencial de motores DC y sensores que se definen según su funcionalidad como:

- Sensores de obstáculo
- Sensores de piso
- Sensor para detección de carga

Una maleta didáctica cuenta con un PLC DELTA, un módulo inalámbrico RS-232/Bluetooth, una interfaz Hombre-Máquina, mandos e indicadores locales.

Las prácticas propuestas en este proyecto guían al estudiante a desarrollar e implementar temas específicos como transmisión de datos con enlaces industriales inalámbricos, en conjunto con sensores y actuadores, se familiarizan con el uso de un control PID, y demás puntos que se validan en el capítulo 4.

Palabras claves: PLC, DELTA, Arduino, RS232, Bluetooth, DC, PID.

ABSTRACT

YEAR	STUDENT	DIRECTOR OF TECHNICAL PROJECT	TECHNICAL PROJECT THEME
2018	ANGEL FABIÁN MANZANO GALARZA	ING. BYRON XAVIER LIMA CEDILLO MSc.	"DESIGN AND IMPLEMENTATION OF A MOBILE ROBOT CONTROLLED BY A DELTA PLC FOR DIDACTIC AUTOMATION AND CONTROL APPLICATIONS"

The present project is oriented to the development and implementation of a prototype mobile robot with wireless link to a didactic case, for implementation of automation and control practices.

The prototype mobile robot has an Arduino microcontroller, a TTL converter with RS-232 interface, a Bluetooth wireless module with RS232 interface, local controls and indicators, a DC motor differential system and a variety of sensors that are defined according to their functionality as:

- Obstacle sensors
- Floor sensors
- Load detection sensor

A didactic case with a DELTA programmable logic controller installed, a Bluetooth wireless module with RS-232 interface, a Man-Machine interface, emergency stop and local controls and indicators. The practices proposed in this project guide the student to develop analyze and implement specific topics as the transmission of data through wireless industrial links between controllers of different application or development, encourage the joint work of sensors and actuators, become familiar with the use of a PID control, and other points that are validated in each of the practices detailed in chapter 4.

Keywords: DELTA, Arduino, RS232, Bluetooth, DC, PID.

ÍNDICE GENERAL

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA	II
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR.....	III
CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN.....	IV
DEDICATORIA	V
AGRADECIMIENTO	VI
RESUMEN.....	VII
ABSTRACT	VIII
ÍNDICE GENERAL	IX
ÍNDICE DE FIGURAS.....	XI
ÍNDICE DE TABLAS.....	XIV
INTRODUCCIÓN.....	1
1. EL PROBLEMA	2
1.1. Antecedentes.....	2
1.2. Importancia y Alcances.....	2
1.3. Beneficiarios.....	2
1.4. Delimitación.....	3
1.4.1. Temporal	3
1.4.2. Espacial.....	3
1.4.3. Académica.....	3
1.5. Objetivos	3
1.5.1. Objetivo general	3
1.5.2. Objetivos específicos.....	3
2. FUNDAMENTOS TEÓRICOS.....	4
2.1. Controlador programable DELTA DVP SX2.....	4
2.2. Panel Táctil HMI DELTA serie B.....	5
2.3. Conversor TTL a RS-232.....	5
2.4. Módulo Firefly	6
2.4.1. Características de Módulo FireFly	6
2.4.2. Conexiones físicas.....	7
2.5. Interruptor automático 6KA 1POL C2	7
2.6. Arduino MEGA 2560 R3	8
2.7. Fuente LOGO! Power 110VAC/24VDC 2.5A	9
2.8. Motorreductor 70:1	9
2.9. Matriz de sensores QTR-8A	10
2.10. Controlador de motor dual 12A.....	11

2.11. Sensor de distancia Analógico.....	12
3. MARCO METODOLÓGICO	12
3.1. Diseño de compartimentos para maleta didáctica.....	13
3.1.1. Autómata Programable Delta DVP-SX2.....	14
3.1.2. HMI Delta	14
3.1.3. Módulo Firefly	15
3.2. Robot móvil	15
3.2.1. Diseño de la Tarjeta robot.....	17
3.2.2. Driver Motores	18
3.3. Diagrama de bloques.....	18
3.4. Conexiones internas para maleta didáctica	19
3.5. Topología de comunicación	20
4. PRÁCTICAS DE LABORATORIO	22
4.1. Práctica #1	22
4.2. Práctica #2	39
4.3. Práctica #3	53
4.4. Práctica #4	59
4.5. Práctica #5	71
4.6. Práctica #6	84
4.7. Práctica #7	95
4.8. Práctica #8	108
5. CONCLUSIONES	118
6. RECOMENDACIONES.....	119
REFERENCIAS BIBLIOGRÁFICAS.....	120

ÍNDICE DE FIGURAS

Figura 1: PLC DELTA DVP SX2	4
Figura 2: HMI DOP-B10S615.....	5
Figura 3: Conversor TTL a RS-232.....	6
Figura 4: Módulo FireFly.	7
Figura 5: Conexión FireFly.....	7
Figura 6: Interruptor automático.....	7
Figura 7: MEGA 2560 R3 Board Black ATmega2560	8
Figura 8: Fuente de poder conmutada LOGO! Power 24VDC 2,5A	9
Figura 9: Motorreductor 70:	10
Figura 10: Matriz de sensores QRT-8A.....	11
Figura 11: Controlador de motor dual 12A.	11
Figura 12: Sensor de distancia GP2Y0A60SZ	12
Figura 13: Diseño Panel de Control.....	13
Figura 14: Elementos Internos.....	13
Figura 15: Diseño de la serigrafía en CAD e impreso en el acrílico	14
Figura 16: Interfaz Humano Maquina Delta.....	14
Figura 17: Modulo Firefly	15
Figura 18: Diseño Robot Móvil.....	16
Figura 19: Planos constructivos Robot Móvil	15
Figura 20: Sistema de Tracción Robot Móvil.....	16
Figura 21: Cobertor para sensores de obstrucción	16
Figura 22: Cobertor para sensores de obstrucción	17
Figura 23: Cobertor para sensores de piso.....	17
Figura 24: Tarjeta de Control Robot Móvil	17
Figura 25: Tarjeta de Control Robot Móvil	18
Figura 26: Diagrama de bloques.....	19
Figura 27: Diagrama de Conexiones eléctricas.....	19
Figura 28: Conexiones Entradas PLC.....	20
Figura 29: Topología de Comunicación Robot-Maleta Didáctica.....	21
Figura 30: Topología de comunicación	24
Figura 31: Conexión PLC en maleta didáctica.	25
Figura 32.2: Módulos RS-232 Bluetooth	26
Figura 34: Módulos RS-232 Bluetooth LED'S indicadores	27
Figura 35: Conexión USB de PLC Delta.	28
Figura 36: Ruta de ubicación de WPLSoft.	28
Figura 37: Ventana principal WPLsoft.....	29
Figura 38: Seleccionar tipo de PLC.	29
Figura 39: Configuración de comunicación.	30
Figura 40: Configuración de comunicación.	30
Figura 41: Parámetros puertos COM1.	31
Figura 42: Declaración de función MOVER.....	32
Figura 43: Configuración de COM1.....	32
Figura 44: Configuración de comunicación serial.....	33
Figura 45: Configuración de transferencia a PLC.....	34
Figura 46: Advertencia de carga de programación en PLC.....	34
Figura 47: Propiedades de accionamientos.	35
Figura 48: Ventana principal Arduino.	35
Figura 49: Configuración de Placa Arduino.....	36
Figura 50: Datos configurados de Placa Arduino.	36
Figura 51: Programación Arduino.	36

Figura 52: Compilación de programa Arduino.....	37
Figura 53: Variables para señales de sensores en Arduino	41
Figura 54: Variables Arduino para puerto serial	41
Figura 55: Configurar como Entrada/salidas, habilitación de puerto serial	42
Figura 56: lectura de sensores.....	42
Figura 57: Configuración de sensores por bit para Dato1	43
Figura 58: Configuración de sensores por bit para Dato1	43
Figura 59: Envío de datos al PLC	43
Figura 60: Configuración, envío y recepción de datos en PLC.....	44
Figura 61: Arreglo de recepción de datos	44
Figura 62: Desfragmentación de Dato1 a bits.....	45
Figura 63: Desfragmentación de Dato2 a bits.....	45
Figura 64: Ruta de ubicación de DOPsoft.....	45
Figura 65: Configuración de HMI.....	46
Figura 66: Configuración de comunicación HMI.....	47
Figura 67: Pantalla inicial DOPsoft.....	47
Figura 68: Barra de elementos.....	47
Figura 69: Barra de tipos de botones.....	48
Figura 70: Direccionamiento de Entradas/Salidas en HMI.....	48
Figura 71: Direccionamiento de marca "M16" en HMI.....	49
Figura 72: Selección de figura conforme a función de Marca "M16".....	50
Figura 73: Alineación y escalamientos de "M16".....	50
Figura 74: Plantilla Actuadores/indicadores.....	51
Figura 75: Declaración de variables para PID.....	54
Figura 76: Bloque de configuraciones y puerto serial.....	55
Figura 77: Bloque de configuraciones y calibración de sensores	55
Figura 78: Sintonización de control PID	56
Figura 79: Direccionamiento de Motores con control PID	56
Figura 80: Variables de entrada Seguidor ON/OFF	61
Figura 81: Condiciones de giro seguidor HISTÉRISIS.....	62
Figura 82: Declaración de variables para PID.....	62
Figura 83: Bloque de configuraciones y puerto serial.....	63
Figura 84: Bloque de configuraciones y calibración de sensores	63
Figura 85: Sintonización de control PID	63
Figura 86: Direccionamiento de Motores con control PID	64
Figura 87: Pista para seguidor de línea	65
Figura 88: Datos de muestreo seguidor de línea HISTÉRISIS-Motor derecho.....	66
Figura 89: Datos de muestreo seguidor de línea HISTÉRISIS-Motor izquierdo.....	66
Figura 90: Datos de muestreo seguidor de línea HISTÉRISIS-Tira de sensores.....	67
Figura 91: Datos de muestreo seguidor de línea PID - Motor dereho.....	67
Figura 92: Datos de muestreo seguidor de línea PID-Motor izquierdo.....	68
Figura 93: Datos de muestreo seguidor de línea PID-Tira de sensores.....	68
Figura 94: Analisis HISTÉRISIS-PID Motor derecho.....	69
Figura 95: Analisis HISTÉRISIS-PID Motor izquierdo.....	69
Figura 96: Analisis HISTÉRISIS-PID Motor izquierdo.....	70
Figura 97: Topología de comunicación	72
Figura 98: Configuración de comunicación serial en PLC	74
Figura 99: Programación control de robot móvil.....	75
Figura 100: Programación control de robot móvil.....	76
Figura 101: Ruta de ubicación de DOPsoft.....	77
Figura 102: Configuración de HMI.....	77
Figura 103: Configuración de comunicación HMI.....	78

Figura 104: Pantalla inicial DOPsoft.....	78
Figura 105: Barra de elementos.....	79
Figura 106: Barra de tipos de botones.	79
Figura 107: Configuración botón set/reset	80
Figura 108: Configuración de botones para dirección del robot móvil	80
Figura 109: : Topología de comunicación	85
Figura 110: Configuración Puerto de comunicación COM1.....	87
Figura 111: Configuración de código para comandar robot móvil.	88
Figura 112: Configuración de código para comandar robot móvil.	88
Figura 113: Configuración de estado.	89
Figura 114: Configuración de transferencia a PLC.....	89
Figura 115: Advertencia de carga de programación en PLC.....	90
Figura 116: Propiedades de accionamientos.	90
Figura 117: Ventana principal Arduino.	91
Figura 118: Configuración de Placa Arduino.	91
Figura 119: Datos configurados de Placa Arduino.	91
Figura 120: Configuración de variables de entrada.....	92
Figura 121: Calibración de sensores.....	92
Figura 122: Acondicionamiento de señales de sensores.	92
Figura 123: Compilación de programa Arduino.....	93
Figura 124: Topología de comunicación	96
Figura 125: Configuración Puerto de comunicación COM1.....	98
Figura 126: Configuración de comunicación serial.....	99
Figura 127: Configuración de código para comandar robot móvil.	99
Figura 128: contador de posición de robot móvil.....	100
Figura 129: Configuración de transferencia a PLC.....	101
Figura 130: Advertencia de carga de programación en PLC.....	101
Figura 131: Propiedades de accionamientos.	102
Figura 132: Ventana principal Arduino.	102
Figura 133: Configuración de Placa Arduino.....	103
Figura 134: Datos configurados de Placa Arduino.	103
Figura 135: Declaración de variables en Arduino.....	103
Figura 136: calibración de sensores.	104
Figura 137: Bloque principal Arduino.	104
Figura 138: Bloque acondicionamiento de sensores.....	105
Figura 139: Bloque para direccionamiento de motores.	105
Figura 140: Compilación de programa Arduino.....	106
Figura 141: Topología de comunicación	109
Figura 142: Configuración Puerto de comunicación COM1.....	111
Figura 143: Reordenamiento de cabecera.....	112
Figura 144: Condiciones Robot laberinto.	112
Figura 145: Configuración de transferencia a PLC.....	113
Figura 146: Advertencia de carga de programación en PLC.....	113
Figura 147: Propiedades de accionamientos.	114
Figura 148: Ventana principal Arduino.	114
Figura 149: Configuración de Placa Arduino.....	115
Figura 150: Datos configurados de Placa Arduino.	115
Figura 151: Programación Arduino.	115
Figura 152: Compilación de programa Arduino.....	116

ÍNDICE DE TABLAS

Tabla 2.2: Características Arduino MEGA.	8
Tabla 4.1: Definición de Puertos PLC	24
Tabla 4.2: Definición de Puertos HMI.....	25
Tabla 4.3: Definición de Puertos Arduino MEGA	25
Tabla 4.4: Definición de módulos Firefly RS-232/Bluetooth Maleta didáctica	27
Tabla 4.5: Definición de módulos Firefly RS-232/Bluetooth robot móvil	27
Tabla 4.6: Definición de LED Módulo Firefly	27
Tabla 4.7: Resultados práctica#1.....	37
Tabla 4.8: Definición de Sensores según ubicación.....	40
Tabla 4.9: Definición de Sensores	42
Tabla 4.10: Resultados práctica#2.....	51
Tabla 4.11: Parametrización de sensores para PID	54
Tabla 4.12: Resultados práctica#2.....	56
Tabla 4.13: Resultados práctica#3.....	57
Tabla 4.14: Parametrización tira de sensores	60
Tabla 4.15: Definición de Puertos PLC	73
Tabla 4.16: Definición de Puertos HMI.....	73
Tabla 4.17: Definición de Puertos Arduino MEGA.....	74
Tabla 4.18: Resultados práctica#5.....	81
Tabla 4.19: Definición de Puertos PLC	86
Tabla 4.20: Definición de Puertos HMI.....	86
Tabla 4.21: Definición de Puertos Arduino MEGA.....	87
Tabla 4.22: Resultados práctica#6.....	93
Tabla 4.23: Definición de Puertos PLC	97
Tabla 4.24: Definición de Puertos HMI.....	97
Tabla 4.25: Definición de Puertos Arduino MEGA.....	98
Tabla 4.26: Resultados práctica#7.....	106
Tabla 4.23: Definición de Puertos PLC	110
Tabla 4.24: Definición de Puertos HMI.....	110
Tabla 4.25: Definición de Puertos Arduino MEGA.....	110
Tabla 4.26: Resultados práctica#8.....	116

INTRODUCCIÓN

El uso de la robótica tiene gran magnitud de crecimiento en el ambiente industrial, médico y comercial, ya que se pueden realizar actividades y/o procesos automáticos que para el ser humano pueden resultar agotadores, peligrosos e incluso no tan precisos, causa principal y donde más sobresale ya que un robot es el resultado del trabajo conjunto de mecanismos con sistemas electrónicos para el control, desplazamiento y desarrollo de procesos completos, permitiendo elevar el nivel de productividad y calidad del producto para el cual este destinado, se pueden clasificar de la siguiente manera:

- Repite tareas programadas con intervalos de tiempo programados.
- Dependiendo del escenario las posibles alteraciones de su entorno no le afectan.
- Puede clasificar y detectar esfuerzos adaptando su potencia conforme a los trabajos programados

El proyecto está enfocado en el desarrollo e implementación de un robot móvil prototipo con enlace inalámbrico a una maleta didáctica, gobernados por un microcontrolador Arduino y un controlador lógico programable de la marca DELTA consecutivamente, se dispondrá de una nueva plataforma de programación propia que no necesita de licencias con algún valor monetario para su uso, facilitando y a la vez generando así nuevos criterios de programación y aplicativos de Robótica y Automatización, desarrollado así nuevos conocimientos y prácticas con un robot móvil no tripulado orientado a la robótica y automatización industrial.

1. EL PROBLEMA

1.2. Antecedentes

Los módulos de prácticas que están a disposición de los estudiantes en los laboratorios de la Universidad Politécnica Salesiana sede Guayaquil, están equipados con PLC's estandarizados a una sola marca. La diversidad de conocimientos de investigación que los estudiantes demandan, genera esa necesidad de contar con una gama más amplia en cuanto a tipos de controladores programadores, siendo esta la principal problemática y sumándose también a esta, se expone la falta de prácticas utilizando prototipos robóticos controlados desde un PLC.

1.3. Importancia y Alcances

Generar nuevas tendencias de conocimientos referentes a la programación industrial y aplicación de nuevos equipos industriales que involucrará directamente con la robótica industrial, utilizando diversos protocolos de comunicación alámbrica e inalámbrica, microcontroladores y controladores lógicos programables.

La magnitud del proyecto contempla aplicaciones de un proceso automático con sensado y transporte de carga.

Por consiguiente, construye un aporte más en los recursos didácticos de la implementación técnica de la carrera de Ingeniería Electrónica en la Universidad Politécnica Salesiana sede Guayaquil, ya que este proyecto permitirá a los maestros complementar con la práctica, la teoría referente a controladores lógicos programables, control y robótica.

1.4. Beneficiarios

El aporte de este proyecto está directamente relacionado con la carrera de Ingeniería Electrónica de la Institución ya que los maestros y estudiantes dispondrán de la maleta didáctica para realizar las prácticas propuestas e investigar nuevos aplicativos con el robot móvil.

1.5. Delimitación

1.5.1. Temporal

La implementación de este proyecto tiene el lapso de un año desde la aprobación del mismo.

1.5.2. Espacial

El proyecto está destinado para el uso en los laboratorios de automatización y control de la Universidad Politécnica Salesiana sede Guayaquil.

1.5.3. Académica

El desarrollo e implementación de este proyecto está orientado a sistemas de control aplicado con controladores lógicos programables, microcontroladores, sistemas embebidos, sensores y actuadores. Utiliza 3 tipos de entornos de programación según el equipo que se necesite programar.

1.6. Objetivos

1.6.1. Objetivo general

Diseñar e implementar un robot móvil controlado por un PLC DELTA para aplicaciones didácticas de automatización y control.

1.6.2. Objetivos específicos

- Diseñar e implementar un módulo didáctico compacto para el controlador Delta.
- Diseñar la estructura adecuada para un robot móvil.
- Establecer enlaces inalámbricos entre el módulo didáctico y el robot móvil.
- Fomentar el uso de las herramientas de programación Arduino y WPLsoft para desarrollo futuro de procesos industriales aplicados a robótica.
- Implementar un control PID para el desplazamiento automático del robot móvil mediante detección de línea negra.
- Comparar el comportamiento de un control PID frente a un control on/off.
- Desarrollar una interfaz hombre-máquina para todo el proceso utilizando el software de programación DOPsoft.

- Desarrollar un manual de 8 prácticas de laboratorio que involucren procesos orientados a robótica móvil u automatización.

2. FUNDAMENTOS TEÓRICOS

2.1. Controlador programable DELTA DVP SX2

Equipos industriales, con aplicaciones estables y de alta fiabilidad en todo tipo de máquinas de automatización. Además de la operación lógica rápida, instrucciones abundantes y varias tarjetas de función.

Son compatibles con varios protocolos de comunicación, tiene conexión a unidades Delta AC motor, servo, interfaz hombre-máquina y el controlador de temperatura a través de la red industrial en una completa. La serie DVP-SX2 está equipada con:

- Puntos de E/S MPU: 20(8DI/6DO, 4AI/2AO).
- Max. Puntos E/S: 494(14+480).
- Capacidad de Programación: 16K pasos.
- Velocidad de ejecución del programa: 0.35-1us.
- Puerto RS-232 & RS-485, compatible con Modbus normal en protocolo ASCII/RTU. Puede ser Maestro o Esclavo, USB mini para la carga y descarga del programa y monitoreo.
- Soporta 4 salidas a pulso de alta velocidad: 2p a 100 kHz y 2p a 10 kHz.
- 8 puntos de entrada de alta velocidad: 2p 100 kHz y 6p 10 kHz. (DELTA, 2017)



Figura 1: PLC DELTA DVP SX2 (DELTA, 2017)

2.2. Panel Táctil HMI DELTA serie B

Plataforma que permite la interacción entre los usuarios y equipos de automatización. La serie B Contiene una pantalla táctil multicolor, cuenta con varios puertos de comunicación con protocolos como RS-485, RS-232, USB host, USB esclavo.

Permite la introducción de parámetros intuitiva y una gran variedad de formas de mostrar los datos que son enlazados a un controlador compatible, incluyendo gráficos de tendencias y elementos de alarma.

- 10.1 "(1024 x 600 píxeles) LCD TFT de 65.536 colores
- 3 series de puertos COM, el apoyo RS232 / RS422 / RS485
- Para la transferencia de datos/descarga: RS232, USB
- Compatible con USB Host, conexión directa al disco USB, impresora y ratón.



Figura 2: HMI DOP-B10S615

2.3. Conversor TTL a RS-232

Este adaptador ayuda a convertir señales de nivel TTL (Logic Transistor-Transistor) a la interfaz RS-232, cuenta con un conector DB9. Adecuado para usar con la mayoría de los microcontroladores, de 300 a 115.200bps, tiene dos niveles de alimentación 3.3V y 5V, pines RTS, y CTS.

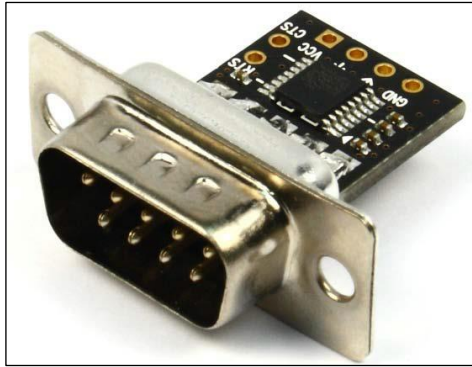


Figura 3: Conversor TTL a RS-232

2.4. Módulo Firefly

Firefly es un módem Bluetooth Clase 1 con antena de chip de cerámica RF de 2.4GHz. Compatible con la versión 1.2 de Bluetooth. Se pueden configurar dos dispositivos Firefly para realizar un reemplazo o extensión de cable inalámbrico RS232. (Connect, 2018)

2.4.1. Características de Módulo FireFly

Se caracteriza por su facilidad de configuración en parámetros como:

- Velocidad de transmisión: 9600, 19200, 38400, 57600, 115200, 232400bps, las velocidades diferentes a 9600 o 115200 se configuran en modo de comando.
- Paridad par, impar o ninguna, 7/8 bits de datos, 1 bit de parada.
- Control de flujo de hardware a través de RTS, CTS. NO son señales de control del módem.
- Radio Bluetooth TM de alta potencia (clase 1, 20dB TX, 330', 100m)
- Conector DB9 para conexión directa con equipos RS-232
- Led de estado, verde (conexión) y amarillo (TX / RX), rojo (error).
- A operaciones de baja potencia requiere solo 3.3VDC a 40mA cuando transmite y está conectado, en modo de espera. Consumé solo 1mA
- Alimentado por un adaptador externo a 5VDC. Manteniendo un rango de voltaje tolerable de 4-9VDC. Puede conectarse utilizando el conector DB9. (Connect, 2018)



Figura 4: Módulo FireFly. (Connect, 2018)

2.4.2. Conexiones físicas

La configuración de pines se muestra en la siguiente tabla.

Pin	Description	Source
1 -DCD	Carrier Detect (NOT Connected)	
2 - RD	Receive Data	Attached Device (In)
3 - TD	Transmit Data	Host Device (Out)
4 - DTR	Data Terminal Ready (NOT Connected)	
5 - GND	Ground	
6 - DSR	Data Set Ready (NOT Connected)	
7 - RTS	Request to Send	Host Device (Out)
8 - CTS	Clear to Send	Attached Device (In)
9 -PWR	4-7 VDC	External Power Supply

Figura 5: Conexión FireFly (Connect, 2018)

2.5. Interruptor automático 6KA 1POL C2

Interruptor con mecanismo de protección automático de montaje en panel con unidad de disparo termo-magnética un polo, 2 Amperios de corriente nominal a 40°C, trabaja a 50 y 60 Hertzios con 20000 ciclos de mecanismo. Trabaja a un a temperatura ambiente de -40 a 70 °C. (Electric, 2018)



Figura 6: Interruptor automático. (Electric, 2018)

2.6. Arduino MEGA 2560 R3

Es una tarjeta de desarrollo open-source construida con un microcontrolador Atmega2560 que posee pines de entradas y salidas (E/S), analógicas y digitales. Esta tarjeta es programada en un entorno de desarrollo que implementa el lenguaje Processing/Wiring.

Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede comunicarse a un PC a través del puerto serial (conversión con USB) utilizando lenguajes como Flash, Processing, MaxMSP, etc. Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación. (Elegoo, 2017)



Figura 7: MEGA 2560 R3 Board Black ATmega2560 (Elegoo, 2017)

El Arduino Mega tiene 54 pines de entradas/salidas digitales (14 de las cuales pueden ser utilizadas como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serial por hardware), cristal oscilador de 16MHz, conexión USB, jack de alimentación, conector ICSP y botón de reset. Arduino Mega incorpora todo lo necesario para que el microcontrolador trabaje; simplemente conéctalo a tu PC por medio de un cable USB o con una fuente de alimentación externa (9 hasta 12VDC). (Elegoo, 2017)

Tabla 2.1: Características Arduino MEGA. (Elegoo, 2017)

Microcontrolador	ATmega2560
Tensión de funcionamiento	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V

Pines de E / S digitales	54 (de los cuales 15 proporcionan salida de PWM)
Corriente DC por Pin E / S	20 mA
Corriente DC para 3.3V Pin	50 mA
Memoria flash	256 KB de los cuales 8 KB utilizados por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz
LED_BUILTIN	13

2.7. Fuente LOGO! Power 110VAC/24VDC 2.5A

LOGO! Power son fuentes de alimentación conmutadas para la conexión de un sistema de corriente alterna monofásicos a sistemas de corriente continua. Regulada electrónicamente y se puede variar con un potenciómetro que se encuentra en la portada del dispositivo. La salida del dispositivo está aislada y a prueba de cortocircuitos. La luz LED indica el estado de funcionamiento. (Siemens, 2018)



Figura 8: Fuente de poder conmutada LOGO! Power 24VDC 2,5A (Siemens, 2018)

2.8. Motorreductor 70:1

Este motorreductor es un potente motor de tipo escobillas funciona con voltaje de corriente continua a 12V, con caja de engranajes metálica con reducción 70:1.

Especificaciones principales:

- Voltaje de alimentación: 12V
- Revoluciones por Minuto. 200 RPM
- Corriente de consumo: 300 mA en funcionamiento libre, 5A pico.
- Torque: 170 oz-in (12 kg-cm) (Pololu Corporation , 2018)

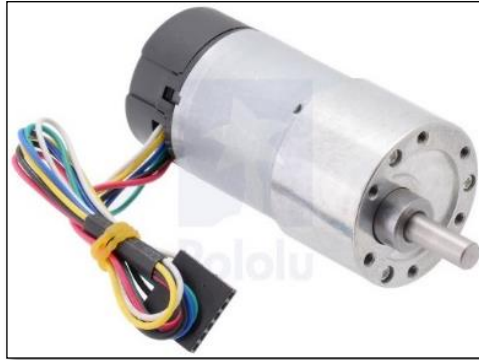


Figura 9: Motorreductor 70:1 (Pololu Corporation , 2018)

2.9. Matriz de sensores QTR-8A

El conjunto de sensores de la serie QTR está diseñado como un sensor de línea, pero puede usarse como un sensor de proximidad de propósito general. El módulo contiene una tira 8 pares de sensores un emisor y otro receptor IR (fototransistor) espaciados a 9.525mm.

Cada sensor está conectado a una resistencia pull-up creando un divisor de voltaje que será la señal de salida de voltaje analógica entre 0 V y VIN. Las salidas son todas independientes, pero los emisores están dispuestos en pares para reducir a la mitad el consumo de corriente.

El consumo de cada sensor es de 20 a 25mA, lo que hace que el consumo total de la tarjeta sea ligeramente inferior a 100mA. (Pololu Corporation , 2018)

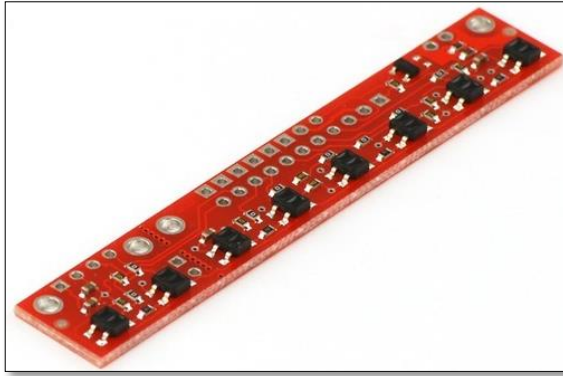


Figura 10: Matriz de sensores QRT-8A (Pololu Corporation , 2018)

2.10. Controlador de motor dual 12A

Sabertooth 2X12 es uno de los controladores de motores de corriente continua considerado uno de los más eficientes del mercado. Es adecuado para robots de potencia media: hasta 30 lb en combate o 100 lb para robótica de uso general. Puede suministrar hasta 12 Amperios a cada motor con corrientes de pico a 25A.

Cuenta con protecciones de contra corrientes, sobre-corrientes y térmica, brindando la tranquilidad al operador de posibles errores de conexión

Este controlador admite señales de control de tipo: (Engineering, 2004)

- Voltaje analógico
- Control de radio RF
- Serial
- Serial empaquetado.

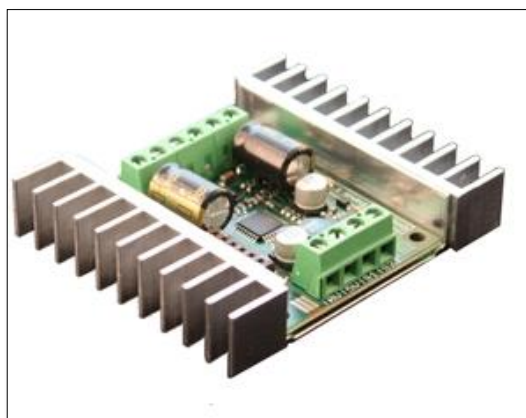


Figura 11: Controlador de motor dual 12A. (Engineering, 2004)

2.11. Sensor de distancia Analógico

El sensor de distancia GP2Y0A60SZ de Sharp ofrece un amplio rango de detección de 4 "a 60" (10 cm a 150 cm) y una alta tasa de actualización de 60 Hz. La distancia se indica mediante una tensión analógica, por lo que solo se requiere una entrada analógica para interactuar con el módulo. (Pololu Corporation , 2018)

- Tensión de funcionamiento:
- Versión 5V: 2.7 V a 5.5 V
- Consumo de corriente promedio: 33 mA (típico)
- Rango de medición de distancia: 10 cm a 150 cm (4 "a 60")
- Tipo de salida: voltaje analógico
- Diferencial de voltaje de salida sobre el rango de distancia versión 5V: 3.0 V (típico)
- Periodo de actualización: 16.5 ± 4 ms
- Peso sin pasadores de cabezal: 2.5 g (0.09 oz)



Figura 12: Sensor de distancia GP2Y0A60SZ (Pololu Corporation , 2018)

3. MARCO METODOLÓGICO

El proyecto técnico se divide en dos partes, maleta didáctica y el robot móvil, el mismo va ser controlado desde la maleta.

3.1. Diseño de compartimentos para maleta didáctica

En el diseño de la maleta se tomó en cuenta la distribución de los elementos de acuerdo a sus dimensiones entre ellos están el PLC Delta DVP-SX2, el panel táctil HMI Delta y el conversor TTL a RS-232 además de un compartimento donde se guardará el robot móvil. Se observa la estructura del panel de control, posee un pistón neumático para abrir y cerrar la misma.

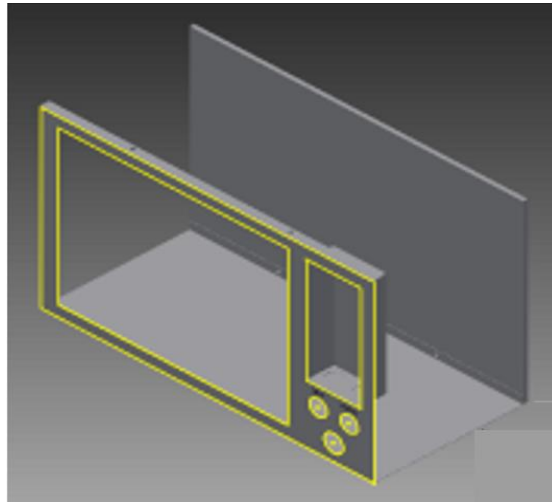


Figura 13: Diseño Panel de Control

Se cuenta con un plafón con riel para colocar el PLC, la fuente de tensión a 24 VDC, el breaker principal y las borneras para distribución de señales de alimentación y digitales, en la parte izquierda de la maleta se encuentra el switch de encendido y apagado, además del enchufe para el cable de alimentación a 110 Voltios.



Figura 14: Elementos Internos

3.1.1. Autómata Programable Delta DVP-SX2

El controlador lógico programable Delta es un autómata para el ambiente industrial el cual cuenta con varias entradas/salidas digitales y analógicas, diversos tipos de comunicación serial integradas en un solo equipo, para evitar en lo posible estructuras modulares.



Figura 15: Diseño de la serigrafía en CAD e impreso en el acrílico

3.1.2. HMI Delta

El interfaz humano máquina delta, permite observar la información proporcionada por el autómata programable en tiempo real, para interactuar con los elementos fijos y móviles del prototipo durante el desarrollo del proceso intuitivamente.



Figura 16: Interfaz Humano Maquina Delta

3.1.3. Módulo Firefly

Módulo de comunicación inalámbrica bluetooth clase A con un rango de alcance sin obstáculos de 100 metros con conectores db9 y comunicación serial RS-232, utilizado para la interacción de la maleta didáctica con el robot móvil.



Figura 17: Modulo Firefly

3.2. Robot móvil

El diseño del robot está basado en sistemas diferenciales de cuatro ruedas y dos motores eléctricos de tracción, la estructura metálica soporta en superficies planas un peso máximo de 5 libras, con un espacio de carga limitado a 28 x 22cm.

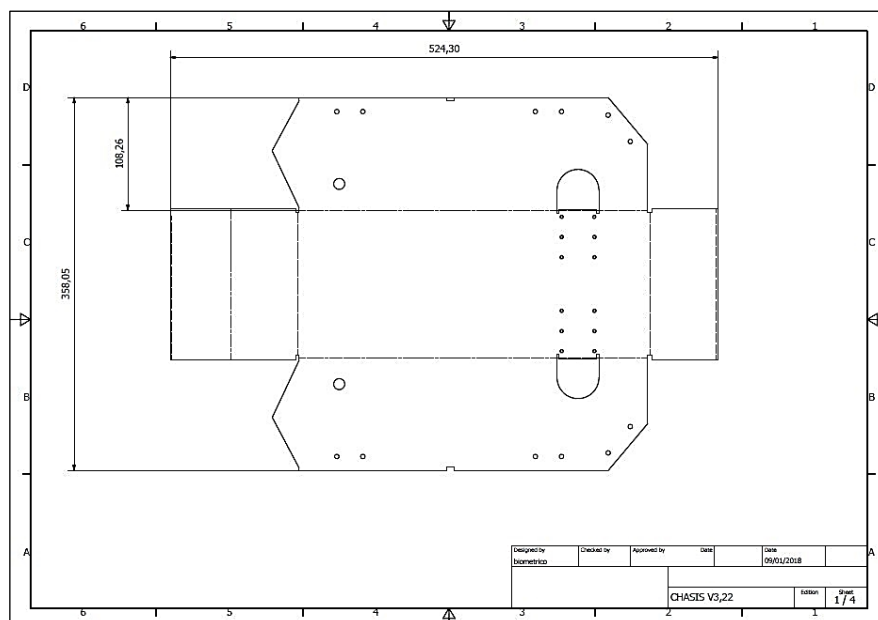


Figura 18: Planos constructivos Robot Móvil

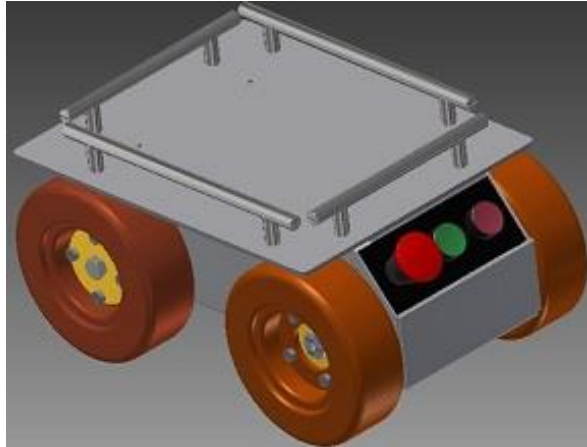


Figura 19: Diseño 3D Robot Móvil

El robot tiene incorporado en su estructura tres pulsadores y dos indicadores luminosos que están a disposición del usuario conforme a la práctica, uno de paro general que desactivará la alimentación principal del mismo, un botón de paro de emergencia para desconectar el circuito de fuerza para los motores, dos sensores de obstrucción en los laterales y no en la estructura.

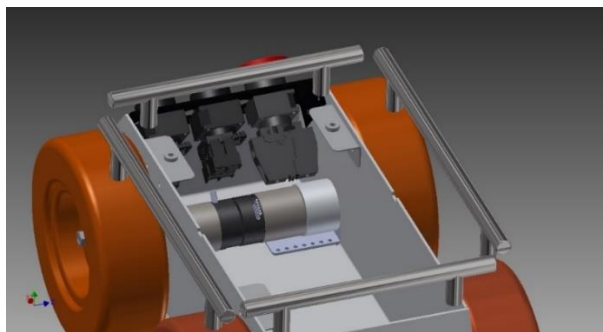


Figura 20: Sistema de Tracción Robot Móvil

Los cobertores para los sensores fueron diseñados para adaptarse a la estructura del robot móvil.

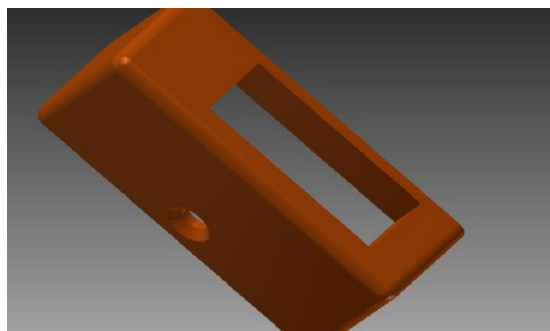


Figura 21: Cobertor para sensores de obstrucción

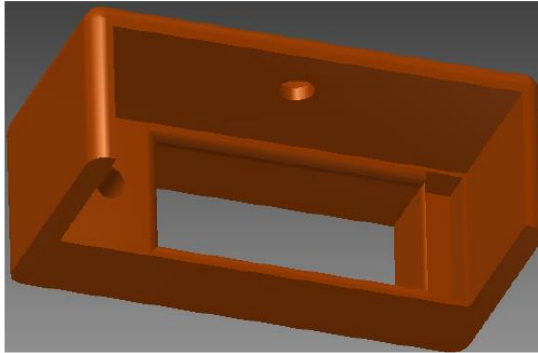


Figura 22: Cobertor para sensores de obstrucción

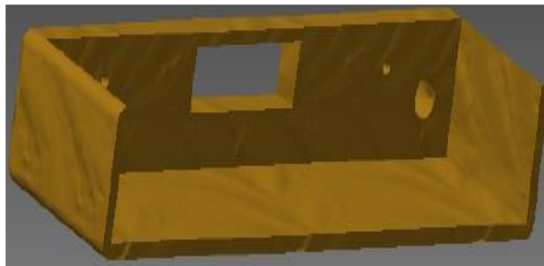


Figura 23: Cobertor para sensores de piso

3.2.1. Diseño de la Tarjeta robot

La PCB para el robot fue diseñada en Proteus, se tomó en cuenta la medida de los elementos que van dentro del prototipo, se indica el funcionamiento de las entradas/salidas digitales y analógicas en las aplicaciones pre-programadas.

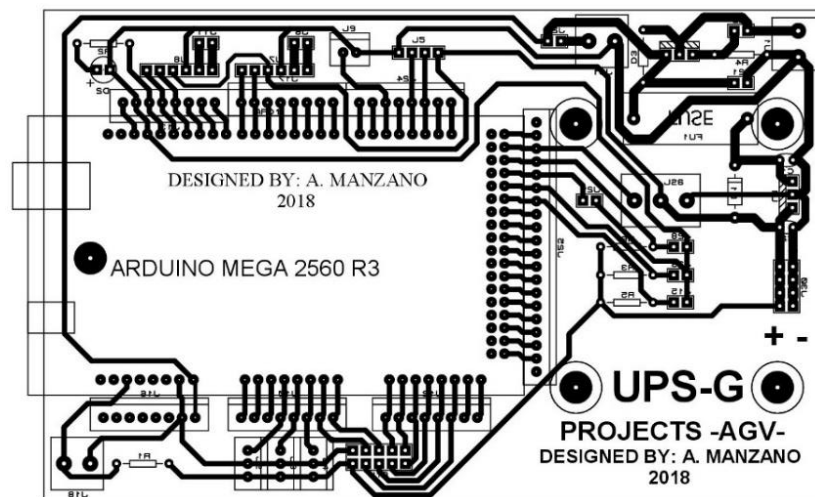


Figura 24: Tarjeta de Control Robot Móvil

3.2.2. Driver Motores

Este hardware controlara el movimiento y la dirección de los motores por medio del puerto serial que recibe señales en bits enviadas desde el autómata programable Delta, se alimenta por medio del voltaje que proporcionan las baterías del robot móvil, se encarga de acondicionar el voltaje para el cambio de giro en los motores dependiendo de la rutina programada en el prototipo.

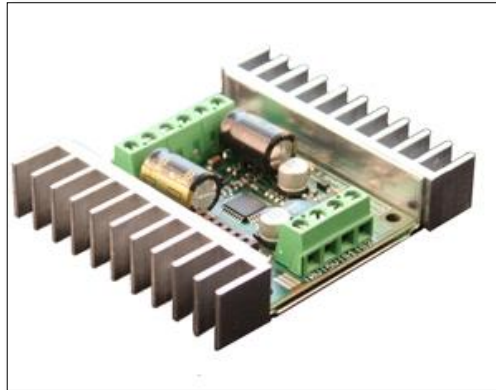


Figura 25: Tarjeta de Control Robot Móvil

3.3. Diagrama de bloques

La maleta didáctica está diseñada para trabajar con un sistema en Lazo cerrado, con la interacción HMI-PLC mediante el puerto RS-485, seguido de las señales de entrada y salida que están conectadas al autómata programable, el mismo que se comunica a través de un módulo inalámbrico RS-232 al robot móvil, mientras tanto;

El robot móvil ejecuta las órdenes enviadas por el autómata programable a través de un microcontrolador, el mismo que actúa y/o envía una respuesta, este a la vez recepta y empaqueta las señales emitidas por el panel de control y los sensores locales. Para el control de motores de dirección el microcontrolador utilizara el puerto serial designado.

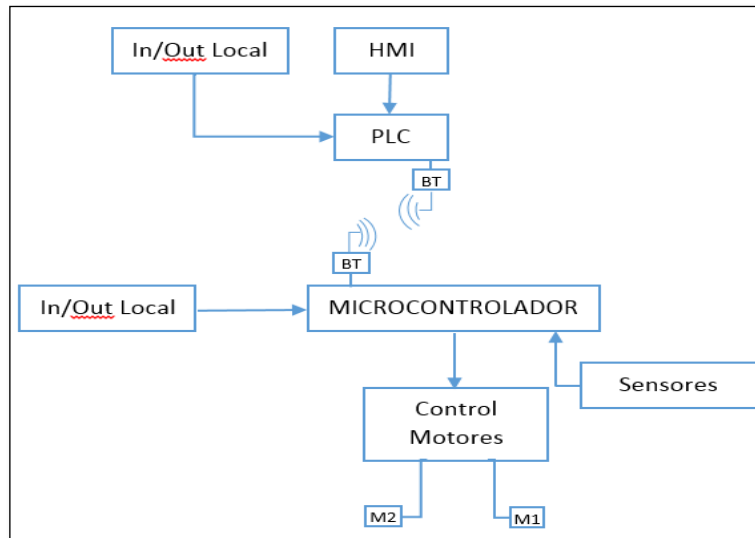


Figura 26: Diagrama de bloques

3.4. Conexiones internas para maleta didáctica

Para la maleta didáctica internamente se tienen las conexiones eléctricas mostradas, se energiza con una tensión de 110 VAC a una fuente Siemens que entrega 24 VDC en su salida para alimentar los siguientes equipos: PLC Delta, HMI Delta.

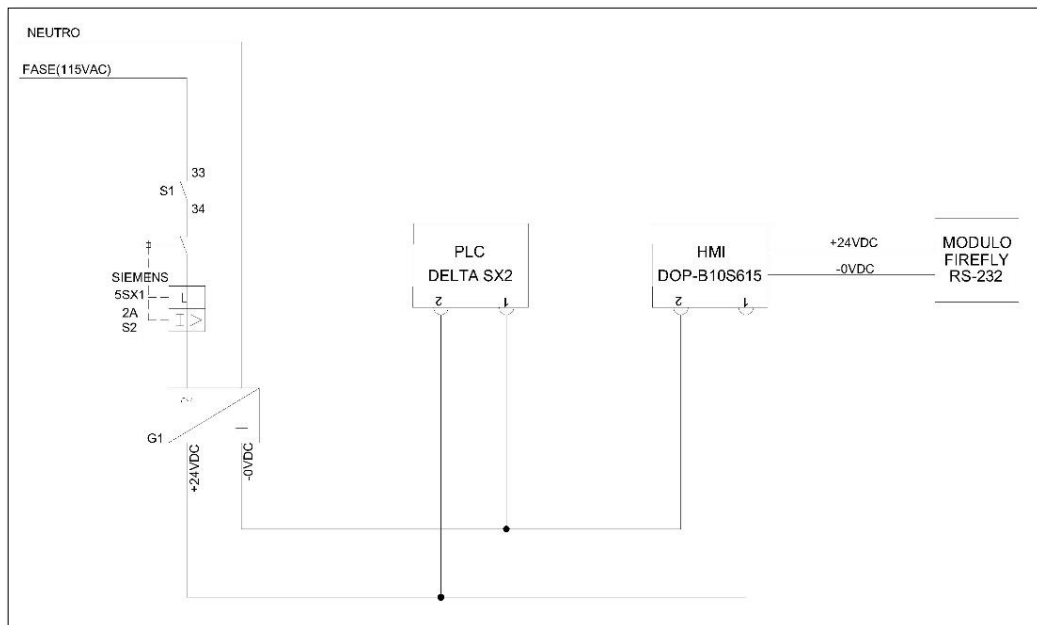


Figura 27: Diagrama de Conexiones eléctricas

En la se tienen las conexiones a las entradas digitales del autómata programable a una tensión de 24 VDC.

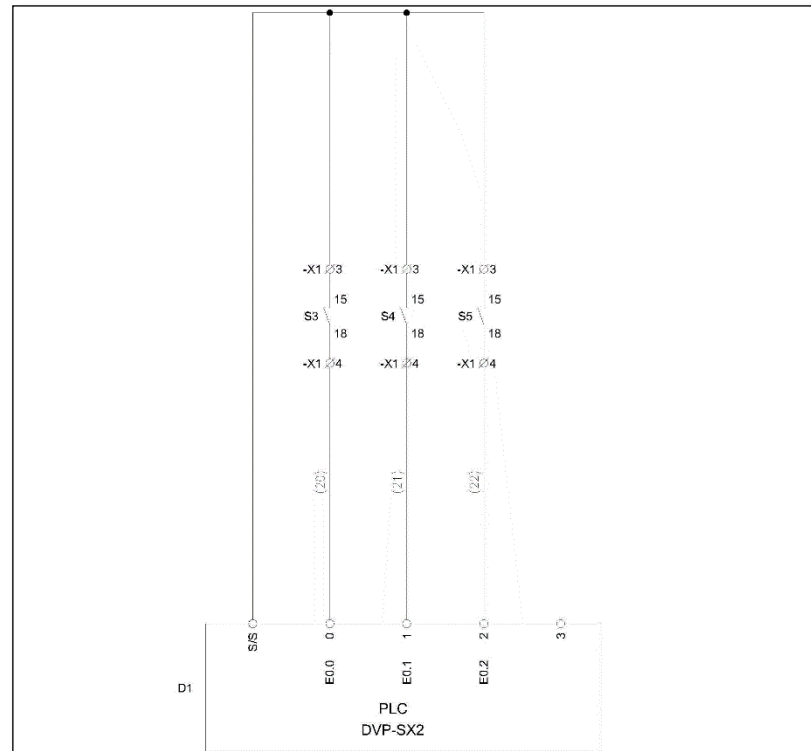


Figura 28: Conexiones Entradas PLC

3.5. Topología de comunicación

En la topología de comunicaciones definida para la maleta didáctica y el robot móvil se utiliza cuatro tipos de comunicación identificados como:

- RS-232
- RS485
- Serial TTL
- USB

Esta topología se mantiene para el efecto de las prácticas propuestas la misma que se configura de la siguiente manera:

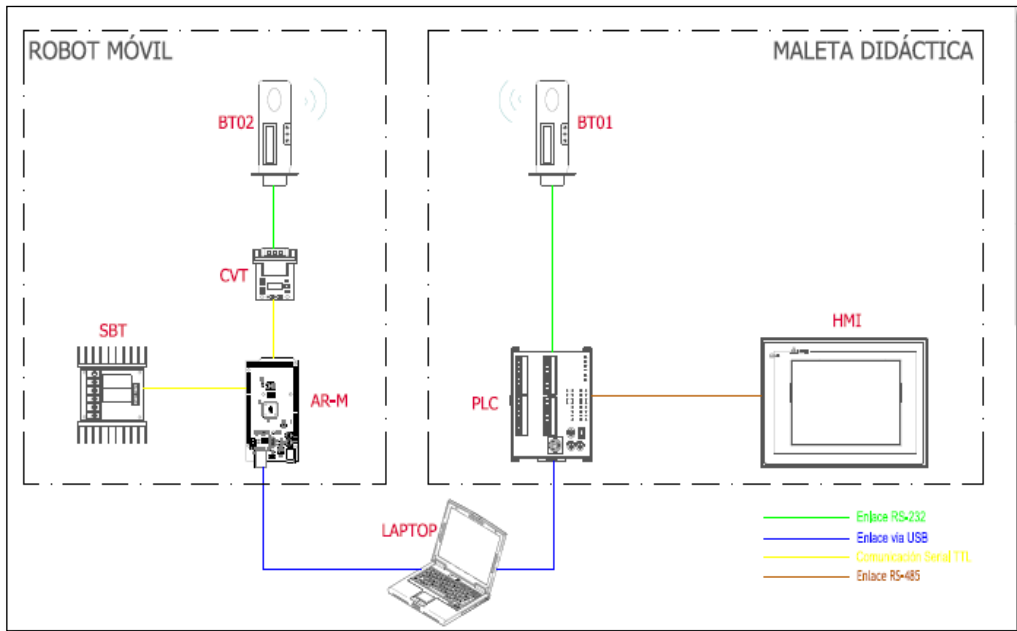



Figura 29: Topología de Comunicación Robot-Maleta Didáctica

4. PRÁCTICAS DE LABORATORIO

4.1. Práctica #1

		GUÍA DE PRÁCTICA DE LABORATORIO	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Automatización Industrial	
NRO. PRÁCTICA:	1	TÍTULO PRÁCTICA: Enlace del módulo didáctico compacto con el robot móvil	
Objetivo: <ul style="list-style-type: none">• Establecer una comunicación inalámbrica entre un PLC de la marca DELTA y un controlador Arduino MEGA.			
Objetivos específicos: <ul style="list-style-type: none">• Reconocer los protocolos de comunicación.• Asignar puertos de comunicación en PLC y Arduino MEGA.• Configurar módulo inalámbrico Bluetooth con interfaz RS232• Elaborar un programa en el software WPLsoft, que sea capaz de enlazar el PLC Delta con una tarjeta Arduino MEGA.• Elaborar un programa en el software Arduino, que sea capaz de enlazar el Arduino MEGA con el PLC Delta.• Realizar pruebas de comunicación con un indicador de estado			
INSTRUCCIONES:		<ul style="list-style-type: none">• Disponer de un computador con programas ya instalados Arduino, WPLSoft.• Disponer de un punto de alimentación 110VAC para la maleta didáctica.• Verificar que la botonera de paro de emergencia no este accionada en la maleta y el robot móvil.• Deslizar a “ON” el switch de encendido general ubicado en la maleta didáctica.	

- Visualizar los indicadores de encendido tanto en tarjeta del robot móvil como en maleta didáctica.

ACTIVIDADES A DESARROLLAR

- **Paso 1.** Reconocimiento de topología de comunicación
- **Paso 2.** Asignación puertos de comunicación en PLC, HMI y Arduino MEGA
- **Paso 3.** Configuración de módulos inalámbricos Firefly RS-232/Bluetooth
- **Paso 4.** Configuración de programa WPLsoft para PLC.
- **Paso 5.** Creación de código en WPLsoft.
- **Paso 6.** Configuración de Programa ARDUINO para robot móvil.
- **Paso 7.** Creación de código en Arduino.
- **Paso 8.** Pruebas de enlace entre PLC y Robot móvil.

Paso 1. Reconocimiento de topología de comunicación.

En la topología de comunicaciones definida para la maleta didáctica y el robot móvil se utiliza cuatro tipos de comunicación identificados como:

- RS-232
- RS485
- Serial TTL
- USB

Los mismos que se configuran de la siguiente manera:

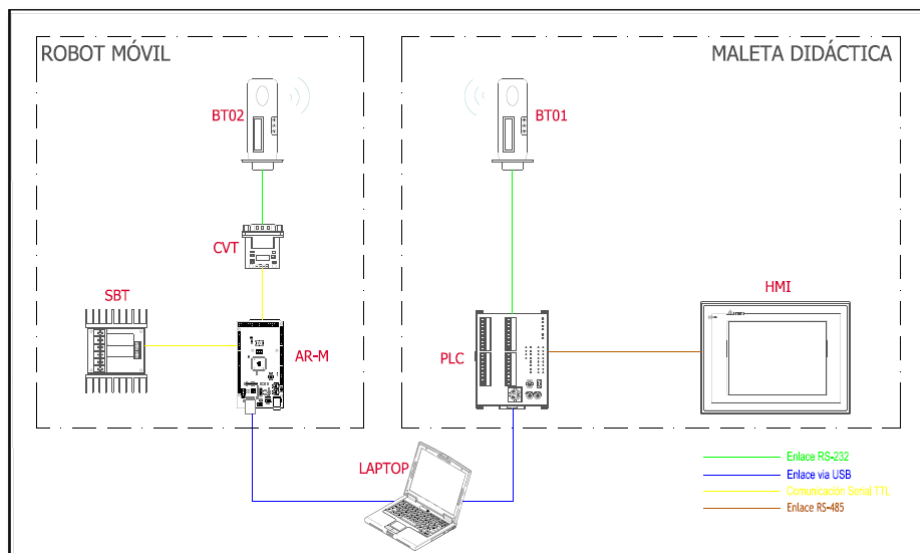


Figura 30: Topología de comunicación

Nota: La topología mostrada en la Figura anterior, solo se muestra los tipos de enlaces de comunicación utilizados; fuentes y tipos de alimentación no se describen.

La configuración de la topología no es modificable para ninguna de las prácticas aquí descritas.

Paso 2. Asignación puertos de comunicación en PLC, HMI y Arduino MEGA

Según la topología de comunicación los puertos del **PLC** se definen de la siguiente manera:

Tabla 4.1: Definición de Puertos PLC

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	Comunicación con Robot Móvil
COM2	RS-485	Comunicación con HMI
COM3	Mini USB	Programación de PLC

El puerto serial COM3 del PLC cuenta con una extensión USB hembra tipo A acoplada a un costado de la maleta didáctica la misma que se utilizará plena y únicamente para cargar o descargar códigos de programación del PLC.

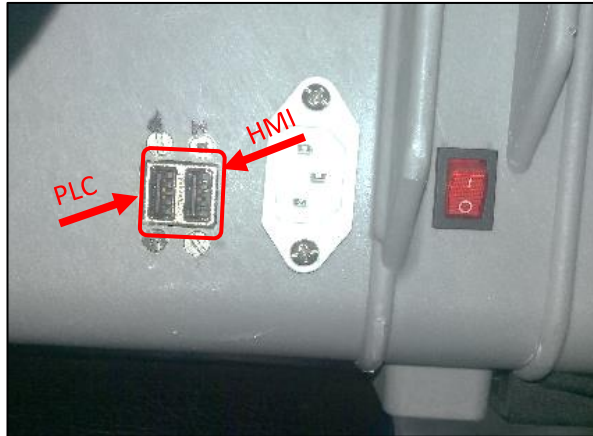


Figura 31: Conexión PLC en maleta didáctica.

El **HMI** define sus puertos de comunicación como:

Tabla 4.2: Definición de Puertos HMI

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	NO conectado
COM2	RS-232/ RS-422/ RS-485	Comunicación RS-485(PLC)
COM3	RS-232/ RS-422/ RS-485	NO conectado
USB host	USB tipo A	Alimentación 5VDC, Firefly
USB esclavo	USB tipo B	Programación de HMI

La Arduino **MEGA** configura los puertos de comunicación:

Tabla 4.3: Definición de Puertos Arduino MEGA

NOMBRE DEL PUERTO	(x)PIN Arduino MEGA		USO
Serial0	(0)TX	(1)RX	Programación MEGA
Serial1	(18)TX	(19)RX	Comunicación Módulo Firefly RS-232/Bluetooth(PLC)

Serial2	(16)TX	(17)RX	Conexión Sabertooth
Serial3	(14)TX	(15)RX	NO conectado

Paso 3. Configuración de módulos inalámbricos Firefly RS-232/Bluetooth

La transmisión inalámbrica entre la maleta didáctica y el robot móvil se efectúa gracias a los módulos Firefly RS-232/Bluetooth; los mismos que necesitan de una configuración inicial, esta configuración se realiza a través de conmutadores instalados en la parte posterior donde se puede configurar parámetros como:

- Velocidad de transmisión: (9600=ON; 115k=OFF)
- Modo Maestro/esclavo: (Maestro=ON; Esclavo=OFF)
- Auto búsqueda: (ON; OFF)
- Configuración de fábrica: (ON; OFF)



Figura 32.33: Módulos RS-232 Bluetooth

Los módulos Firefly RS-232/Bluetooth utilizan una fuente de alimentación externa a 5VDC/300mA, a través de Jack de conexión.

Configuración de Módulos

Módulo 1. Maleta didáctica:

Tabla 4.4: Definición de módulos Firefly RS-232/Bluetooth Maleta didáctica

NUMERO DE CONMUTADOR	ON	OFF
1	X	
2		X
3	X	
4		X

Módulo 2. Robot Móvil:

Tabla 4.5: Definición de módulos Firefly RS-232/Bluetooth robot móvil

NUMERO DE CONMUTADOR	ON	OFF
1	X	
2	X	
3	X	
4		X

Los módulos Firefly RS-232/Bluetooth cuenta con tres LED's indicadores de estados con su función específica según el color, a continuación, se detalla:



Figura 34: Módulos RS-232 Bluetooth LED'S indicadores

Tabla 4.6: Definición de LED Módulo Firefly

# LED	Color de LED	Estado	Función
1	Verde	Continúo / Parpadeante	Dispositivo encendido y enlazado con módulo 1 con 2 / Encendido pero no enlazado.
2	Naranja	Parpadeante o continuo	Se enciende conforme se envíe o reciba datos del módulo 1 al módulo 2.
3	Rojo	Parpadeante o continuo	Presencia de errores en la comunicación

Paso 4. Configuración de programa WPLsoft para PLC.

Utilizar el cable USB-USB para conectarse con el computador, el PLC se reconoce como: **Silicon Labs CP210x USB to UART Bridge (COMX)**, en el ejemplo se muestra conectado en el COM61.

Los controladores son instalados automáticamente por lo que se recomienda verificar y esperar hasta que culmine la instalación.

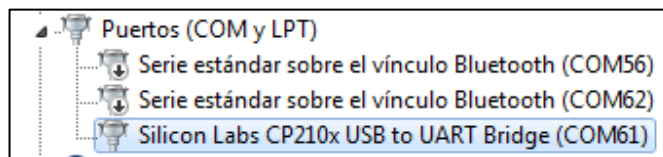


Figura 35: Conexión USB de PLC Delta.

El programa WPLsoft que típicamente es instalado en la dirección: inicio/todos los programas/Delta industria Automation/PLC/WPLSoft.

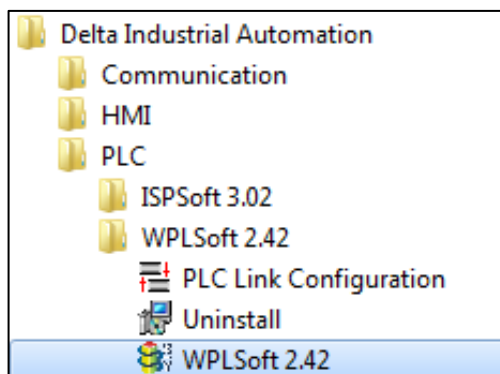


Figura 36: Ruta de ubicación de WPLSoft.

Al iniciar el programa se mostrará una ventana principal similar a:

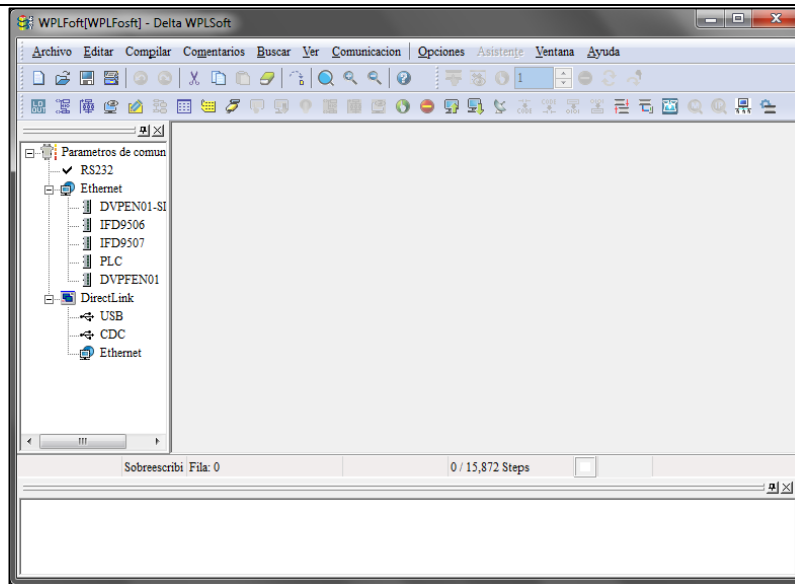


Figura 37: Ventana principal WPLsoft.

En el Menú Archivo dar clic en “Nuevo”, desplegándose una pantalla en donde se debe llenar los campos en blanco y la serie del PLC.

- Serie: DVP20SX211R.
- Puerto COM: asignado por la PC.
- Nombres de Programa y archivo

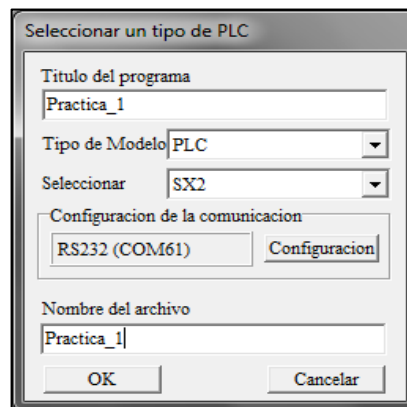


Figura 38: Seleccionar tipo de PLC.

Al dar clic en “**Configuración de la comunicación**”, aparecerá una segunda pantalla en donde se configura lo siguiente:

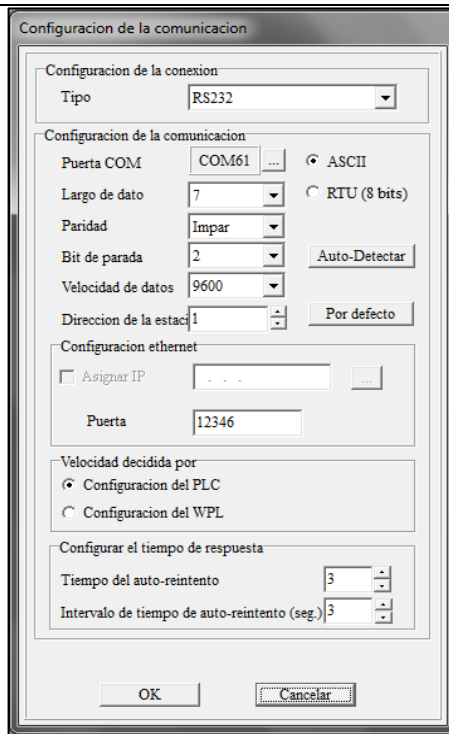


Figura 39: Configuración de comunicación.

Para confirmar los parámetros configurados damos clic en “OK”, finalmente se abrirá las ventanas de programación lo que significa que está listo para iniciar el código.

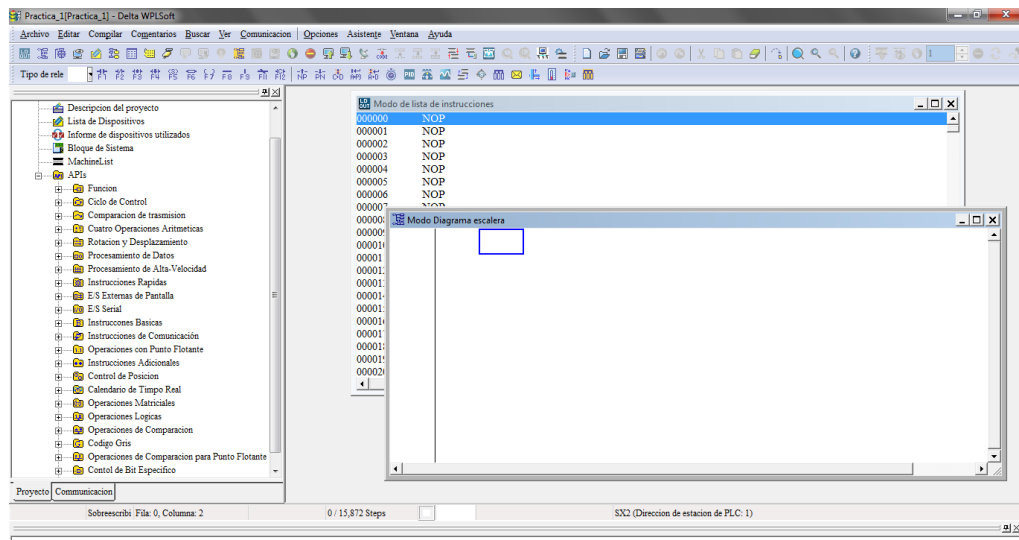


Figura 40: Configuración de comunicación.

Paso 5. Creación de código en WPLsoft.

El módulo inalámbrico RS-232 está conectado al puerto COM1 del PLC, por lo que debemos utilizar los parámetros para COM1:


Parámetros a utilizar:

- Número de puerto: COM1 (RS232)
- Velocidad de transmisión: 9600 bps
- Longitud de datos: 8 bits
- Paridad: Ninguna
- Longitud de bit de parada: 1 bit
- Registro de configuración: D1036
- Retener formato de configuración: M1128
- Modo ASCII: M1139 (ASCII=set, RTU=reset)

Puerto COM	RS-232 (COM1)	RS-485 (COM2)	RS-485 (COM3)	RS-485 (COM3)
Parámetro				
Velocidad de transmisión	110~115200 bps	110~921000 bps		110~115200 bps
Longitud de datos	7~8 bits			
Paridad	Verificación de paridad Par / Impar / Ninguna			
Longitud de bit de parada	1~2 bits			
Registro para configuración	D1036	D1120	D1109	
Retener formato de configuración	M1138	M1120	M1136	
Modo ASCII	Disponible para ambos maestro/esclavo			Disponible para esclavo
Modo RTU	Disponible para ambos maestro/esclavo			Disponible para esclavo
Selección de modo ASCII/RTU	M1139	M1143	M1320	
Dirección de comunicación de Esclavo	D1121		D1255	
Longitud de datos para acceso (ASCII)	100 registros			
Longitud de datos para acceso (RTU)	100 registros			

Figura 41: Parámetros puertos COM1 (Delta, 2011).

Para configurar los parámetros seleccionados en líneas de programación seguiremos los siguientes pasos:

- En la ventada principal del programa elegimos la pestaña  "Instrucción de aplicación"

- Iniciamos con la función que necesitamos para este caso será MOVER que se ubica en la instrucción de aplicación API: 12.

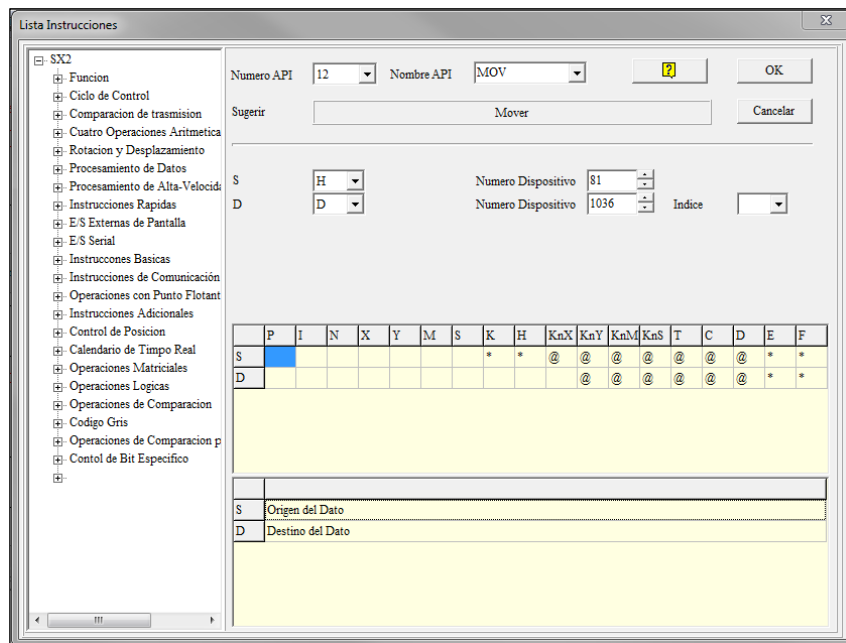



Figura 42: Declaración de función MOVER.

Como parámetros propios de MOVER tenemos dos variables:

- S: Origen del Dato
- D: Destino del Dato

La tecla  nos permite seleccionar todas las funciones a utilizar en el programa, a continuación, se declara los parámetros a configurar.

Código de programación.

Configuración y asignación de parámetros de comunicación.



Figura 43: Configuración de COM1.

Para el envío y recepción de datos se configura la función "RS" y se utiliza dos botoneras físicas agregando las marcas de dirección para el HMI.

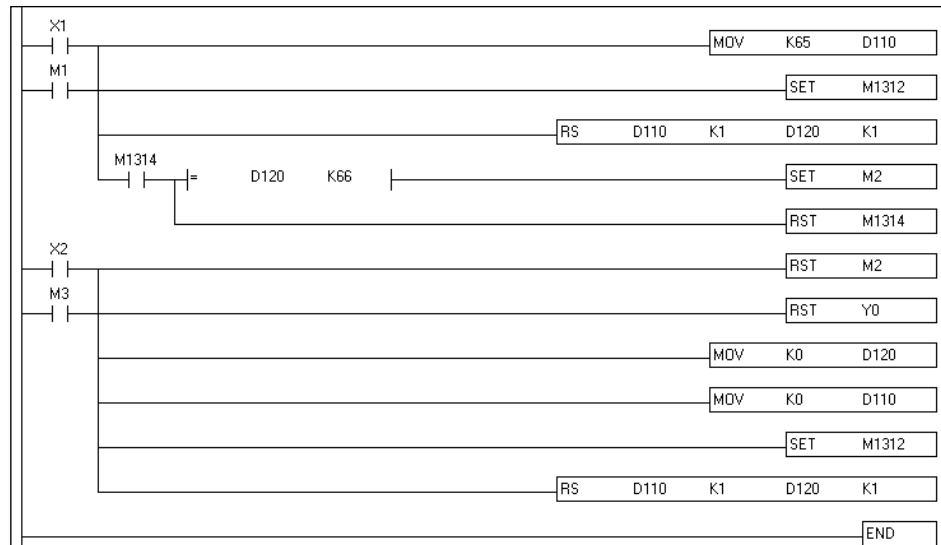




Figura 44: Configuración de comunicación serial.

Configuración de transferencia de PC a PLC.

Una vez terminada la programación se procede a realizar la transferencia al PLC en el siguiente orden:

- Compilar  y verificar que no existan errores de programación
- Cargar en el PLC .
- Esta transferencia es bidireccional ya que se puede seleccionar entre cargar programación al PLC(PC=>PLC) o descargar programación desde el PLC(PLC=>PC), para este caso será (PC=>PLC).
- Verificar que este marcado el cajón de "Programa" las demás son opciones que se utilizarán según su necesidad.
- Finalmente damos OK.

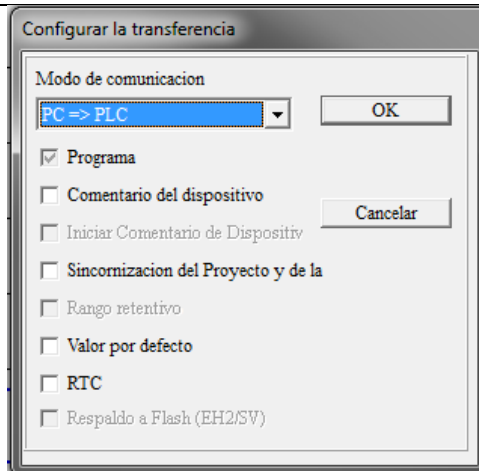


Figura 45: Configuración de transferencia a PLC.

Si Existe un programa ejecutándose en el PLC se mostrará una ventana de advertencia dar clic en OK y a continuación se cargará la nueva programación.

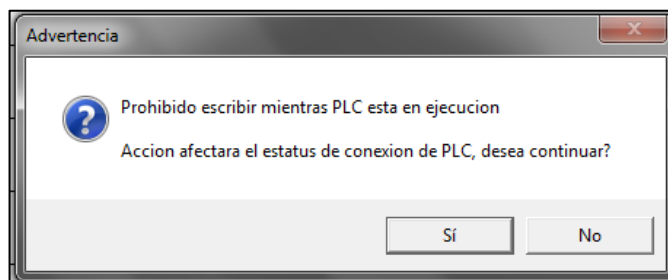


Figura 46: Advertencia de carga de programación en PLC.

Paso 6. Configuración de Programa ARDUINO para robot móvil.

Como se especificó en el paso anterior, se complementará el enlace de comunicación con un dato enviado desde el PLC al Arduino por lo que se debe recibir y responder con otro dato diferente mostrado en un indicador luminoso tanto en el robot móvil como en la maleta didáctica, por lo que se programa está bajo esas condiciones.

Antes de iniciar se recomienda:

- Conectar el robot móvil a través del puerto USB y utilizando el cable codificado como ARUSB01.
- Configurar el tipo y el puerto serial con que el PC asignó a la Arduino MEGA.

Al conectar el Arduino MEGA al PC, se podrá visualizar en la ventana “puertos COM y LPT” de la siguiente manera: **Arduino Mega 2560(COM53)**, como se muestra a continuación.

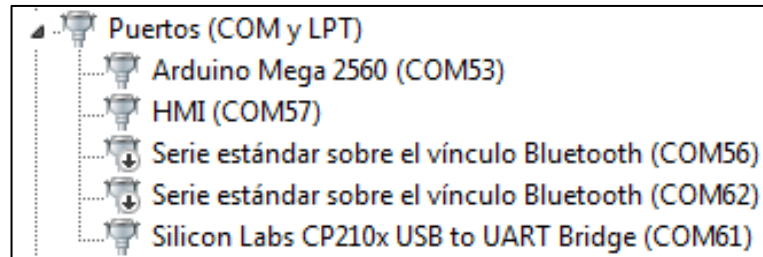


Figura 47: Propiedades de accionamientos.

Al abrir el programa se podrá visualizar una ventana similar a la siguiente:

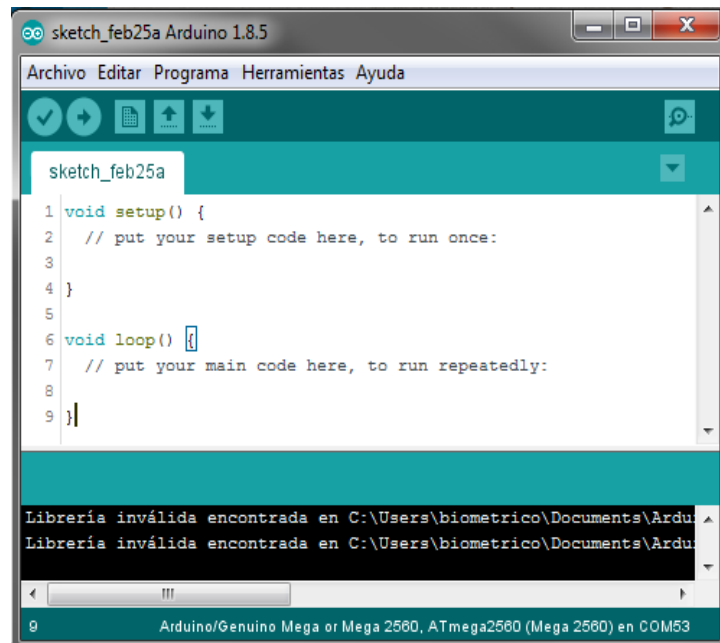


Figura 48: Ventana principal Arduino.

Para configurar el tipo de placa que se utiliza y el puerto serial al que está conectada debemos ir a la pestaña “Herramientas”.

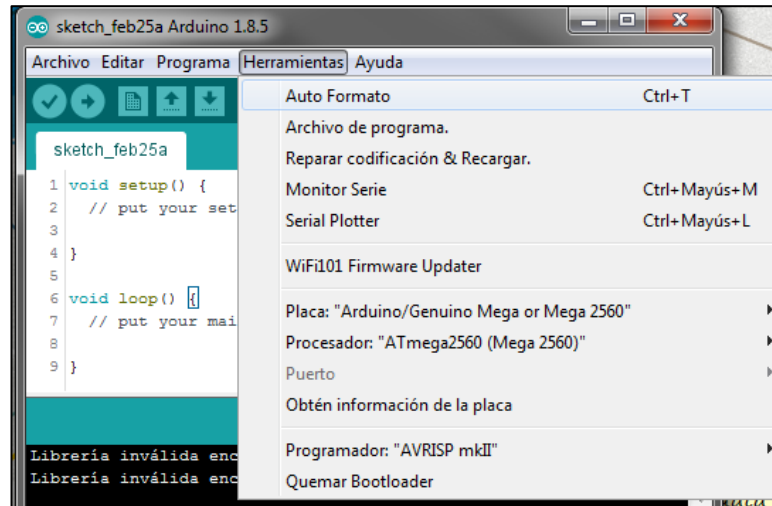


Figura 49: Configuración de Placa Arduino.

Luego de configurar se podrá verificar en la parte inferior de la ventana principal.



Figura 50: Datos configurados de Placa Arduino.

Paso7. Programación Arduino MEGA

A continuación, se muestra el programa capaz de realizar el enlace de comunicación propuesto.

```

1  /* Proyecto de Titulación DELTA-ARDUINO
2  * Autor: Angel Fabián Mansano Galarsa
3  * Fecha: 19/02/2018
4  * Numero de práctica = 1
5  */
6  // Declaracion de Variables
7
8  int datoPLC; // almacena todo lo enviado por el PLC
9  int LED=26; // Salida para verificar la comunicación entre PLC-Arduino MEGA
10
11 void setup() {
12 /*Declara el puerto serial a utilizar por defecto se configura como:
13 9600 bps, 8 bits de datos, No paridad y un bit de paro.*/
14 Serial1.begin(9600); //Serial PLC
15 pinMode(LED, OUTPUT); //Configura como salida digital
16 }
17 void loop() {
18 if (Serial1.available() > 0){
19 datoPLC = Serial1.read(); //Si el PLC envia datos el Arduino lee y almacena
20 if (datoPLC == 65){ //Cabecera de PLC; 65 = SET Comunicación
21 digitalWrite(LED, HIGH);
22 Serial1.write(66); //Responde a PLC
23 }
24 if (datoPLC == 0){ //0 = RESET Comunicación
25 digitalWrite(LED, LOW);
26 }
27 }
28 }

```

Figura 51: Programación Arduino.

Compilación y descarga en Arduino MEGA



Para poder verificar que no exista errores de programación presionar , si no existe errores dar click en , el programa empieza a cargar en la Arduino MEGA



Figura 52: Compilación de programa Arduino.

Una vez realizada la carga del programa en el Arduino MEGA, deberíamos poder comunicarnos con la maleta didáctica.

RESULTADO(S) OBTENIDO(S):

Tabla 4.7: Resultados práctica#1

	<ul style="list-style-type: none"> El módulo Firefly buscará y se enlazará automáticamente con el módulo instalado en el robot móvil, no significa que está habilitado el puerto serial
	<ul style="list-style-type: none"> Al presionar la botonera IO.1 inicia el enlace de comunicación y envía un dato al robot móvil y, En respuesta a esto el robot móvil debe enviarme un dato al PLC, debe encender un led local
	<ul style="list-style-type: none"> Con el dato enviado desde la maleta didáctica el robot móvil enciende la luz verde y posterior a esto envía un dato de respuesta al PLC



- De la misma manera podemos detener la comunicación con la botonera I0.2 quedando listo para la siguiente acción.




- Al momento de presionar I0.1 en la maleta didáctica, el led se apaga

RECOMENDACIONES

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica y posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Reconocer el uso de los puertos COM para el PLC y la Arduino MEGA
3. Elegir correctamente los parámetros de comunicación, tomando en cuenta que tanto el dispositivo transmisor con el receptor debe coincidir con los mismos parámetros.
4. Revisar el tipo de datos que envía el Robot móvil a la maleta didáctica debido.
5. Verificar en modo ONLINE el tipo de dato que está recibiendo tanto el PLC y el Arduino MEGA.

4.2. Práctica #2

		GUÍA DE PRÁCTICA DE LABORATORIO
CARRERA: Ingeniería Electrónica		ASIGNATURA: Automatización Industrial
NRO. PRÁCTICA:	2	TÍTULO PRÁCTICA: “Acondicionamiento de señales provenientes de los sensores ubicados en el robot móvil”
Objetivo: <ul style="list-style-type: none">Realizar un programa base en el Arduino MEGA para detección, empaquetado y envío de señales que generan todos los sensores en el robot móvil. Así mismo el direccionamiento de las mismas en el PLC Delta.		
Objetivos específicos: <ul style="list-style-type: none">Reconocer la función de los sensores según su ubicaciónInterpretar los tipos de señales que emiten los sensoresProgramar el Arduino MEGA para envío de datos al PLCProgramar el PLC para envío/recepción de datos de sensoresConfigurar la trama de datos a enviar a través del puerto serialVisualizar las señales de los sensores en los indicadores locales del HMI.		
INSTRUCCIONES:		<ul style="list-style-type: none">Disponer de un computador con programas ya instalados Arduino, WPLSoft y DOPSft.Disponer de un punto de alimentación 110VAC para la maleta didáctica.Verificar que la botonera de paro de emergencia no este accionada en la maleta didáctica y en el robot móvil.Visualizar los indicadores de encendido tanto en tarjeta del robot móvil como en maleta didáctica.

ACTIVIDADES POR DESARROLLAR

- **Paso1.** Reconocer e interpretar la función de los sensores según su ubicación
- **Paso2.** Programar el Arduino MEGA para envío de datos al PLC
- **Paso3.** Programar el PLC para envío/recepción de datos de sensores y configurar la trama de datos a enviar a través del puerto serial
- **Paso4.** Programar HMI para visualizar las señales de los sensores.

Paso 1. Reconocer e interpretar la función de los sensores según su ubicación

El robot móvil cuenta con varios sensores a bordo que se definen de la siguiente manera:

- **Sensores de obstáculo:** Detección de obstáculos lados frontal, lateral izquierda, y lateral derecha.
- **Sensores de piso:** Tira de sensores para detección de línea negra.
- **Sensor de carga:** Detección de carga localizada en el robot móvil

Cada tipo de sensor tienen sus propias limitaciones, sus capacidades y el tipo de salida, que se definen como:

Tabla 4.8: Definición de Sensores según ubicación

Función	Características principales del sensor	Tipo de señal de salida	Designación de PIN en Arduino MEGA
Sensores de obstáculo	“IR-LED” Diodo de emisión infrarroja, detector sensible a la posición con rango de señal efectiva 10 a 150CM	Analógica	<ul style="list-style-type: none"> • Frente: A3 • Izq: A4 • Der: A5
Sensores de piso	“IR-LED” Diodo de emisión infrarroja, fototransistor, con rango de señal efectiva 1 a 4MM	Analógica	<ul style="list-style-type: none"> • A10 • A9 • A8 • A15

			<ul style="list-style-type: none"> • A14 • A11 • A12 • A13
Sensor de carga	“IR-LED” Diodo de emisión infrarroja, fototransistor, con rango de señal efectiva 4 a 150CM ajustable con potenciómetro.	Digital	<ul style="list-style-type: none"> • 22

Paso 2. Programar el Arduino MEGA para envío de datos al PLC

Ya definido la función y el tipo de señal de todos los sensores, iniciamos el programa Arduino.

Para recibir los datos de los sensores en el Arduino se debe guardar en una variable, para luego enviar al PLC.

```

Prctica_2$
1 /* Proyecto de Titulación DELTA-ARDUINO
2 * Autor: Angel Fabián Manzano Galarza
3 * Fecha: 19/02/2018
4 * Numero de práctica = 2
5 */
6 // Declaracion de Variables
7 // Sensores Digitales
8 byte SC = 22; //Sensor de carga
9 byte LED = 26; //Led Verde
10 int s1; //Para guardar dato de sensorfrente A3
11 int s2; //Para guardar dato de sensorizquierda A4
12 int s3; //Para guardar dato de sensorderecha A5
13 int s4; //Para guardar dato de Piso1 A10
14 int s5; //Para guardar dato de Piso2 A9
15 int s6; //Para guardar dato de Piso3 A8
16 int s7; //Para guardar dato de Piso4 A15
17 int s8; //Para guardar dato de Piso5 A14
18 int s9; //Para guardar dato de Piso6 A11
19 int s10; //Para guardar dato de Piso7 A12
20 int s11; //Para guardar dato de Piso8 A13
21 int s12; //Para guardar dato de SC
22

```

Figura 53: Variables para señales de sensores en Arduino

La trama de datos a enviar a través del puerto serial tiene una longitud de 8 bits las cuales enviaremos en dos variables y recibiremos una variable del PLC tal como se especifica a continuación:

```

23 //Para enviar al PLC a travez del puerto serial se necesitan 2bytes
24 byte dato1; // Byte para enviar datos de los Sensores de (x3)Obstáculo, (x1)Carga
25 byte dato2; // Byte para enviar datos de los Sensores de (x8)Piso
26 //Byte de recepción de datos del PLC
27 byte datoPLC;

```

Figura 54: Variables Arduino para puerto serial

Se configura las señales digitales como entrada y/o salida y a la vez se inicializa el puerto serial 01 que le utilizamos para enviar y recibir datos hacia del PLC.

```

29 void setup() {
30 // Declara el puerto serial a utilizar por defecto se configura con 8 bits de datos,
31 // No paridad y un bit de paro, con velocidad en 9600bps.
32 Serial1.begin(9600); //Serial TTL-RS232 ARDUINO
33
34 pinMode(SC, INPUT);
35 pinMode(LED, OUTPUT);
36 }

```

Figura 55: Configurar como Entrada/salidas, habilitación de puerto serial

Se guarda la lectura de cada sensor en las variables antes declaradas:

```

36 void loop() {
37 //lectura de señales de sensores
38 s1 = analogRead(A3); //Sensor Frente
39 s2 = analogRead(A4); //Sensor izquierda
40 s3 = analogRead(A5); //Sensor derecha cables
41 s4 = analogRead(A10); //P1
42 s5 = analogRead(A9); //P2
43 s6 = analogRead(A8); //P3
44 s7 = analogRead(A15); //P4
45 s8 = analogRead(A14); //P5
46 s9 = analogRead(A11); //P6
47 s10 = analogRead(A12); //P7
48 s11 = analogRead(A13); //P8 cables
49 s12 = digitalRead(SC); //Sensor de carga

```

Figura 56: lectura de sensores

El método para enviar todas las señales de los sensores al PLC, es asignar un bit para cada sensor por lo que en la siguiente tabla se designa el orden y la cantidad de datos a utilizar:

Tabla 4.9: Definición de Sensores

Variable de envió(8 bits)	Bits de Datos	Designación de PIN en Arduino MEGA
Dato1	0	Frente: A3
	1	Izq: A4
	3	Der: A5
	4	22
Dato2	0	A10
	1	A9
	2	A8
	3	A15
	4	A14
	5	A11
	6	A12
	7	A13

Los bits de Dato1 se declaran de la siguiente manera

```
51 ///////////////////////////////////////////////////Acondicionamiento de señal-dato1////////////////////////////////////
52 if(s1>= 450) { bitSet(dato1, 0); }
53 else { bitClear(dato1, 0); }
54 ///////////////////////////////////////////////////
55 if(s2>= 500) { bitSet(dato1, 1); }
56 else { bitClear(dato1, 1); }
57 ///////////////////////////////////////////////////
58 if(s3>= 500) { bitSet(dato1, 2); }
59 else { bitClear(dato1, 2); }
60 ///////////////////////////////////////////////////
61 if(s12 == LOW) { bitSet(dato1, 3); }
62 else { bitClear(dato1, 3); }
```

Figura 57: Configuración de sensores por bit para Dato1

Los bits de Dato2 se declaran de la siguiente manera

```
66 ///////////////////////////////////////////////////
67 if(s5>= 36 && s5<=850) { bitSet(dato2, 1); }
68 else { bitClear(dato2, 1); }
69 ///////////////////////////////////////////////////
70 if(s6>= 36 && s6<=850) { bitSet(dato2, 2); }
71 else { bitClear(dato2, 2); }
72 ///////////////////////////////////////////////////
73 if(s7>= 36 && s7<=850) { bitSet(dato2, 3); }
74 else { bitClear(dato2, 3); }
75 ///////////////////////////////////////////////////
76 if(s8>= 36 && s8<=850) { bitSet(dato2, 4); }
77 else { bitClear(dato2, 4); }
78 ///////////////////////////////////////////////////
79 if(s9>= 36 && s9<=850) { bitSet(dato2, 5); }
80 else { bitClear(dato2, 5); }
81 ///////////////////////////////////////////////////
82 if(s10>= 36 && s10<=850) { bitSet(dato2, 6); }
83 else { bitClear(dato2, 6); }
84 ///////////////////////////////////////////////////
85 if(s11>= 36 && s11<=850) { bitSet(dato2, 7); }
86 else { bitClear(dato2, 7); }
87 ///////////////////////////////////////////////////envío de dato1 y dato2 a PLC////////////////////////////////////
88
```

Figura 58: Configuración de sensores por bit para Dato1

Una vez declarado y asignado las variables de los sensores queda todo listo para iniciar la transmisión de datos hacia el PLC.

```
87 ///////////////////////////////////////////////////envío de dato1 y dato2 a PLC////////////////////////////////////
88
89 if (Serial1.available() > 0)
90 {
91   datoPLC = Serial1.read();
92 }
93 if(datoPLC == 65) //Dato para inicio sensado
94 {
95   Serial1.write(66); //Cabecera para PLC
96   Serial1.write(dato1);
97   Serial1.write(dato2);
98   digitalWrite(LED, HIGH);
99 }
100 digitalWrite(LED, LOW);
101 }
```

Figura 59: Envío de datos al PLC

Paso 3. Programar el PLC para envió/recepción de datos de sensores y configurar la trama de datos a enviar a través del puerto serial

Se habilita el puerto serial con un muestreo de 1ms, y configurado para que envíe un dato y reciba tres a la vez.

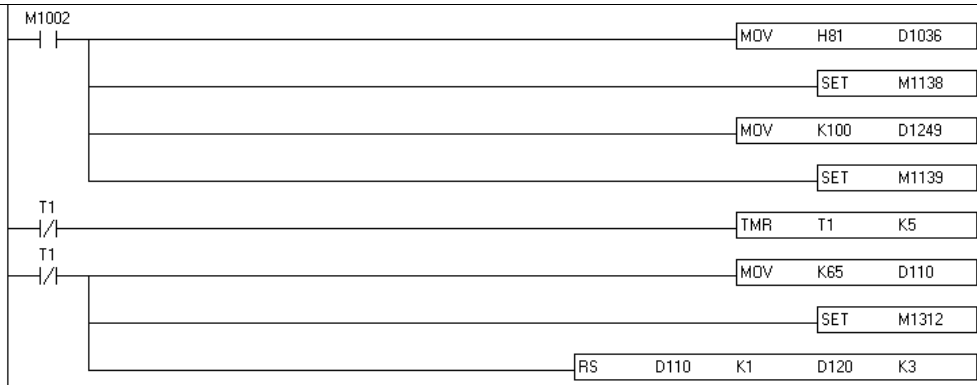


Figura 60: Configuración, envío y recepción de datos en PLC.

Si bien es cierto el Arduino MEGA está enviando 3 datos a la vez incluyendo la constante “66” que desde ahora será la cabecera para ordenar los datos y poder decodificar por bits.

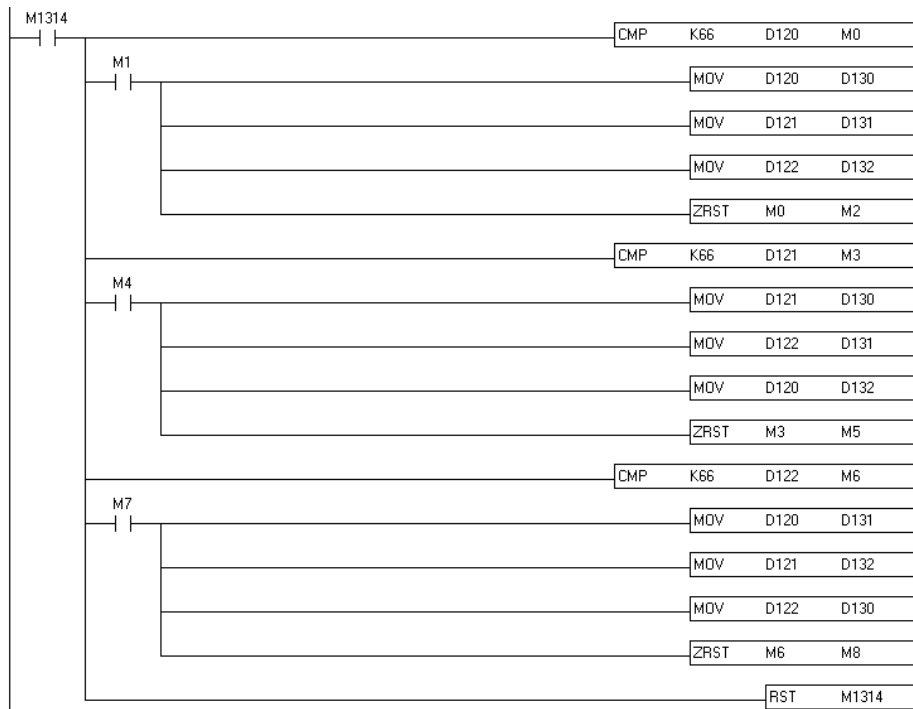


Figura 61: Arreglo de recepción de datos

El dato ya ordenado se guarda en las variables D130 a D132, siendo estas las variables a tratar. A partir de este punto podemos desfragmentar cada variable, tener en cuenta que:

$$\mathbf{D130 = 66; D131=Dato1; D132=Dato2}$$

Ya que Dato1 utiliza solo los 4 bits menos significativos podemos declarar solo 4btis en el PLC, tal como se muestra en la imagen siguiente:



Figura 62: Desfragmentación de Dato1 a bits.

Dato2 contiene los valores de la tira de sensores de piso, por lo que la desfragmentación en bits será de la siguiente manera:



Figura 63: Desfragmentación de Dato2 a bits.

Paso 4. Programar HMI para visualizar las señales de los sensores

Para inicializar el programa DOPsoft, vamos a la ruta de instalación por defecto creada por Windows.

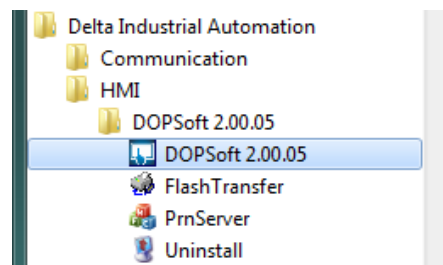



Figura 64: Ruta de ubicación de DOPsoft.

Crear un nuevo proyecto dando clic en , desplazándose una segunda pantalla en donde se podrá configurar la serie del HMI, Modelo, nombre de proyecto, etc.

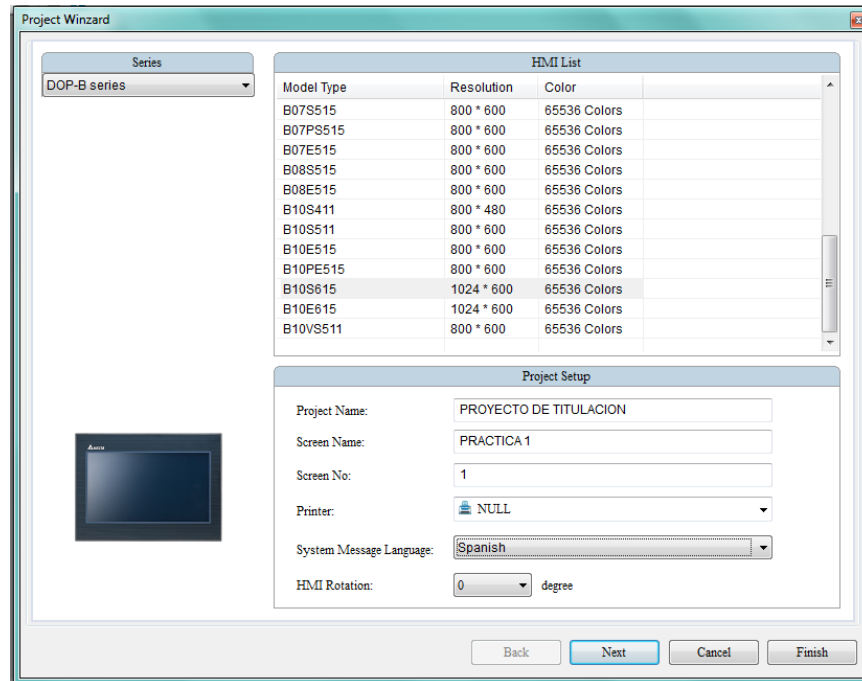


Figura 65: Configuración de HMI.

Luego de configurar estos parámetros damos clic en “next”, desplazándose el panel de configuración de comunicación donde debemos ingresar la marca y serie del PLC al que se va a conectar, también debemos ingresar los parámetros de comunicación tales como interfaz, trama de datos, etc. Lo que se puede visualizar en la siguiente imagen.

Para efecto de la práctica agregaremos un botón tipo momentáneo dando clic en



, en este icono podemos encontrar todo lo relacionado a un botón.

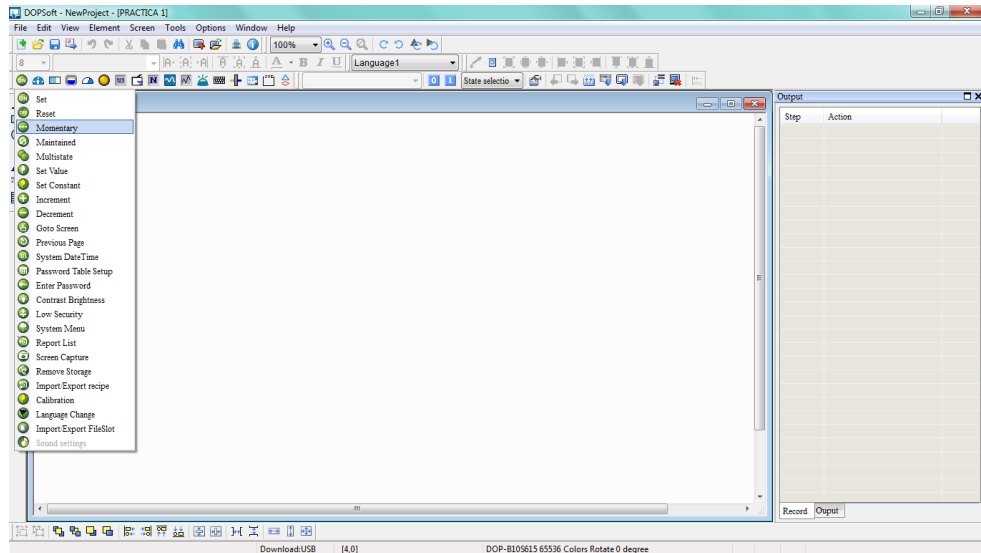


Figura 69: Barra de tipos de botones.

Luego de seleccionar el tipo de botón damos clic sobre el para insertar en la pantalla principal, posterior a eso dar doble clic sobre el nuevo botón creado con objetivo de direccionar y configurar toda la información para que este se enlace con el programa del PLC. Como el botón escribirá en el programa del PLC damos clic en “Write Address”, donde digitalizaremos para este caso que apunte en estado “0” a la marca “M16”.

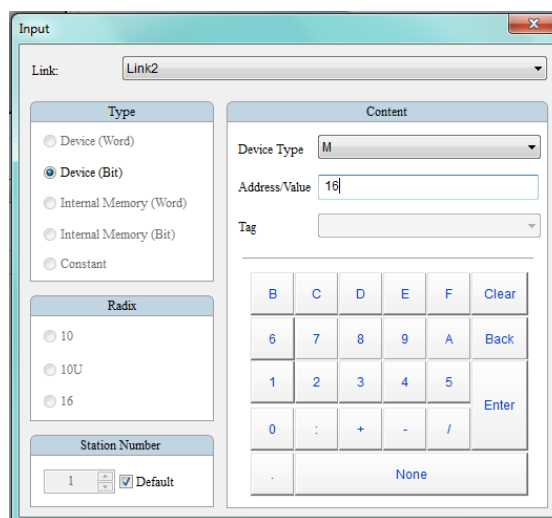


Figura 70: Direccionamiento de Entradas/Salidas en HMI.

Luego de configurar la Marca "M16". Damos clic en "Enter", guardándose de la siguiente manera:

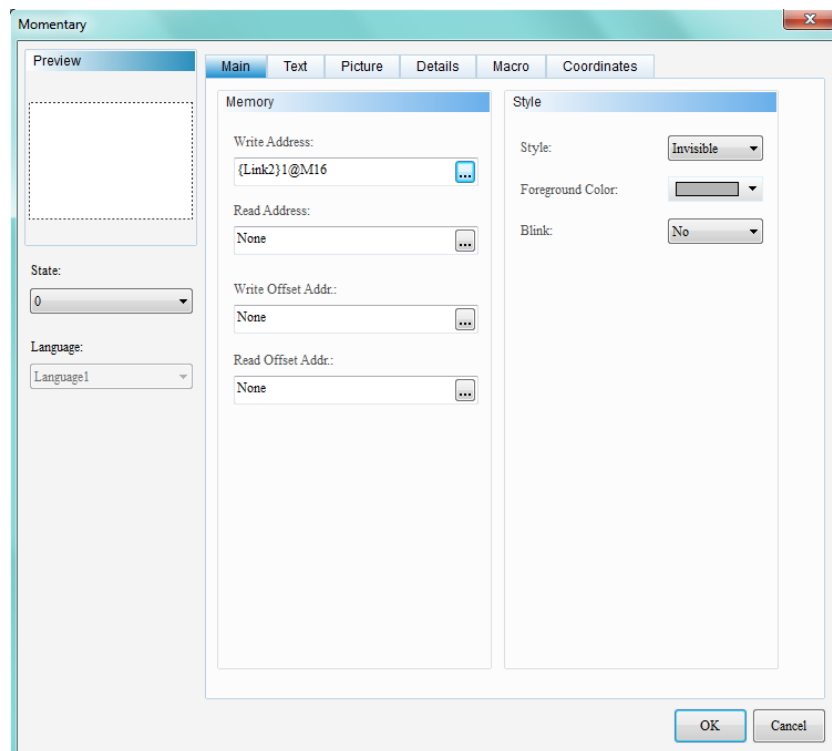


Figura 71: Direccionamiento de marca "M16" en HMI.

Si bien es cierto nosotros podemos agregar leyendas o imagen referentes a la función de cada botón, indicador o cualquier herramienta que estemos usando en el HMI por lo que a continuación configuraremos "M16" como una botonera circular de color verde y agregaremos la leyenda de función "ON".

Damos clic en "Picture", luego en "Picture Bank Name", donde se desplazará una lista de imágenes que podremos elegir conforme a nuestra necesidad para este caso como la botonera está en estado "0" será la imagen de nombre "BUTOON_01", y para el estado "1" será "BUTOON_02"

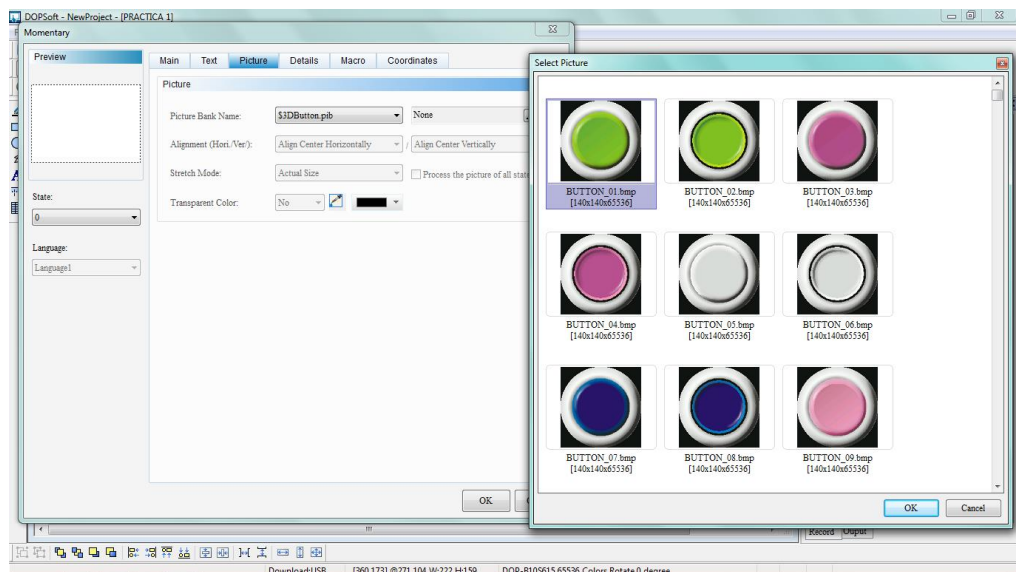


Figura 72: Selección de figura conforme a función de Marca "M16".

Conforme al detalle que se necesite podemos hacer alineaciones, escalamientos, fondos sin color, a fin de darle más animación a cada acción en el HMI.

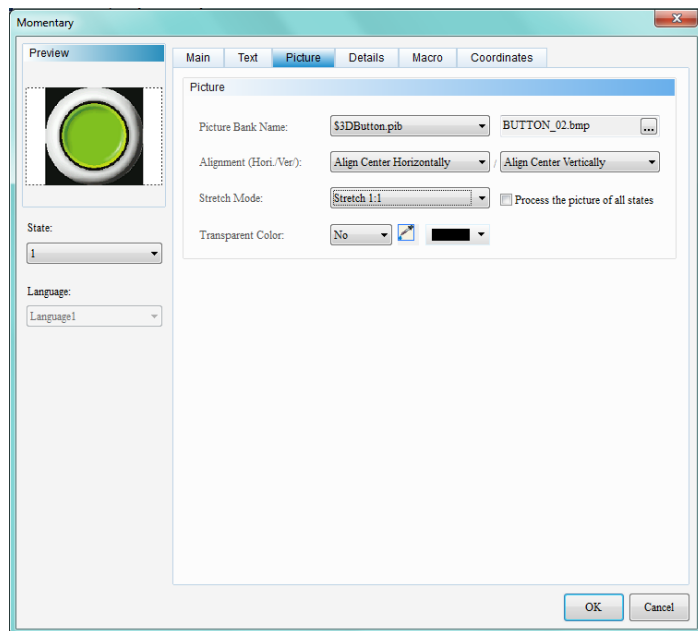


Figura 73: Alineación y escalamientos de "M16".

Para los siguientes indicadores/actuadores debemos realizar el mismo esquema agregando las luces indicatoras de estado on/off para todos los sensores, quedando como plantilla principal algo similar a lo que se muestra a continuación.

PRÁCTICA 2

Acondicionamiento de señales provenientes de los sensores ubicados en el robot móvil

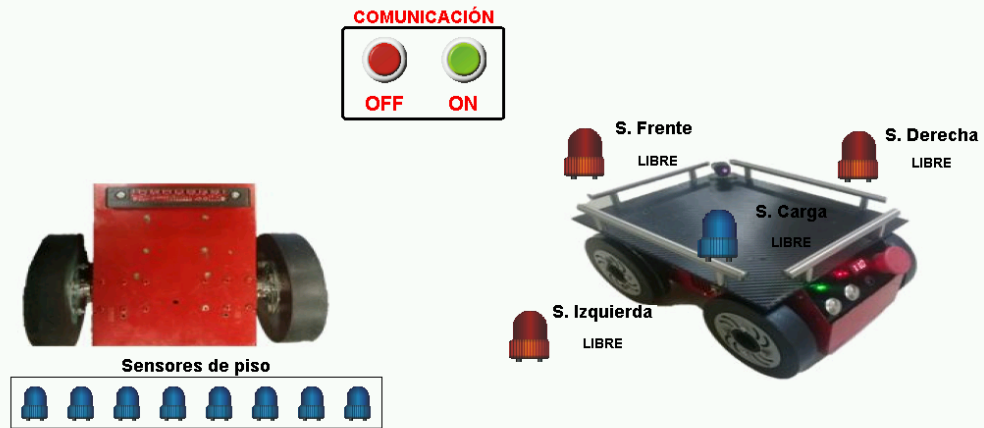


Figura 74: Plantilla Actuadores/indicadores.

RESULTADO(S) OBTENIDO(S):

Tabla 4.10: Resultados práctica#2

	<ul style="list-style-type: none"> • Para iniciar la recepción del estado de los sensores damos clic en el botón "ON".
	<ul style="list-style-type: none"> • El robot móvil deberá ubicarse en la línea negra de la pista encendiendo los indicadores conforme se mueva manualmente el robot




- Para visualizar el estado de los sensores de obstáculo, debemos ubicar algún objeto dentro de los límites de sensado.
- La práctica está diseñada para recibir todas las señales de los sensores al mismo tiempo.

RECOMENDACIONES:

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Verificar a que PIN del Arduino están todos los sensores a bordo antes de enviar a través del puerto serial debido a que podrá haber confusiones en cuanto a la configuración de envío de datos por bits.
3. Configurar correctamente el orden de recepción de datos en el PLC, tomando en cuenta que necesitamos de una cabecera.
4. Distribuir de manera ordenada el número de marcas utilizadas en el HMI, a fin de evitar funcionamientos indebidos.

4.3. Práctica #3

		GUÍA DE PRÁCTICA DE LABORATORIO	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Electiva	
NRO. PRÁCTICA:	3	TÍTULO PRÁCTICA: “Programación como seguidor de línea con un control PID.”	
Objetivo: <ul style="list-style-type: none">• Implementar un código en Arduino para el robot móvil capaz cumplir una trayectoria de línea negra utilizando un control PID.			
Objetivos específicos: <ul style="list-style-type: none">• Programar el Arduino MEGA para leer los sensores de Piso como señal de entrada del PID• Implementar el control PID para los motores de dirección			
INSTRUCCIONES:		<ul style="list-style-type: none">• Disponer de un computador con el programa ya instalado Arduino.• Verificar que la botonera de paro de emergencia no esté accionada en el robot móvil.• Visualizar los indicadores de encendido del robot móvil estén encendidos.	
ACTIVIDADES POR DESARROLLAR <ul style="list-style-type: none">• <u>Paso1.</u> Reconocer e interpretar la función de los sensores según su ubicación• <u>Paso2.</u> Programar el Arduino MEGA			

Paso 1. Reconocer e interpretar la función de los sensores según su ubicación

Se define parametrizar en una función de valores enteros que van de 0 a 7000, siendo 0 el valor mínimo cuando se detecte la línea negra por el costado izquierdo y 7000 el valor máximo cuando se detecte la línea negra por el costado derecho, contando que el valor central será 3000.

Tabla 4.11: Parametrización de sensores para PID

Numero de Sensor	S1	S2	S3	S4	S5	S6	S7	S8
PIN en Arduino MEGA	A10	A9	A8	A15	A14	A11	A12	A13
Valor Parametrizado	0	1000	2000	3000	4000	5000	6000	7000
Posición del Robot Móvil	Izq			Centro	Der			

Paso 2. Programar el Arduino MEGA

Conforme a la parametrización de los sensores se define variables propias para lectura en el Arduino, se detalla plenamente las condiciones y el tipo de variables. Se define las variables propias para el bloque de control PID, la variable donde se guarda los valores de los sensores, y el valor de salida del PID.

```
1
2 #include <QTRSensors.h>
3 #define PIN_PWM_QTR          11 // pin del PWM agregado al codigo.
4 #define NUM_SENSORS         8 // Numero de sensores de piso
5 #define NUM_SAMPLES_PER_SENSOR 4 // 4 lecturas por sensado
6 #define EMITTER_PIN         2 // Led emisor en PIN 2
7
8 /* Los sensores de piso se ubican de Izquierda a derecha
9 y se conectan a los pines analogicos 10, 9, 8, 15, 14, 11, 12, 13*/
10
11 QTRSensorsAnalog qtra((unsigned char[]) { 10, 9, 8, 15, 14, 11, 12, 13},
12 NUM_SENSORS, NUM_SAMPLES_PER_SENSOR, EMITTER_PIN);
13
14 int proportional, last_proportional, integral, derivative, Vmax, a, b, c;
15 unsigned int sensorValues[NUM_SENSORS];
16 double power_difference;
17
```

Figura 75: Declaración de variables para PID

En el bloque de configuraciones los valores del PID se declaran en cero y a la vez se inicializa el puerto serial2.

```
18 void setup() {
19   //valores iniciales para PID
20   proportional = 0;
21   last_proportional = 0;
22   integral = 0;
23   derivative = 0;
24   power_difference = 0;
25
26   delay(150); //tiempo para ubicar el Robot movil en la pista
27   /* Declara el puerto serial2 a utilizar,
28   * por defecto se configura con 8 bits de datos, no paridad y un bit de paro,
29   * con velocidad en 9600bps.*/
30   Serial2.begin(9600);
31 }
```

Figura 76: Bloque de configuraciones y puerto serial

En el bloque de calibración de sensores se realiza un muestreo por un periodo de tiempo a fin de capturar la mejor lectura con respecto a la superficie de trabajo, se enciende un indicador luminoso mientras esta en este bloque

```
32 //Enciende LED Verde mientras calibra los sensores de piso.
33 pinMode(26, OUTPUT);
34 digitalWrite(26, HIGH);
35 // Calibracion de sensores de piso
36 for (int i = 0; i < 200; i++) {
37   qtra.calibrate();
38 }
39 //Una vez termina la calibracion el LED se apaga
40 digitalWrite(26, LOW);
41 }
```

Figura 77: Bloque de configuraciones y calibración de sensores

En el bucle repetitivo se configura la velocidad máxima del robot móvil, que para este caso será $V_{max} = 30$.

Para sintonización del control PID la práctica está basada en el segundo método de **Ziegler-Nichols** que aplica para sistemas oscilantes de bucle cerrado.

Donde indica instalar un regulador de buque cerrado, inicialmente solo con el Proporcional (Integral y derivativo igualados a cero) con ganancia proporcional baja menor a 1, incrementando gradualmente hasta conseguir una oscilación constante (kp), al conseguir una oscilación constante medimos el periodo de oscilación (Tosc) utilizando un cronómetro.

Finalmente, la aproximación para todos los parámetros será:

Tabla 4.12: Parámetros PID según Ziegler-Nichols

Controlador	Kp	Ki=1/Ti	Kd/Td
P	0.5*kp	0	0
PI	0.45*kp	1.2/Tosc	0
PID	0.6*kp	2/Tosc	1.25*Tosc

La sintonización según el método aplicado indica que es necesario un:

kp=0.01, ki=0, kd=0.

```
45 Vmax = 30; //Valor de Velocidad Maxima establecida
46
47 double kp = 0.01;
48 int ki = 0;
49 int kd = 0;
50 unsigned int position = qtra.readLine(sensorValues);
51 ////////////////////////////////////////////////////
52 proportional = position - 3000; //position-centro de qtr // proportional=error
53 derivative = proportional - last_proportional;
54 integral += proportional;
55 last_proportional = proportional;
56 ////////////////////////////////////////////////////
57 power_difference = (proportional * kp) + (integral * ki) + (derivative * kd);
58 ////////////////////////////////////////////////////
```

Figura 78: Sintonización de control PID

La señal a tratada se guarda en la variable power_difference, la misma que debe parametrizarse a valores de 0 a 255, datos que el driver de los motores permite,

```
59 if (power_difference < -2) {
60   a = constrain(power_difference, -30, 0);
61   a = abs(a);
62   a = map(a, 0, 30, 158, 128);
63   Serial2.write(50);
64   Serial2.write(a);
65 }
66 else if (power_difference > 2) {
67   b = constrain(power_difference, 0, 40);
68   b = abs(b);
69   b = map(b, 0, 40, 30, 1);
70   Serial2.write(b);
71   Serial2.write(178);
72 }
73 else if ((power_difference <= 2) && (power_difference >= -2)) {
74   Serial2.write(40);
75   Serial2.write(168);
76 }
77
78 }
```

Compilado


El Sketch usa 6268 bytes (2%) del espacio de almacenamiento de programa. El máximo es 253952 byte
Las variables Globales usan 267 bytes (3%) de la memoria dinámica, dejando 7925 bytes para las va

62 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) en COM53

Figura 79: Direccionamiento de Motores con control PID

RESULTADO(S) OBTENIDO(S):


Tabla 4.13: Resultados práctica#3

	<ul style="list-style-type: none">• La tira de sensores a bordo en la imagen se define de izquierda a derecha los valores parametrizados. IMPORTANTE: La imagen está tomada desde la parte superior con el robot móvil invertido, considerar la proyección de espejo para la ubicación de los sensores.
	<ul style="list-style-type: none">• Por seguridad la botonera de “Paro de emergencia se mantiene accionada”• Al mover el interruptor de encendido luego de 150ms, el Robot Móvil iniciará el bucle de configuración y calibración de sensores culminando cuando el LED Verde de apague, a partir de esa acción podemos liberar la botonera “Paro de emergencia”
	<ul style="list-style-type: none">• Al momento que el LED Verde de apaga, solo depende de liberar la botonera “Paro de emergencia”, para iniciar su movimiento a través de la línea negra
	<ul style="list-style-type: none">• El robot cruza toda la trayectoria establecida sin problemas u oscilaciones gracias a la sintonización del control PID.

RECOMENDACIONES:

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Verificar todos los sensores a bordo antes de iniciar la calibración
3. Al momento de sintonizar el PID tomar en cuenta que tanto la superficie del robot móvil, la pista (Línea negra fondo blanco), y los niveles de voltaje de alimentación son variantes directas al ajustar a los parámetros.

4.4. Práctica #4

		GUÍA DE PRÁCTICA DE LABORATORIO
CARRERA: Ingeniería Electrónica		ASIGNATURA: CONTROL
NRO. PRÁCTICA:	4	TÍTULO PRÁCTICA: “Análisis de un control PID frente a un control On/Off (HISTÉRESIS).”
Objetivo: <ul style="list-style-type: none">Realizar la rutina de trayectoria de línea negra establecida en los Planos, y concluir que control es el esencial para esta aplicación.		
Objetivos específicos: <ul style="list-style-type: none">Crear un código en el Arduino MEGA para un robot móvil seguidor de línea utilizando un control por HISTÉRESIS.Implementar el código de seguidor de línea con control PID de la práctica #3.Analizar el control más efectivo para la trayectoria de seguidor de línea establecida en los planos.		
INSTRUCCIONES:		<ul style="list-style-type: none">Disponer de un computador con el programa ya instalado Arduino.Verificar que la botonera de paro de emergencia no este accionada en el robot móvil.Visualizar los indicadores de encendido del robot móvil estén encendidos.

ACTIVIDADES POR DESARROLLAR

- **Paso1.** Código en el Arduino MEGA para un robot móvil seguidor de línea utilizando un control por HISTÉRESIS.
- **Paso2.** Seguidor de línea con control PID.
- **Paso3.** Análisis de control más efectivo para la trayectoria de seguidor de línea establecida en los planos.

Paso 1. Código en el Arduino MEGA para un robot móvil seguidor de línea utilizando un control por HISTÉRESIS.

Para solución de la trayectoria del seguidor de línea mediante control por Histéresis primero se debe definir el valor de detección de la tira de sensores por lo que se parametriza en una función de valores enteros que van de 0 a 7000, siendo 0 el valor mínimo cuando se detecte la línea negra por el costado izquierdo y 7000 el valor máximo cuando se detecte la línea negra por el costado derecho, contando que el valor central será 3500.

Tabla 4.14: Parametrización tira de sensores

Numero de Sensor	S1	S2	S3	S4	S5	S6	S7	S8
PIN en Arduino MEGA	A10	A9	A8	A15	A14	A11	A12	A13
Valor Parametrizado	0	1000	2000	3000	4000	5000	6000	7000
Posición del Robot Móvil	Izq			Centro		Der		

A sí mismo el control de los motores se realiza utilizando datos seriales con una longitud de 8 bits divididos de la siguiente manera:

MOTOR 1

- **0** = Bloqueo de motores
- **1....63** = Dirección 1 Motor 1
- **64** = Stop Motor 1

- **65....127** = Dirección 2 Motor 1

MOTOR 2

- **128...191** = Dirección 1 Motor 2
- **192** = Stop Motor 2
- **193....255** = Dirección 2 Motor 2

Los datos se envían a través del puerto serial 2 del Arduino MEGA. Una vez claro las variables de entrada y salida, iniciamos el código de programación:

A fin de eliminar valores erróneos en la lectura se crea un bloque de calibración de sensores en cual delimita valores establecido de 0 a 7000.

```

1 #include <QTRSensors.h>
2 #define PIN_PWM_QTR 11 // pin del PWM agregado al código.
3 #define NUM_SENSORS 8 // Numero de sensores de piso
4 #define NUM_SAMPLES_PER_SENSOR 4 // 4 lecturas por sensado
5 #define EMITTER_PIN 2 // Led emisor en PIN 2
6
7
8 /* Los sensores de piso se ubican de Izquierda a derecha
9 y se conectan a los pines analogicos 10, 9, 8, 15, 14, 11, 12, 13*/
10 QTRSensorsAnalog qtra((unsigned char[]) {10, 9, 8, 15, 14, 11, 12, 13},
11 NUM_SENSORS, NUM_SAMPLES_PER_SENSOR, EMITTER_PIN);
12 unsigned int sensorValues[NUM_SENSORS];
13
14 void setup() {
15 //tiempo para ubicar el Robot movil en la pista
16 delay(100);
17 /* Declara el puerto a utilizar,
18 * por defecto se configura con 8 bits de datos, no paridad y un bit de paro,
19 con velocidad en 9600bps.*/
20 Serial.begin(9600); //Para toma de muestras
21 Serial2.begin(9600); //Para controlador de Motor
22 pinMode(26, OUTPUT); //Indicador luminoso-Inicio de calibracion
23 digitalWrite(26, HIGH); // turn on Arduino's LED to indicate we are in calibrat
24
25 for (int i = 0; i < 300; i++){
26 qtra.calibrate(); }
27
28 digitalWrite(26, LOW); //Indicador luminoso-Finaliza calibración
29 }

```

Figura 80: Variables de entrada Seguidor ON/OFF

En la función de “qtr()”, se analiza la adquisición de los sensores y se direcciona conforme al lado por el cual se desvía la línea negra; En el código los valores de sensor medidos se guardan en la función “posición”, y el bloque d código denominado “Dirección 1” define que el robot avance hacia el frente, “Dirección 2” el robot gira hacia la izquierda, y “Dirección 3” gira hacia la derecha.

```

32 void loop(){
33     qtr();
34 }
35
36 void qtr()
37
38     unsigned int position = qtra.readLine(sensorValues);
39     Serial.println(position);    // imprime datos de muestro
40
41     ///////////////Direccion 1/////////////////
42     if(position>=3000&&position<=4000){
43         Serial2.write(30);
44         Serial2.write(158);
45         Serial.println((int)30);
46         Serial.println((int)158);
47     }
48     ///////////////Direccion 2/////////////////
49     else if(position>=2999){
50         Serial2.write(20);
51         Serial2.write(192);
52         Serial.println((int)20);
53         Serial.println((int)192);
54     }
55     ///////////////Direccion 3/////////////////
56     else if (position<=4001){
57         Serial2.write(64);
58         Serial2.write(148);
59         Serial.println((int)64);
60         Serial.println((int)148);
61     }
62 }

```

Figura 81: Condiciones de giro seguidor HISTÉRISIS.

- **Paso2. Seguidor de línea con control PID.**

Seguir los pasos detallados en la Práctica #3, a fin de utilizar y analizar el código ya desarrollado.

Conforme a la parametrización de los sensores se define variables propias para lectura en el Arduino, se detalla plenamente las condiciones y el tipo de variables.

Se define las variables propias para el bloque de control PID, la variable donde se guarda los valores de los sensores, y el valor de salida del PID.

```

1
2 #include <QTRsensors.h>
3 #define PIN_PWM_QTR          11 // pin del PWM agregado al codigo.
4 #define NUM_SENSORS          8 // Numero de sensores de piso
5 #define NUM_SAMPLES_PER_SENSOR 4 // 4 lecturas por sensado
6 #define EMITTER_PIN          2 // Led emisor en PIN 2
7
8 /* Los sensores de piso se ubicican de Izquierda a derecha
9 y se conectan a los pines analogicos 10, 9, 8, 15, 14, 11, 12, 13*/
10
11 QTRsensorsAnalog qtra((unsigned char[]) { 10, 9, 8, 15, 14, 11, 12, 13},
12 NUM_SENSORS, NUM_SAMPLES_PER_SENSOR, EMITTER_PIN);
13
14 int proportional, last_proportional, integral, derivative, Vmax, a, b, c;
15 unsigned int sensorValues[NUM_SENSORS];
16 double power_difference;
17

```

Figura 82: Declaración de variables para PID

En el bloque de configuraciones los valores del PID se declaran en cero y a la vez se inicializa el puerto serial2.

```

18 void setup() {
19     //valores iniciales para PID
20     proportional = 0;
21     last_proportional = 0;
22     integral = 0;
23     derivative = 0;
24     power_difference = 0;
25
26     delay(150); //tiempo para ubicar el Robot movil en la pista
27     /* Declara el puerto serial2 a utilizar,
28     * por defecto se configura con 8 bits de datos, no paridad y un bit de paro,
29     * con velocidad en 9600bps.*/
30     Serial2.begin(9600);
31 }

```

Figura 83: Bloque de configuraciones y puerto serial

En el bloque de calibración de sensores se realiza un muestreo por un periodo de tiempo a fin de capturar la mejor lectura con respecto a la superficie de trabajo, se enciende un indicador luminoso mientras esta en este bloque.

```

32 //Enciende LED Verde mientras calibra los sensores de piso.
33 pinMode(26, OUTPUT);
34 digitalWrite(26, HIGH);
35 // Calibracion de sensores de piso
36 for (int i = 0; i < 200; i++) {
37     qtra.calibrate();
38 }
39 //Una vez termina la calibracion el LED se apaga
40 digitalWrite(26, LOW);
41 }

```

Figura 84: Bloque de configuraciones y calibración de sensores

En el bucle repetitivo se configura la velocidad máxima del robot móvil, que para este caso será $V_{max} = 30$;

```

45 Vmax = 30; //Valor de Velocidad Maxima establecida
46
47 double kp = 0.01;
48 int ki = 0;
49 int kd = 0;
50 unsigned int position = qtra.readLine(sensorValues);
51 ///////////////////////////////////////////////////
52 proportional = position - 3000; //position-centro de qtr // proportional=error
53 derivative = proportional - last_proportional;
54 integral += proportional;
55 last_proportional = proportional;
56 ///////////////////////////////////////////////////
57 power_difference = (proportional * kp) + (integral * ki) + (derivative * kd);
58 ///////////////////////////////////////////////////

```

Figura 85: Sintonización de control PID

La señal a tratar se guarda en la variable `power_difference`, la misma que debe parametrizarse a valores de 0 a 255.

```

63 if (power_difference < -2) {
64   a = constrain(power_difference, -30, 0);
65   a = abs(a);
66   a = map(a, 0, 30, 158, 128);
67   Serial2.write(50);
68   Serial2.write(a);
69 }
70 }
71 else if (power_difference > 2) {
72   b = constrain(power_difference, 0, 40);
73   b = abs(b);
74   b = map(b, 0, 40, 30, 1);
75   Serial2.write(b);
76   Serial2.write(178);
77 }
78 else if ((power_difference <= 2) && (power_difference >= -2)) {
79   Serial2.write(40);
80   Serial2.write(168);
81 }
82 //Adquisicion de datos de muestreo
83 Serial.println(position); //datos de sensores
84 Serial.println(power_difference); //datos de motores
85 }

```

Figura 86: Direccionamiento de Motores con control PID

Paso3. Análisis de control más efectivo para la trayectoria de seguidor de línea establecida en los planos.

Para análisis de efectividad de un control de HISTÉRESIS frente a un control PID se realiza un muestreo a bordo de señales de entrada (Sensores de piso), y las señales de control(motores), en una trayectoria de línea negra cerrada,

La pista de seguidor de línea se define de la siguiente manera:

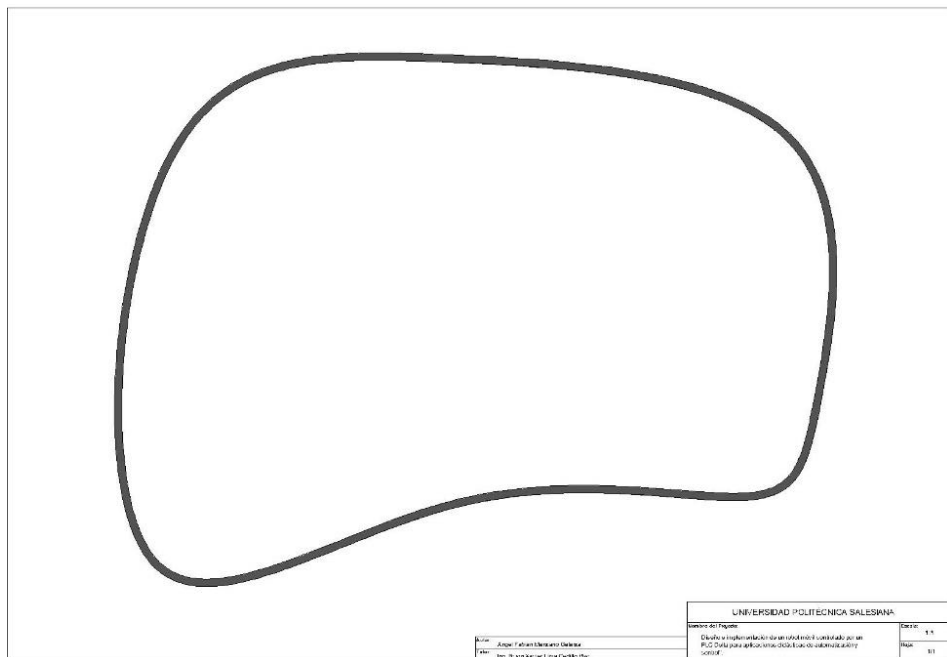


Figura 87: Pista para seguidor de línea

En base a la adquisición de datos generada en las líneas de programación se genera una gráfica de control para cada variable:

Muestreos para seguidor de Línea con Histéresis.

Datos para motor derecho, la potencia para avance del robot en tramos de línea recta es de un 30%, difiriendo en curvas que se eleva a 46% de su potencia total

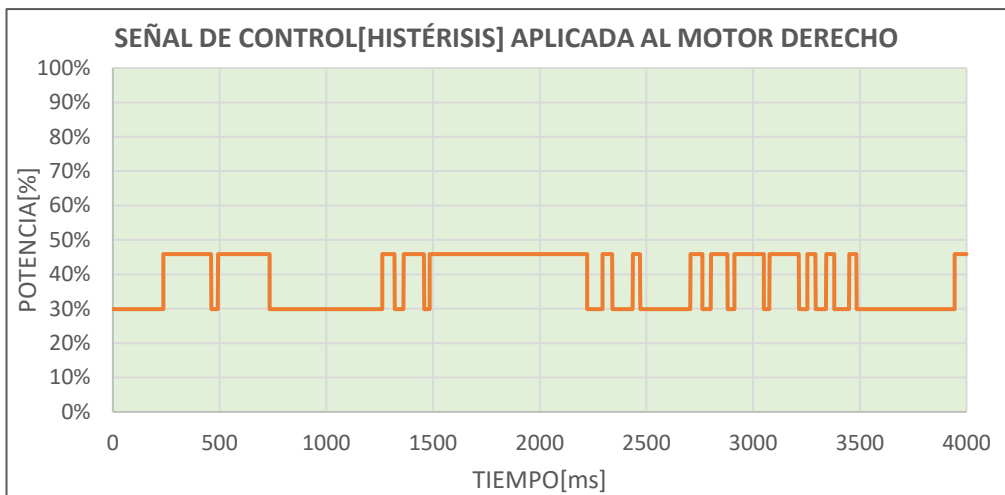


Figura 88: Datos de muestreo seguidor de línea HISTÉRISIS-Motor derecho.

Motor izquierdo, la potencia para avance del robot en tramos de línea recta es de un 10%, difiriendo en curvas que se eleva a 16% de su potencia total.

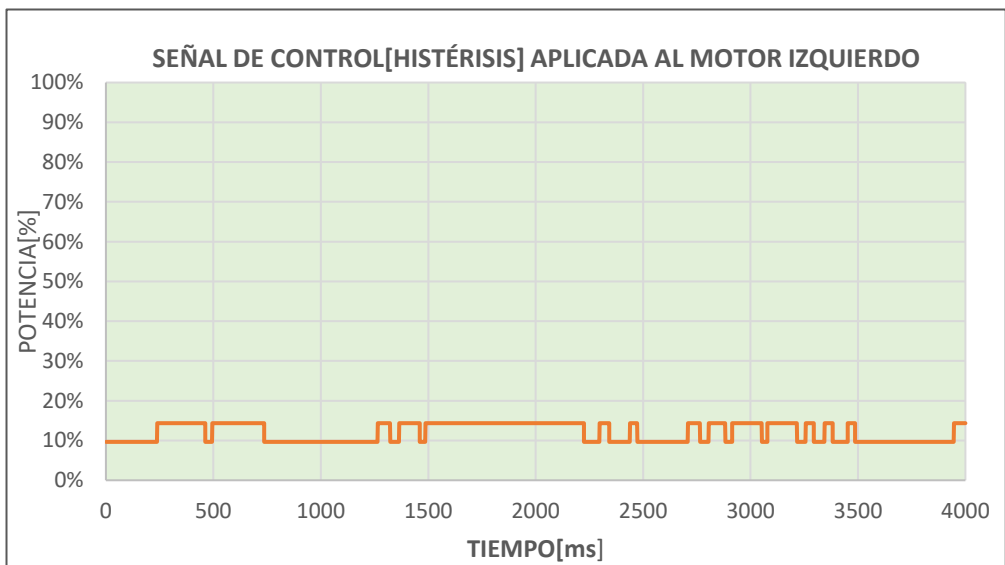


Figura 89: Datos de muestreo seguidor de línea HISTÉRISIS-Motor izquierdo.

Los datos de la tira de sensores según el bloque de programación se definen a un valor que de 0 a 7000.

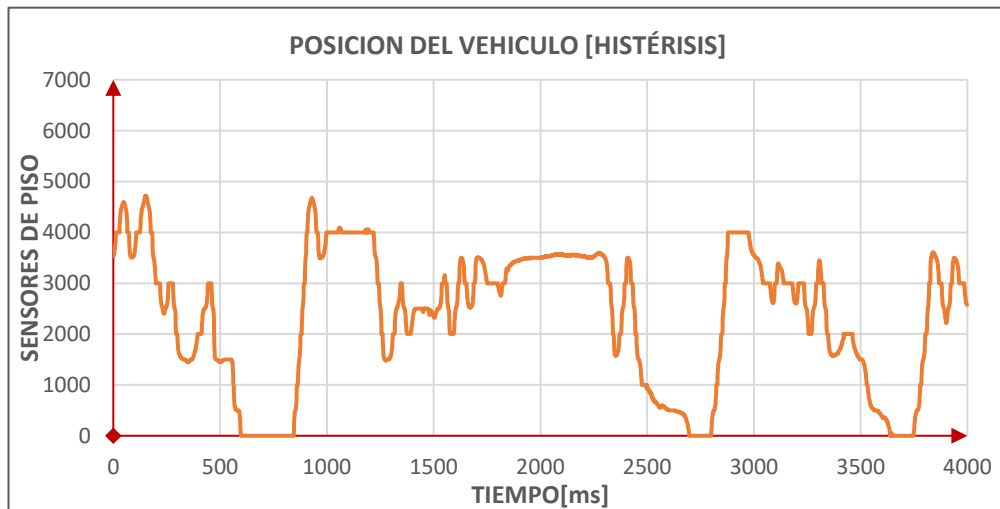


Figura 90: Datos de muestreo seguidor de línea HISTÉRISIS-Tira de sensores.

Muestras para seguidor de Línea con control PID.

Datos para motor derecho, la potencia para avance del robot en tramos de línea recta es de un 30%, difiriendo en curvas que se eleva a 60% de su potencia total.

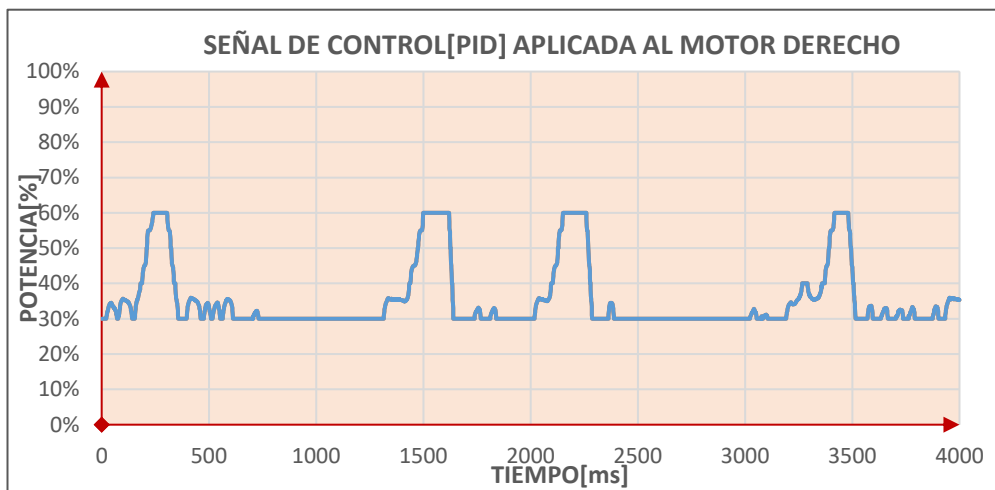


Figura 91: Datos de muestreo seguidor de línea PID - Motor derecho.

Motor izquierdo, la potencia para avance del robot en tramos de línea recta es de un 30%, difiriendo en curvas que se eleva a 46% de su potencia total.

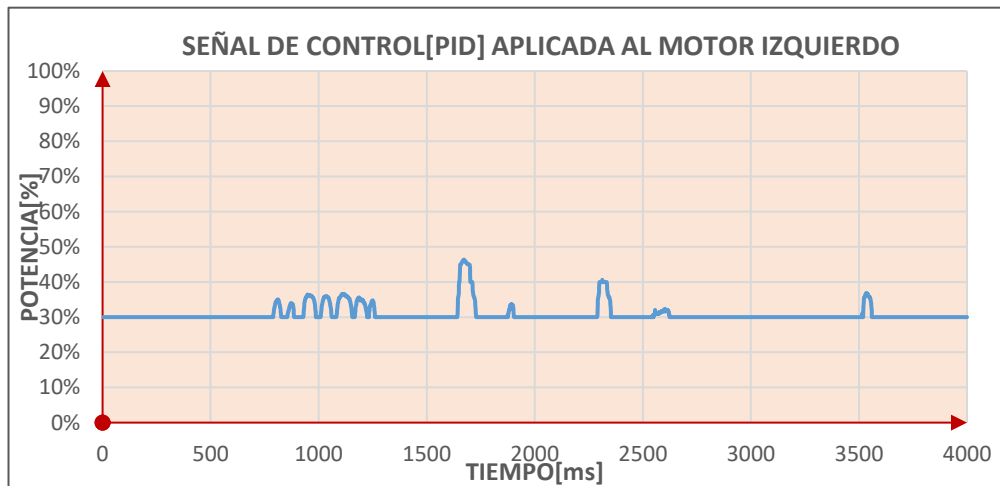


Figura 92: Datos de muestreo seguidor de línea PID-Motor izquierdo.

Los datos de la tira de sensores según el bloque de programación se definen a un valor que de 0 a 7000.

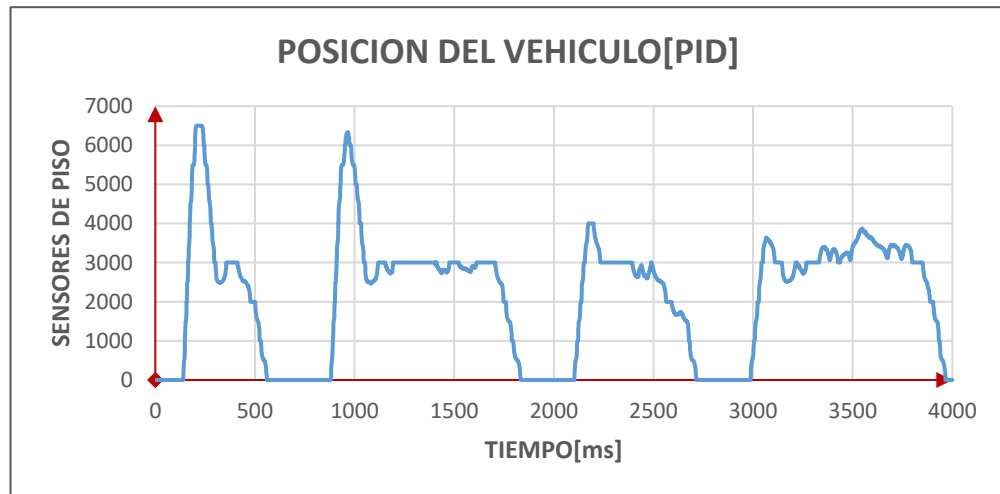


Figura 93: Datos de muestreo seguidor de línea PID-Tira de sensores.

RESULTADO(S) OBTENIDO(S):

Se analiza el comportamiento de las variables monitoreada, visualizando en una sola grafica lo siguiente:

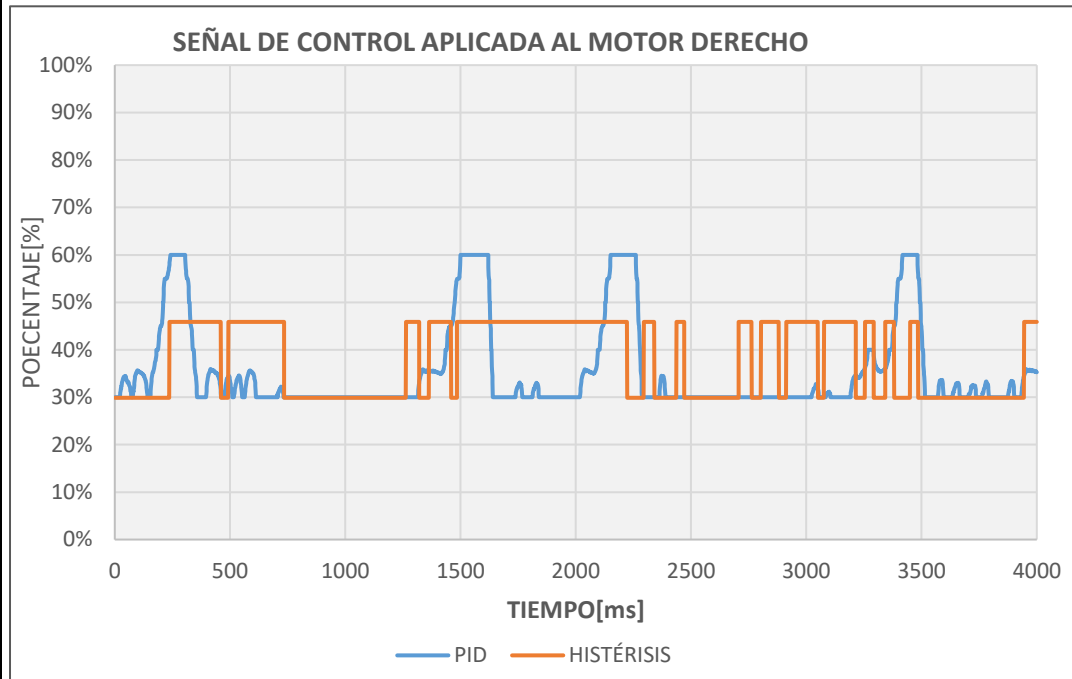


Figura 94: Analisis HISTÉRISIS-PID Motor derecho.

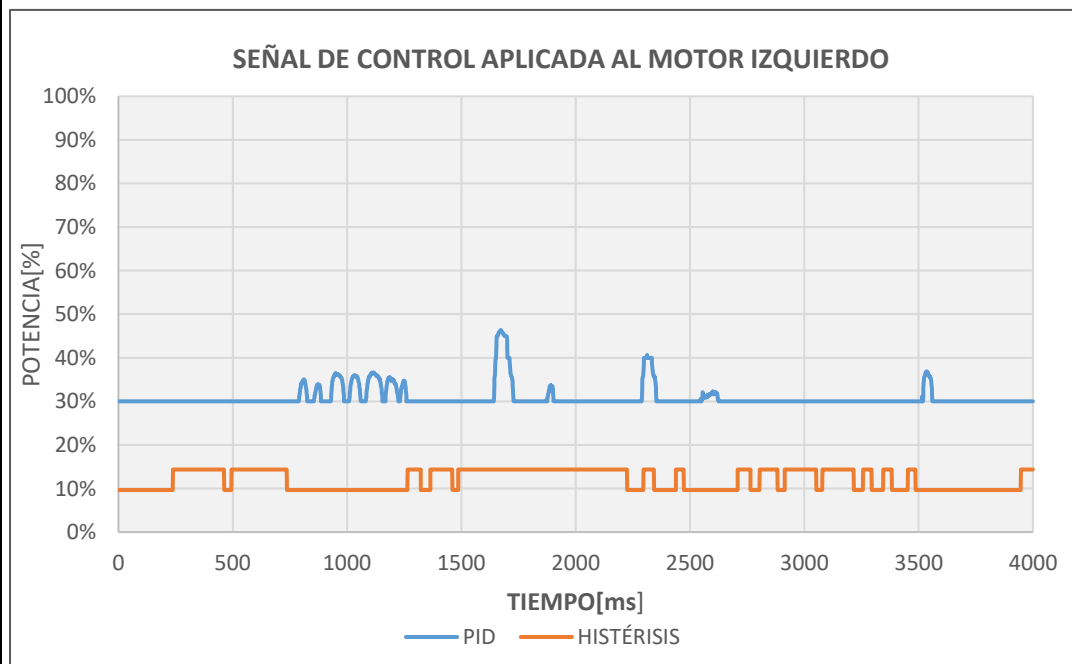


Figura 95: Analisis HISTÉRISIS-PID Motor izquierdo.

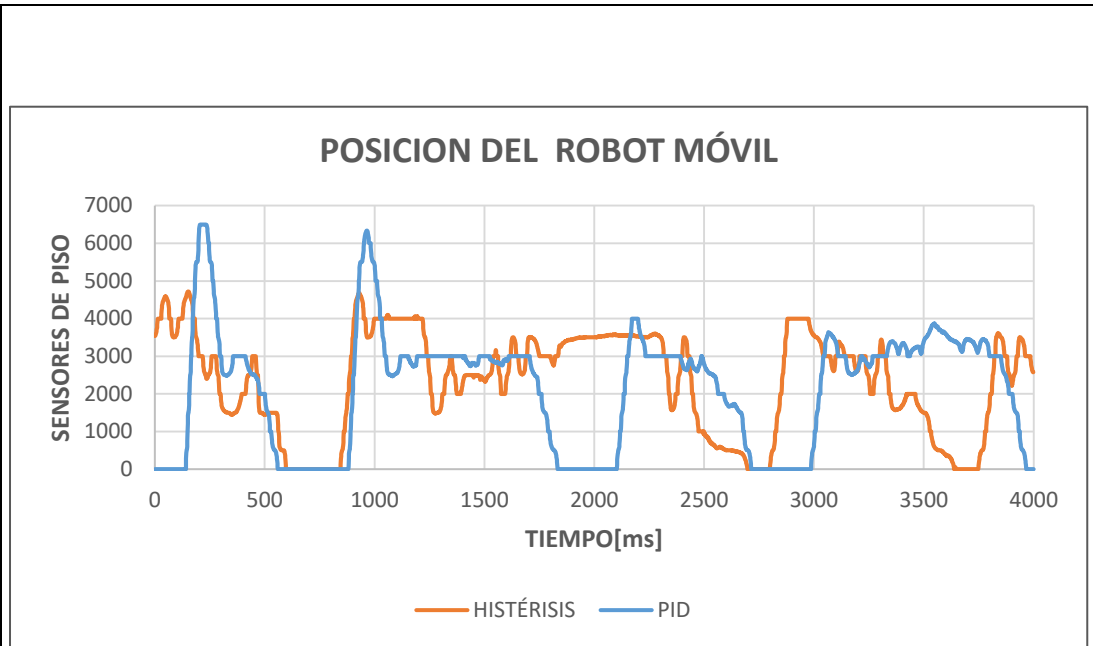



Figura 96: Analisis HISTÉRISIS-PID Motor izquierdo.

RECOMENDACIONES:

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Verificar todos los sensores a bordo antes de iniciar la calibración.
3. Al momento de sintonizar el PID tomar en cuenta que tanto la superficie del robot móvil, la pista (Línea negra fondo blanco), y los niveles de voltaje de alimentación son variantes directas al ajustar a los parámetros.

4.5. Práctica #5

		GUÍA DE PRÁCTICA DE LABORATORIO	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Automatización Industrial	
NRO. PRÁCTICA:	5	TÍTULO PRÁCTICA: Control en modo manual del robot móvil	
Objetivo: <ul style="list-style-type: none">• Controlar manualmente el robot móvil desde la maleta didáctica.			
Objetivos específicos: <ul style="list-style-type: none">• Reconocer los protocolos de comunicación.• Asignar puertos de comunicación en PLC, HMI y Arduino MEGA.• Elaborar un programa en el PLC para envío de datos al Arduino MEGA.• Elaborar un programa en Arduino para recepción de señales en la maleta didáctica y envío de datos al controlador de motores.• Configurar el HMI para enlazar con la programación del PLC.			
INSTRUCCIONES:		<ul style="list-style-type: none">• Disponer de un computador con programas ya instalados Aduino, WPLSoft, DOPsoft.• Disponer de un punto de alimentación 110VAC para la maleta didáctica.• Verificar que la botonera de paro de emergencia no este accionada en la maleta y el robot móvil.• Deslizar a “ON” el switch de encendido general ubicado en la maleta didáctica y robot móvil.• Visualizar los indicadores de encendido tanto en tarjeta del robot móvil como en maleta didáctica.	

ACTIVIDADES A DESARROLLAR

- **Paso1.** Reconocimiento de topología de comunicación
- **Paso2.** Configuración puertos de comunicación en PLC, HMI y Arduino MEGA.
- **Paso3.** Creación de código en WPLsoft.
- **Paso4.** Creación de código en Arduino.
- **Paso5.** Creación de pantallas en DOPsoft
- **Paso6.** Pruebas de enlace entre PLC y Robot móvil.

Paso 1. Reconocimiento de topología de comunicación.

En la topología de comunicaciones definida para la maleta didáctica y el robot móvil se utiliza cuatro tipos de comunicación identificados como:

- RS-232
- RS-485
- Serial TTL
- USB

Los mismos que se configuran de la siguiente manera:

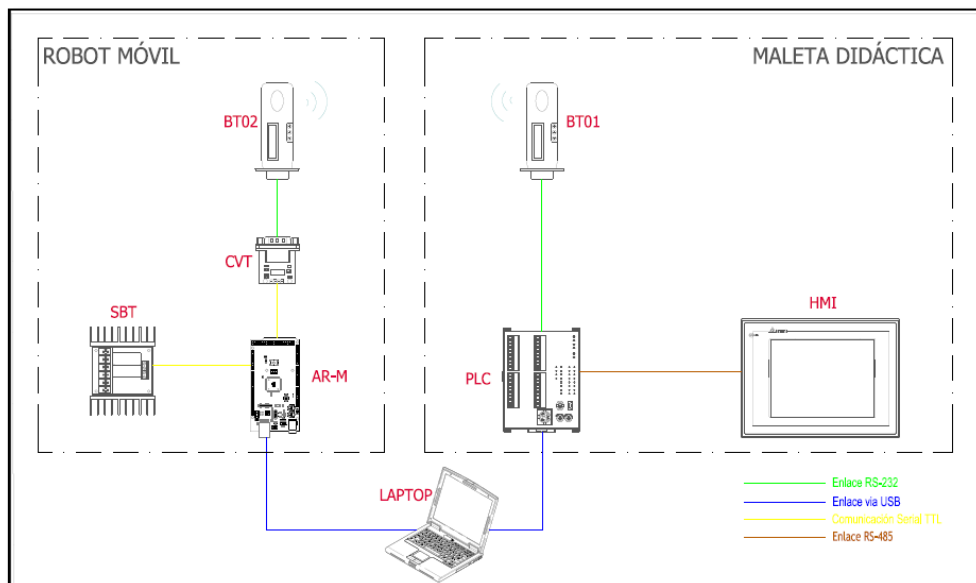


Figura 97: Topología de comunicación

Nota: La topología mostrada, solo se muestra los tipos de enlaces de comunicación utilizados; fuentes y tipos de alimentación no se describen. La configuración de la topología no es modificable para ninguna de las prácticas aquí descritas.

Paso 2. Configuración puertos de comunicación en PLC, HMI y Arduino MEGA.

Según la topología de comunicación los puertos del **PLC** se definen de la siguiente manera:

Tabla 4.15: Definición de Puertos PLC

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	Comunicación con Robot Móvil
COM2	RS-485	Comunicación con HMI
COM3	Mini USB	Programación de PLC

El **HMI** define sus puertos de comunicación como:

Tabla 4.16: Definición de Puertos HMI

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	NO conectado
COM2	RS-232/ RS-422/ RS-485	Comunicación RS-485(PLC)
COM3	RS-232/ RS-422/ RS-485	NO conectado
USB host	USB tipo A	Alimentación 5VDC, Firefly
USB esclavo	USB tipo B	Programación de HMI

La Arduino **MEGA** configura los puertos de comunicación:

Tabla 4.17: Definición de Puertos Arduino MEGA

NOMBRE DEL PUERTO	(x)PIN Arduino MEGA		USO
Serial0	(0)TX	(1)RX	Programación MEGA
Serial1	(18)TX	(19)RX	Comunicación Módulo Firefly RS-232/Bluetooth(PLC)
Serial2	(16)TX	(17)RX	Conexión Sabertooth
Serial3	(14)TX	(15)RX	NO conectado

Paso 3. Creación de código en WPLsoft.

Los parámetros de comunicación se configuran como:

COM1: RS-232, Velocidad:9600, 8 bits de datos, NO paridad, 1 bits de parada

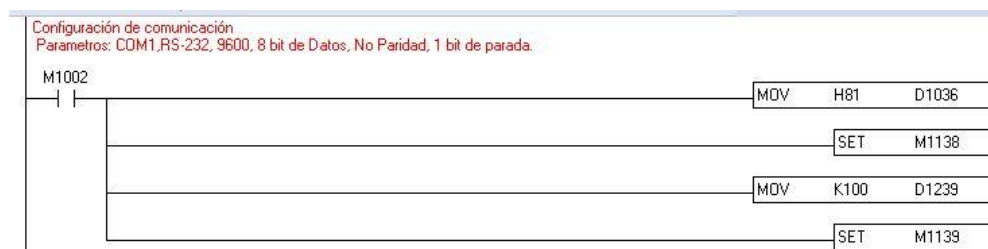


Figura 98: Configuración de comunicación serial en PLC

La programación del PLC para configurar el envío de datos, la marca M5 es la asignada de habilitar la comunicación entre la maleta didáctica y el robot móvil.

Para el control de dirección del Robot móvil se utiliza combinaciones de datos conforme la dirección deseada, mismos que están comandados por:

- M1= Adelante.
- M2= Derecha.
- M3= Atrás.
- M4= Izquierda

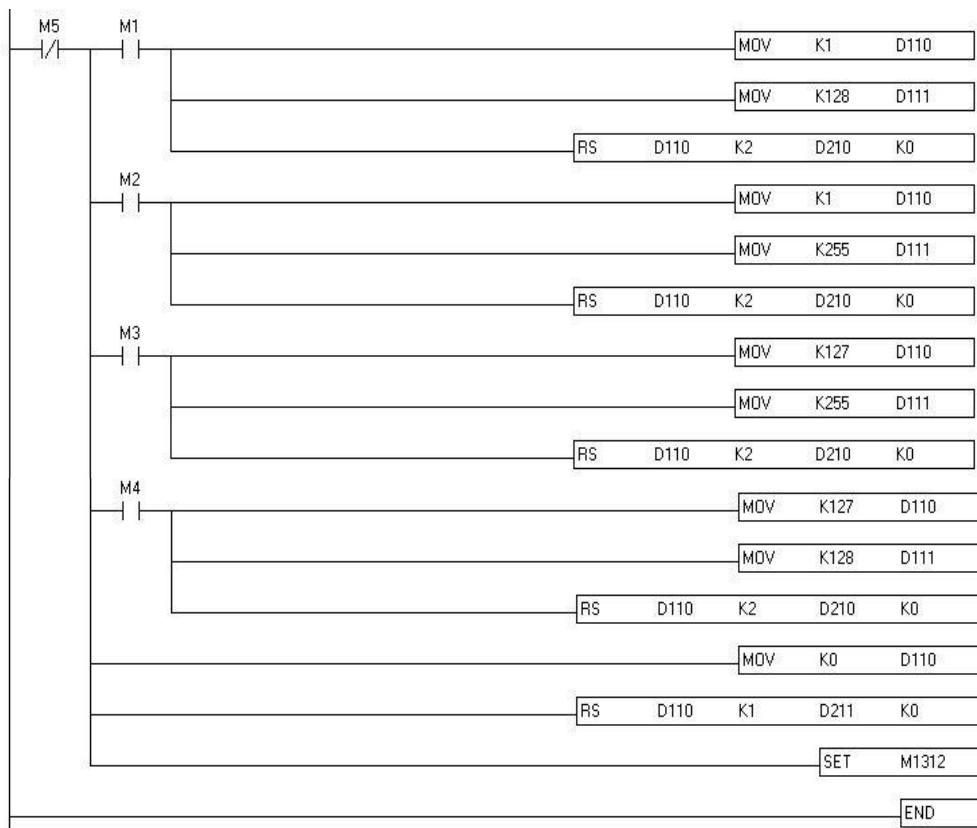


Figura 99: Programación control de robot móvil

Se envía constantemente "K0" para mantener en posición cero del robot móvil.

Paso 4. Creación de código en Arduino.

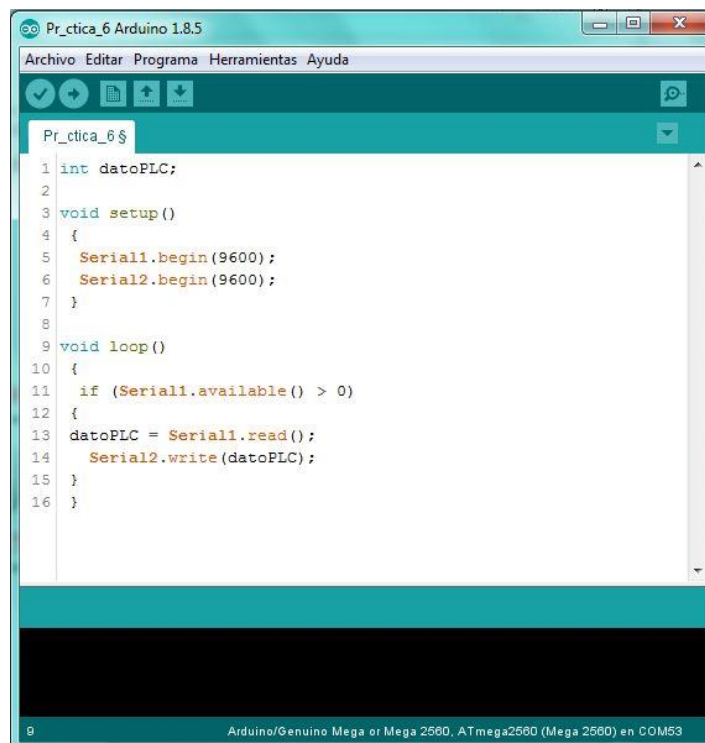
Para comandar los motores de dirección desde el PLC, el Arduino necesita interpretar los datos recibidos y a la misma vez debe de enviar dichos datos al controlador de motores, sin ninguna parametrización.

En el programa configuramos una variable entera de nombre "DatoPLC" para guardar los datos enviados desde la maleta didáctica.

Lo puertos seriales se distribuyen de la siguiente manera:

- Para recepción de datos de PLC: serial1
- Para envío de datos al controlador de motores: serial2

El Arduino almacena momentáneamente el valor enviado por el PLC para enviar al controlador de motores. Ver Fig.(abajo)



```
Prctica_6 Arduino 1.8.5
Archivo Editar Programa Herramientas Ayuda
Prctica_6 $
1 int datoPLC;
2
3 void setup()
4 {
5   Serial1.begin(9600);
6   Serial2.begin(9600);
7 }
8
9 void loop()
10 {
11   if (Serial1.available() > 0)
12   {
13     datoPLC = Serial1.read();
14     Serial2.write(datoPLC);
15   }
16 }
```

Figura 100: Programación control de robot móvil

Paso 5. Creación de pantallas en DOPsoft

Para inicializar el programa DOPsoft, vamos a la ruta de instalación por defecto creada por Windows.

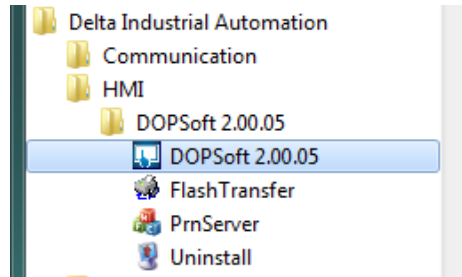



Figura 101: Ruta de ubicación de DOPsoft.

Crear un nuevo proyecto dando clic en , desplazándose una segunda pantalla en donde se podrá configurar la serie del HMI, Modelo, nombre de proyecto, etc.

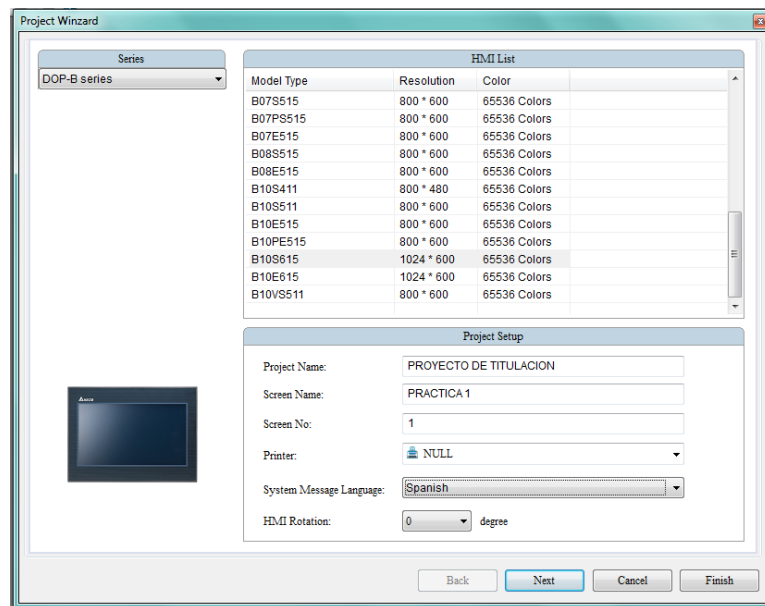


Figura 102: Configuración de HMI.

Luego de configurar estos parámetros damos clic en “next”, desplazándose el panel de configuración de comunicación donde debemos ingresar la marca y serie del PLC al que se va a conectar, también debemos ingresar los parámetros

de comunicación tales como interfaz, trama de datos, etc. Lo que se puede visualizar en la siguiente imagen.

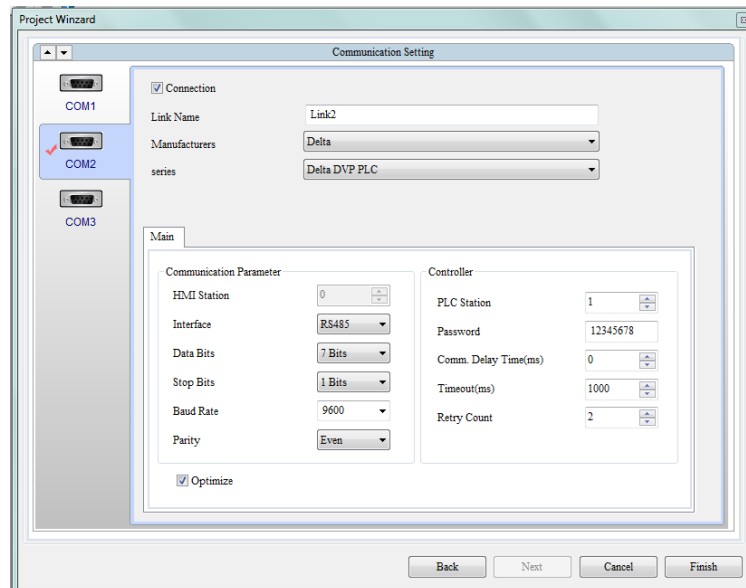


Figura 103: Configuración de comunicación HMI.

Luego de configurar los parámetros necesarios para la comunicación HMI-PLC, está listo para programar a la necesidad de la práctica, al dar clic en “finish” se desplaza una pantalla inicial del programa tal como se visualiza a continuación:

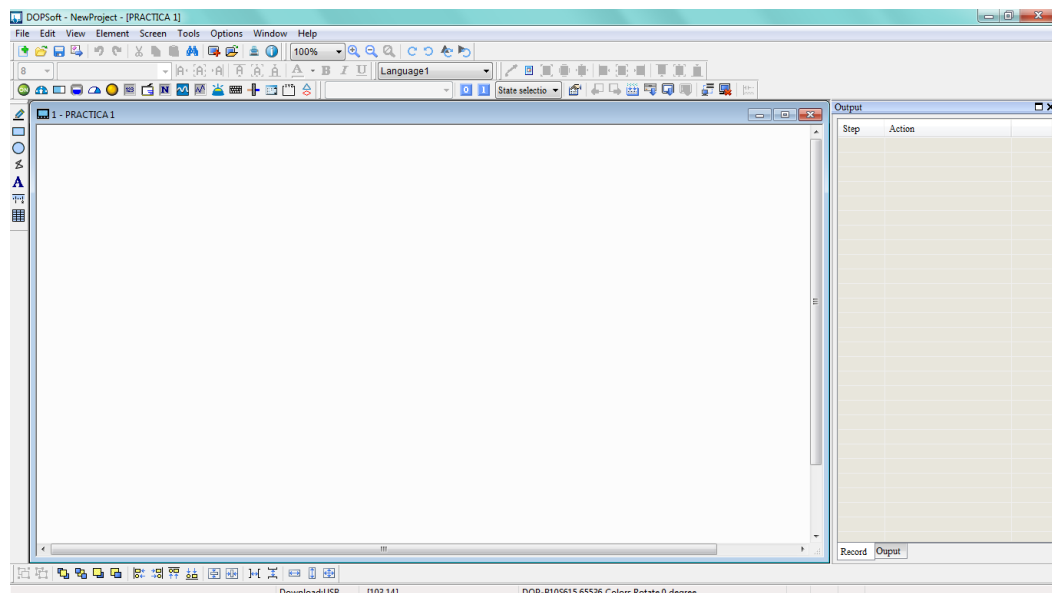


Figura 104: Pantalla inicial DOPsoft.

Para insertar elementos tales como botoneras indicadores medidores, etc. Debemos ir a la barra que se muestra a continuación:



Figura 105: Barra de elementos.

Para efecto de la práctica agregaremos un botón tipo momentáneo dando clic en



, en este icono podemos encontrar todo lo relacionado a un botón.

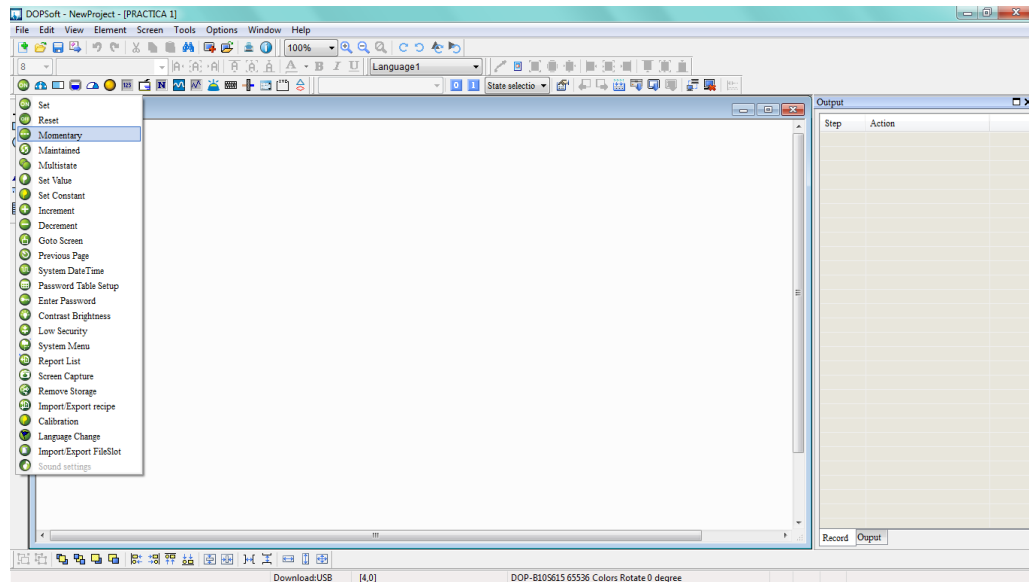


Figura 106: Barra de tipos de botones.

Luego de seleccionar el tipo de botón damos clic sobre el para insertar en la pantalla principal, posterior a eso dar doble clic sobre el nuevo botón creado con objetivo de direccionar y configurar toda la información para que este se enlace con el programa del PLC. Para esta práctica se configuran los botones de set, reset, derecha, izquierda, adelante.

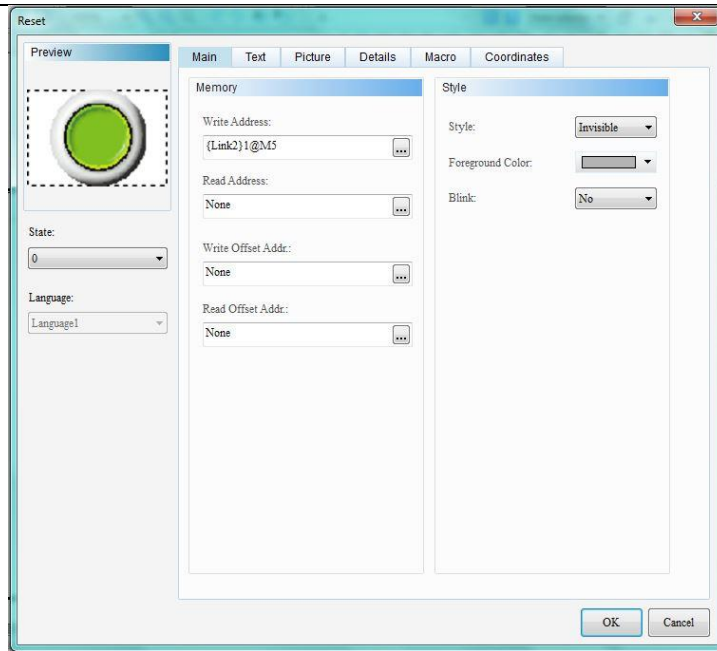


Figura 107: Configuración botón set/reset

La configuración para de los botones que dan dirección al robot desde la pantalla HMI, se enlazaran a las variables M1 hasta M4 esto permitirá él envió de datos desde el PLC hacia el robot móvil.

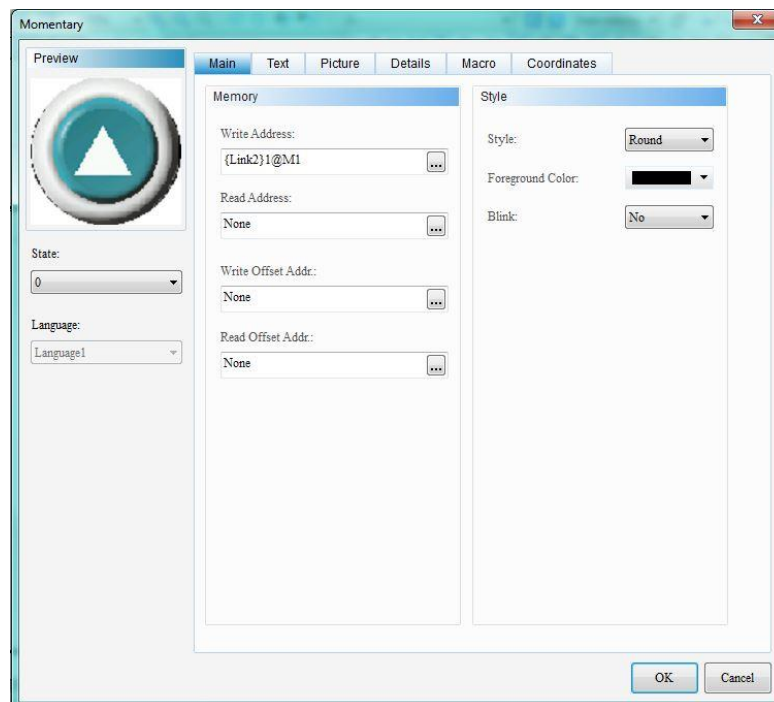
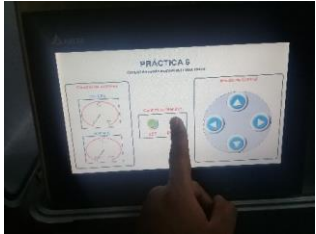
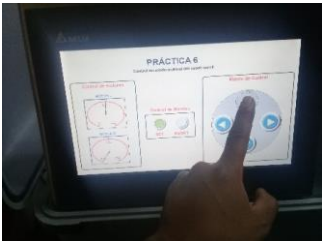

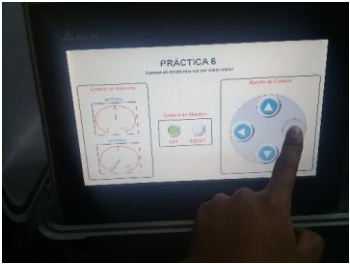


Figura 108: Configuración de botones para dirección del robot móvil

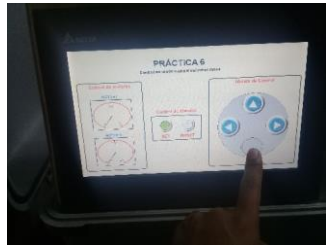
RESULTADO(S) OBTENIDO(S):

Tabla 4.18: Resultados práctica#5

	<ul style="list-style-type: none">• Control de movimiento del robot móvil, debemos presionar ya sea Set para habilitar o reset para deshabilitar movimientos.
	<ul style="list-style-type: none">• Al presionar el botón que se muestra en la figura en respuesta a eso el Robot móvil se moverá hacia adelante.
	<ul style="list-style-type: none">• Robot en movimiento hacia adelante
	<ul style="list-style-type: none">• Al presionar el botón que se muestra en la figura en respuesta a eso el Robot móvil se moverá hacia la derecha.



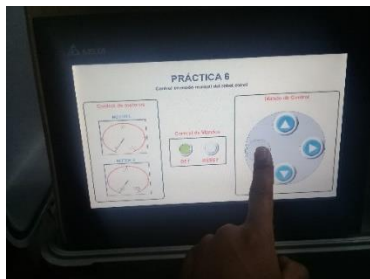
- Robot en movimiento hacia la derecha



- Al presionar el botón que se muestra en la figura en respuesta a eso el Robot móvil se moverá hacia atrás.



- Robot en movimiento hacia la atrás.



- Al presionar el botón que se muestra en la figura en respuesta a eso el Robot móvil se moverá hacia la izquierda.




- Robot en movimiento hacia la izquierda.

RECOMENDACIONES:

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Realizar pruebas de movimiento solo en superficie plana, debido q que si se trabaja en otro escenario podría causar algún daño en la tira de sensores de piso.

4.6. Práctica #6

		GUÍA DE PRÁCTICA DE LABORATORIO	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Automatización Industrial	
NRO. PRÁCTICA:	6	TÍTULO PRÁCTICA: Elaborar una Interfaz gráfica capaz de comandar y visualizar el robot móvil	
<p>Objetivo:</p> <ul style="list-style-type: none"> Realizar la rutina de un seguidor de línea negra y el transporte de carga de la posición 1 a la posición 2 utilizando el HMI como mando on/off para el robot móvil. <p>Objetivos específicos:</p> <ul style="list-style-type: none"> Reconocer los protocolos de comunicación. Asignar puertos de comunicación en PLC, HMI y Arduino MEGA. Interpretar los tipos de señales que emiten los sensores Elaborar un programa en el software WPLsoft, que sea capaz de enlazar el PLC Delta con una tarjeta Arduino MEGA. Elaborar un programa en el software Arduino, que sea capaz de enlazar el Arduino MEGA con el PLC Delta, y que cumpla con la rutina de seguidor de línea. Elaborar un programa en el software DOPsoft, que sea capaz de enlazar el HMI con el PLC Delta, para poner en marcha o parar el Robot remotamente. 			
INSTRUCCIONES:		<ul style="list-style-type: none"> Disponer de un computador con programas ya instalados Arduino, WPLsoft, DOPsoft. Disponer de un punto de alimentación 110VAC para la maleta didáctica. Verificar que la botonera de paro de emergencia no esté accionada en la maleta y el robot móvil. 	

- Deslizar a “ON” el switch de encendido general ubicado en la maleta didáctica y el Robot móvil.
- Visualizar los indicadores de encendido tanto en tarjeta del robot móvil como en maleta didáctica.

ACTIVIDADES A DESARROLLAR

- **Paso1.** Reconocimiento de topología de comunicación
- **Paso2.** Asignación puertos de comunicación en PLC, HMI y Arduino MEGA
- **Paso3.** Creación de código en WPLsoft.
- **Paso4.** Creación de código en Arduino.
- **Paso5.** Creación de pantalla para HMI en DOPsoft.

Paso 1. Reconocimiento de topología de comunicación.

En la topología de comunicaciones definida para la maleta didáctica y el robot móvil se utiliza cuatro tipos de comunicación identificados como:

- RS-232
- RS485
- Serial TTL
- USB

Los mismos que se configuran de la siguiente manera:

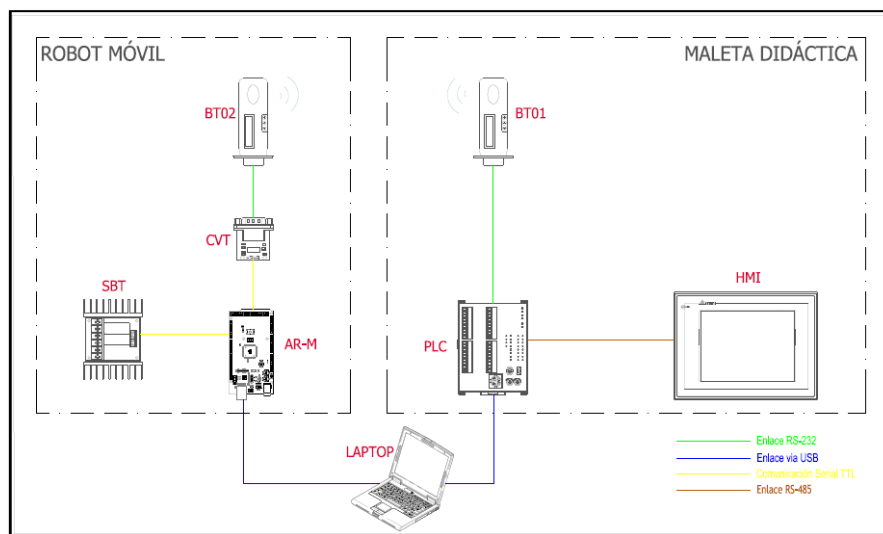


Figura 109: Topología de comunicación

Nota: La topología mostrada, solo se muestra los tipos de enlaces de comunicación utilizados; fuentes y tipos de alimentación no se describen. La configuración de la topología no es modificable para ninguna de las prácticas aquí descritas.

Paso 2. Asignación puertos de comunicación en PLC, HMI y Arduino MEGA

Según la topología de comunicación los puertos del **PLC** se definen de la siguiente manera:

Tabla 4.19: Definición de Puertos PLC

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	Comunicación con Robot Móvil
COM2	RS-485	Comunicación con HMI
COM3	Mini USB	Programación de PLC

El **HMI** define sus puertos de comunicación como:

Tabla 4.20: Definición de Puertos HMI

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	NO conectado
COM2	RS-232/ RS-422/ RS-485	Comunicación RS-485(PLC)
COM3	RS-232/ RS-422/ RS-485	NO conectado
USB host	USB tipo A	Alimentación 5VDC, Firefly
USB esclavo	USB tipo B	Programación de HMI

La Arduino **MEGA** configura los puertos de comunicación:

Tabla 4.21: Definición de Puertos Arduino MEGA

NOMBRE DEL PUERTO	(x)PIN Arduino MEGA		USO
Serial0	(0)TX	(1)RX	Programación MEGA
Serial1	(18)TX	(19)RX	Comunicación Módulo Firefly RS-232/Bluetooth(PLC)
Serial2	(16)TX	(17)RX	Conexión Sabertooth
Serial3	(14)TX	(15)RX	NO conectado

Paso 3. Creación de código en WPLsoft.

Configurar los parámetros de comunicación tener en cuenta lo siguiente:

- Número de puerto: COM1 (RS232).
- Velocidad de transmisión: 9600 bps
- Longitud de datos: 8 bits
- Paridad: Ninguna
- Longitud de bit de parada: 1 bit
- Registro de configuración: D1036
- Retener formato de configuración: M1128
- Modo ASCII: M1139 (ASCII=set, RTU=reset)

Código de programación.

- Configuración y asignación de parámetros de comunicación.

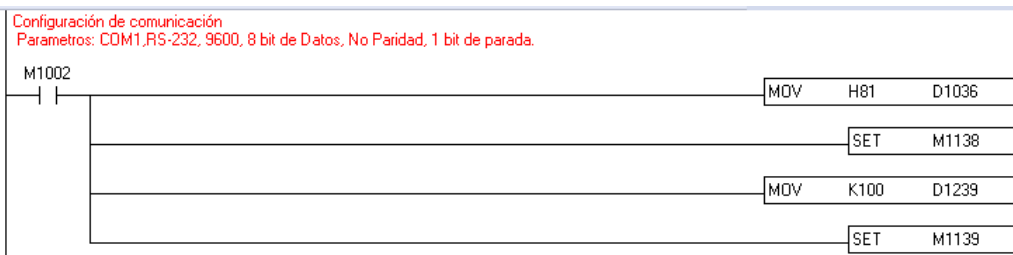


Figura 110: Configuración Puerto de comunicación COM1.

El sistema está controlado por el contacto X0, el cual habilita el puerto serial, la marca M1 da inicio al movimiento del robot móvil así mismo la marca M3 es la encargada de parar el proceso.

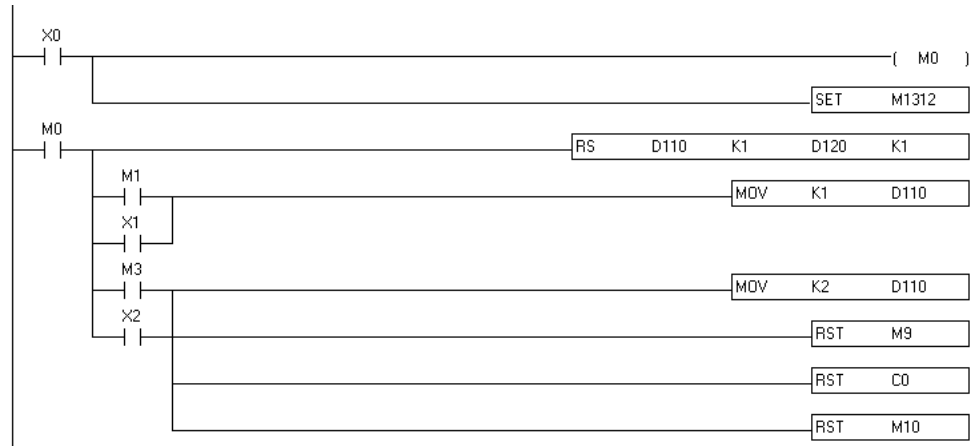


Figura 111: Configuración de código para comandar robot móvil.

Bloque de recepción de datos en el cual se guarda y almacena en la variable D130.



Figura 112: Configuración de código para comandar robot móvil.

Bloque donde se configura cada bit de datos de D130 según la función preestablecida, también se visualiza el uso de un contador incremental numérico el cual es encargado de indicar si el robot está en la posición 1 o la posición 2.

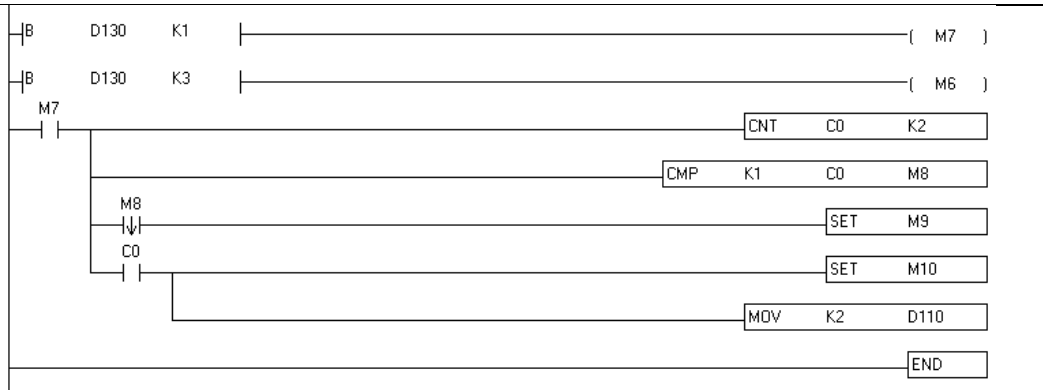




Figura 113: Configuración de estado.

Configuración de transferencia de PC a PLC.

Una vez terminada la programación se procede a realizar la transferencia al PLC en el siguiente orden:

- Compilar  y verificar que no existan errores de programación
- Cargar en el PLC .
- Esta transferencia es bidireccional ya que se puede seleccionar entre cargar programación al PLC(PC=>PLC) o descargar programación desde el PLC(PLC=>PC), para este caso será (PC=>PLC).
- Verificar que este marcado el cajón de “Programa” las demás son opciones que se utilizarán según su necesidad.
- Finalmente damos OK.

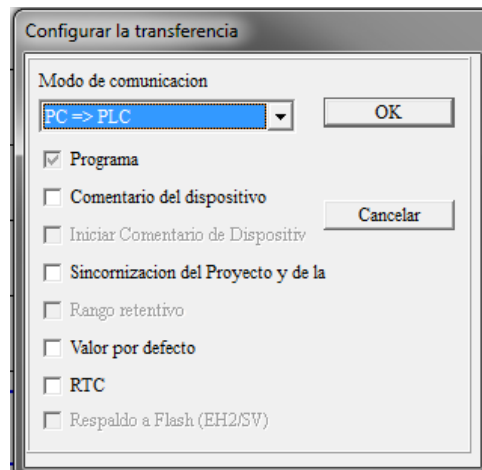


Figura 114: Configuración de transferencia a PLC.

Si Existe un programa ejecutándose en el PLC se mostrará una ventana de advertencia dar clic en OK y a continuación se cargará la nueva programación.

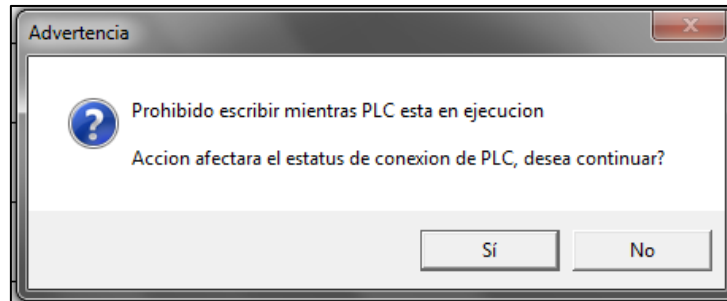


Figura 115: Advertencia de carga de programación en PLC.

Paso 6. Configuración de Programa ARDUINO para robot móvil.

Como se especificó en el paso anterior, se complementará el enlace de comunicación con un dato enviado desde el PLC al Arduino por lo que se debe recibir y responder con otro dato diferente mostrado en un indicador luminoso tanto en el robot móvil como en la maleta didáctica, por lo que se programa está bajo esas condiciones. Antes de iniciar se recomienda:

- Conectar el robot móvil a través del puerto USB y utilizando el cable codificado como ARUSB01.
- Configurar el tipo y el puerto serial con que el PC asignó a la Arduino MEGA.

Al conectar el Arduino MEGA al PC, se podrá visualizar en la ventana “puertos COM y LPT” de la siguiente manera: **Arduino Mega 2560(COM53)**, como se muestra a continuación.

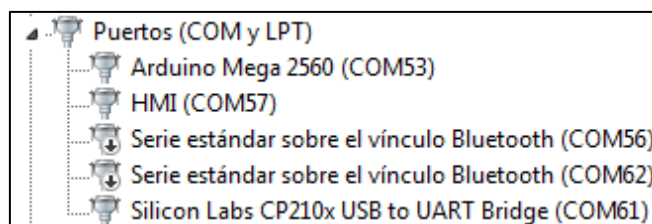


Figura 116: Propiedades de accionamientos.

Al abrir el programa se podrá visualizar una ventana similar a la siguiente:

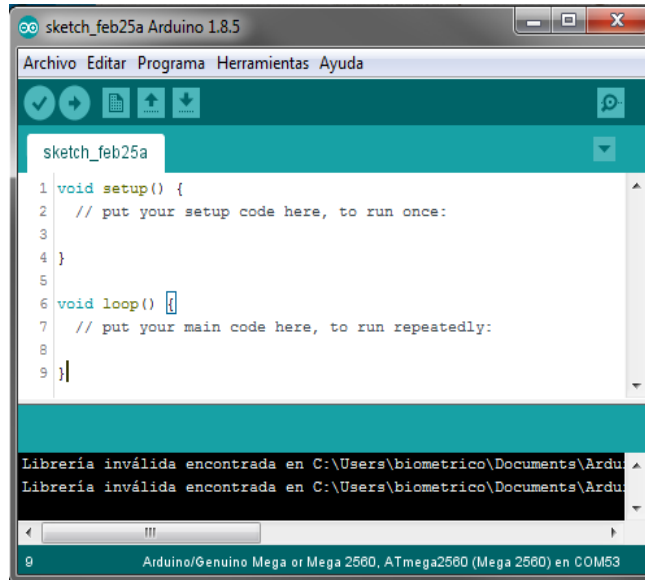


Figura 117: Ventana principal Arduino.

Para configurar el tipo de placa que se utiliza y el puerto serial al que está conectada debemos ir a la pestaña "Herramientas".

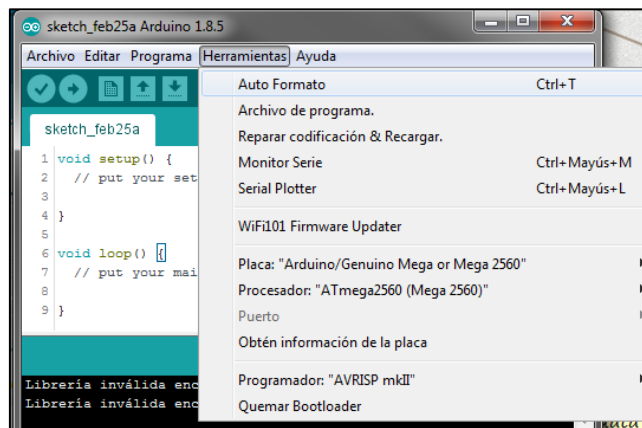


Figura 118: Configuración de Placa Arduino.

Luego de configurar se podrá verificar en la parte inferior de la ventana principal.



Figura 119: Datos configurados de Placa Arduino.

Paso7. Programación Arduino MEGA

A continuación, se muestra el programa capaz de realizar el enlace de comunicación propuesto.

```
1
2 #include <QTRSensors.h>
3 #define PIN_PWM_QTR 11 // pin del PWM agregado al código.
4 #define NUM_SENSORS 8 // number of sensors used
5 #define NUM_SAMPLES_PER_SENSOR 4 // average 4 analog samples per sensor reading
6 #define EMITTER_PIN 2 // emitter is controlled by digital pin 2
7 // sensors 0 through 5 are connected to analog inputs 0 through 5, respectively
8 QTRSensorsAnalog qtra((unsigned char[]) {10, 9, 8, 7, 6, 11, 12, 13},
9 NUM_SENSORS, NUM_SAMPLES_PER_SENSOR, EMITTER_PIN);
10 unsigned int sensorValues[NUM_SENSORS];
11 //////////////////////////////////////////////////
12 byte SC = 23; //Sensor de carga
13 int s2; //Para guardar dato de sensorizquierda A4
14 int s12; //Para guardar dato de SC
15 byte dato1; // Byte para enviar datos de los Sensores de (x3)Obstáculo, (x1)Carga
16 byte datoPDC; // Recepción de PLC
17 //////////////////////////////////////////////////
18 void setup() {
19   delay(500);
20   //////////////////////////////////////////////////
21   pinMode(SC, INPUT);
22   //////////////////////////////////////////////////
23   Serial.begin(9600);
24   Serial1.begin(9600);
25   Serial2.begin(9600);
26   pinMode(27, OUTPUT);
27   digitalWrite(27, HIGH); // turn on Arduino's LED to indicate we are in calibration mode.
```

Figura 120: Configuración de variables de entrada

Bloque de calibración para sensores inicio de puerto serial

```
28   for (int i = 0; i < 400; i++) // make the calibration take about 10 seconds.
29   {
30     qtra.calibrate(); // reads all sensors 10 times at 2.5 ms per six sensors (i.e. ~25 ms per call).
31   }
32   digitalWrite(27, LOW); // turn off Arduino's LED to indicate we are through with calibration.
33   // print the calibration minimum values measured when emitters were on.
34   Serial.begin(9600);
35   for (int i = 0; i < NUM_SENSORS; i++)
36   {
37     Serial.print(qtra.calibratedMinimumOn[i]);
38     Serial.print(' ');
39   }
40
41   Serial.println();
42   // print the calibration maximum values measured when emitters were on
43   for (int i = 0; i < NUM_SENSORS; i++)
44   {
45     Serial.print(qtra.calibratedMaximumOn[i]);
46     Serial.print(' ');
47   }
48   Serial.println();
49   Serial.println();
50   delay(1000);
51 }
```

Figura 121: Calibración de sensores.

Sub rutinas de programación para independencia de sensado según el uso y la acción del robot.

```
70 void sensado(){
71   s2 = analogRead(A4); //Sensor izquierda
72   s12 = digitalRead(SC); //Sensor de carga
73   //////////////////////////////////////////////////
74   if(s2== 500) { bitSet(dato1, 1); }
75   else { bitClear(dato1, 1); }
76   if(s12 == LOW) { bitSet(dato1, 3); }
77   else { bitClear(dato1, 3); }
78 }
79 void qtr(){
80
81   unsigned int position = qtra.readLine(sensorValues);
82   if(position==3000&position<=4000){
83     Serial2.write(40);
84     Serial2.write(168);
85   }
86   else if(position==4001){
87     Serial2.write(1);
88     Serial2.write(192);
89   }
90   else if (position<=2999){
91     Serial2.write(64);
92     Serial2.write(129);
93   }
94 }
95
96
```

Figura 122: Acondicionamiento de señales de sensores.

Compilación y descarga en Arduino MEGA



Para poder verificar que no exista errores de programación presionar , si no existe errores dar click en , el programa empieza a cargar en la Arduino MEGA.






Figura 123: Compilación de programa Arduino.

Una vez realizada la carga del programa en el Arduino MEGA, deberíamos poder comunicarnos con la maleta didáctica.

RESULTADO(S) OBTENIDO(S):


Tabla 4.22: Resultados práctica#6

	<ul style="list-style-type: none"> • Interfaz hombre-máquina que contiene controles e indicadores que cambian su estado conforme al proceso en curso
	<ul style="list-style-type: none"> • El robot se ubica a la altura de la posición 1, iniciara su proceso una vez liberado la botonera de emergencia y presionado la tecla marcha en el HMI
	<ul style="list-style-type: none"> • Luego de transcurrir el proceso de seguimiento de línea hasta cruzar la posición do, luego de cruzar dicha posición el robot inmediatamente para sus motores finalizando el proceso.

RECOMENDACIONES

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica y posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Reconocer el uso de los puertos COM para el PLC y la Arduino MEGA
3. Elegir correctamente los parámetros de comunicación, tomando en cuenta que tanto el dispositivo transmisor con el receptor debe coincidir con los mismos parámetros.
4. Revisar el tipo de datos que envía el Robot móvil a la maleta didáctica debido.
5. Verificar en modo ONLINE el tipo de dato que está recibiendo tanto el PLC y el Arduino MEGA.

4.7. Práctica #7

		GUÍA DE PRÁCTICA DE LABORATORIO	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Automatización Industrial	
NRO. PRÁCTICA:	7	TÍTULO PRÁCTICA: Proceso Automático con sensado y transporte de carga	
Objetivo: <ul style="list-style-type: none">Comandar el robot directamente desde el HMI controles como on/off, paros de emergencia y confirmación de posición, así mismo el robot móvil hará la rutina de seguimiento de línea negra con detección de obstáculos, este iniciará el proceso si detecta alguna carga la misma que será transportada desde la posición 1 hasta la denominada posición 2.			
Objetivos específicos: <ul style="list-style-type: none">Reconocer los protocolos de comunicación.Asignar puertos de comunicación en PLC, HMI y Arduino MEGA.Elaborar un programa en el software WPLsoft, que sea capaz de enlazar el PLC Delta con una tarjeta Arduino MEGA.Elaborar un programa en el software Arduino, que sea capaz de enlazar el Arduino MEGA con el PLC Delta.Elaborar un programa en el software DOPsoft, que sea capaz de comandar el robot móvil desde el HMI.			
INSTRUCCIONES:	<ul style="list-style-type: none">Disponer de un computador con programas ya instalados Arduino, WPLsoft, DOPsoft.Disponer de un punto de alimentación 110VAC para la maleta didáctica.Verificar que la botonera de paro de emergencia no esté accionada en la maleta y el robot móvil.		

- Deslizar a “ON” el switch de encendido general ubicado en la maleta didáctica y el Robot móvil.
- Visualizar los indicadores de encendido tanto en tarjeta del robot móvil como en maleta didáctica.

ACTIVIDADES A DESARROLLAR

- **Paso1.** Reconocimiento de protocolos de comunicación
- **Paso2.** Asignación puertos de comunicación en PLC, HMI y Arduino MEGA
- **Paso3.** Creación de código en WPLsoft.
- **Paso4.** Creación de código en Arduino.
- **Paso5.** Creación de pantalla para HMI en DOPsoft.

Paso 1. Reconocimiento de topología de comunicación.

En la topología de comunicaciones definida para la maleta didáctica y el robot móvil se utiliza cuatro tipos de comunicación identificados como:

- RS-232
- RS485
- Serial TTL
- USB

Los mismos que se configuran de la siguiente manera:

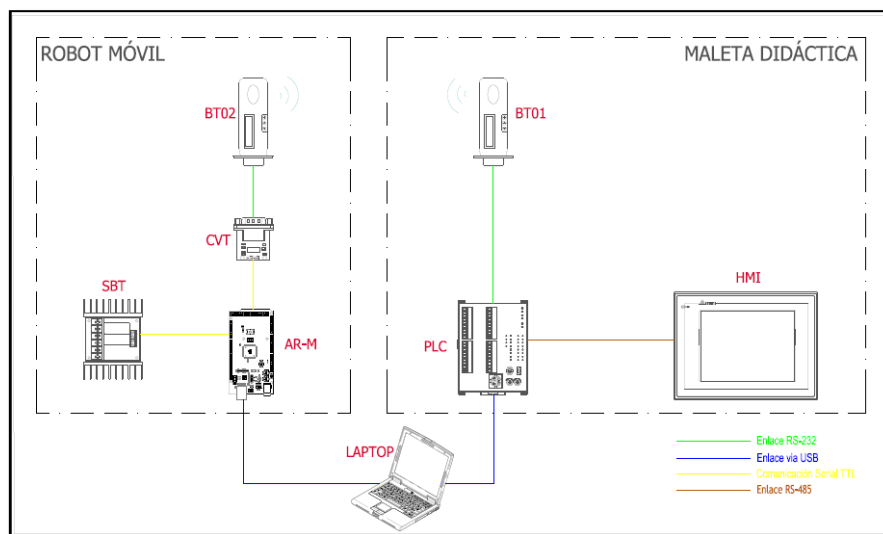


Figura 124: Topología de comunicación

Nota: La topología mostrada, solo se muestra los tipos de enlaces de comunicación utilizados; fuentes y tipos de alimentación no se describen. La configuración de la topología no es modificable para ninguna de las prácticas aquí descritas.

Paso 2. Asignación puertos de comunicación en PLC, HMI y Arduino MEGA

Según la topología de comunicación los puertos del **PLC** se definen de la siguiente manera:

Tabla 4.23: Definición de Puertos PLC

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	Comunicación con Robot Móvil
COM2	RS-485	Comunicación con HMI
COM3	Mini USB	Programación de PLC

El **HMI** define sus puertos de comunicación como:

Tabla 4.24: Definición de Puertos HMI

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	NO conectado
COM2	RS-232/ RS-422/ RS-485	Comunicación RS-485(PLC)
COM3	RS-232/ RS-422/ RS-485	NO conectado
USB host	USB tipo A	Alimentación 5VDC, Firefly
USB esclavo	USB tipo B	Programación de HMI

La Arduino **MEGA** configura los puertos de comunicación:

Tabla 4.25: Definición de Puertos Arduino MEGA

NOMBRE DEL PUERTO	(x)PIN Arduino MEGA		USO
Serial0	(0)TX	(1)RX	Programación MEGA
Serial1	(18)TX	(19)RX	Comunicación Módulo Firefly RS-232/Bluetooth(PLC)
Serial2	(16)TX	(17)RX	Conexión Sabertooth
Serial3	(14)TX	(15)RX	NO conectado

Paso 3. Creación de código en WPLsoft.

Al utilizar el puerto de comunicación del PLC se necesita establecer los parámetros de comunicación que se definen como:

- Número de puerto: COM1 (RS232)
- Velocidad de transmisión: 9600 bps
- Longitud de datos: 8 bits
- Paridad: Ninguna
- Longitud de bit de parada: 1 bit
- Registro de configuración: D1036
- Retener formato de configuración: M1128
- Modo ASCII: M1139 (ASCII=set, RTU=reset)

Código de programación.

- Configuración y asignación de parámetros de comunicación

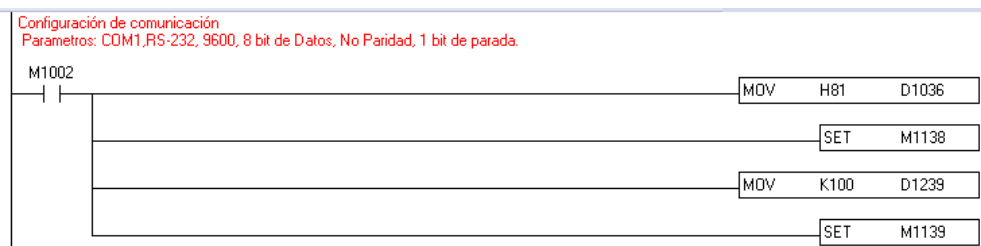


Figura 125: Configuración Puerto de comunicación COM1.

El sistema está controlado por el contacto X0, el cual habilita el puerto serial, la marca M1 da inicio al movimiento del robot móvil así mismo la marca M3 es la encargada de parar el proceso.

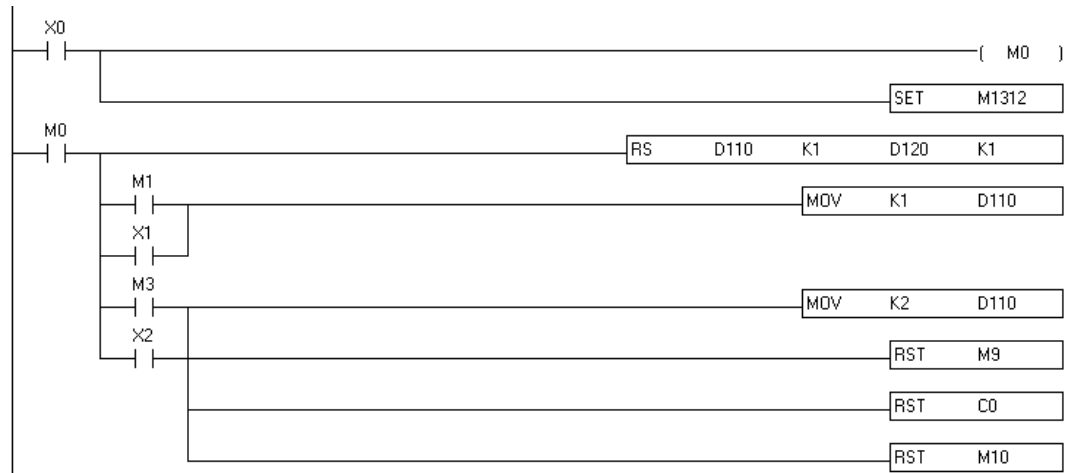


Figura 126: Configuración de comunicación serial.

Bloque de recepción de datos en el cual se guarda y almacena en la variable D130.



Figura 127: Configuración de código para comandar robot móvil.

En este bloque se configura por cada bit la función reprogramada, M7 inicia el contador el cual nos permite supervisar la posición en la cual se encuentra el robot móvil.

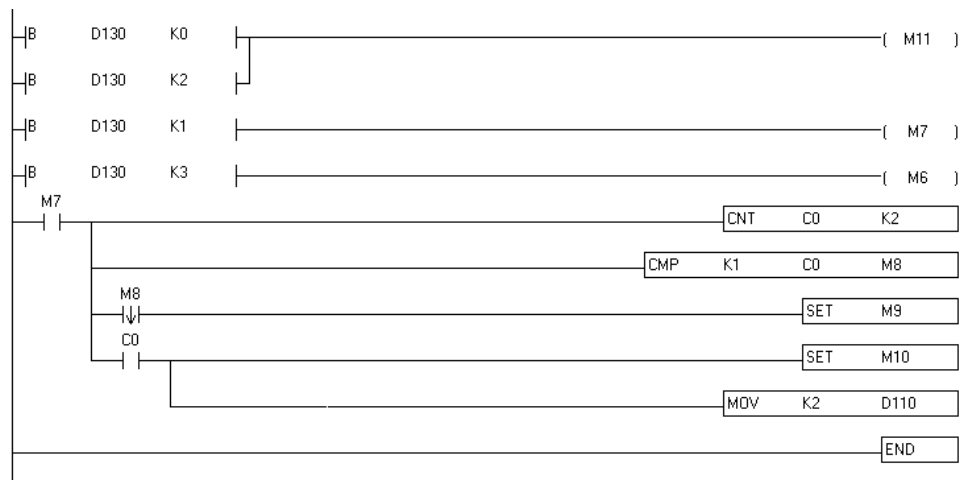




Figura 128: contador de posición de robot móvil.

Configuración de transferencia de PC a PLC

Una vez terminada la programación se procede a realizar la transferencia al PLC en el siguiente orden:

- Compilar  y verificar que no existan errores de programación
- Cargar en el PLC .
- Esta transferencia es bidireccional ya que se puede seleccionar entre cargar programación al PLC(PC=>PLC) o descargar programación desde el PLC(PLC=>PC), para este caso será (PC=>PLC).
- Verificar que este marcado el cajón de "Programa" las demás son opciones que se utilizarán según su necesidad.
- Finalmente damos OK.

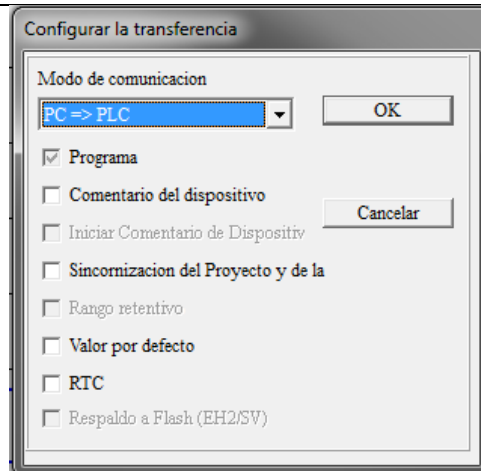


Figura 129: Configuración de transferencia a PLC.

Si Existe un programa ejecutándose en el PLC se mostrará una ventana de advertencia dar clic en OK y a continuación se cargará la nueva programación.

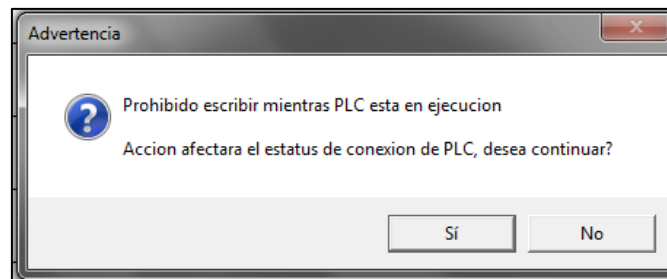


Figura 130: Advertencia de carga de programación en PLC.

Paso 6. Configuración de Programa ARDUINO para robot móvil.

Como se especificó en el paso anterior, se complementará el enlace de comunicación con un dato enviado desde el PLC al Arduino por lo que se debe recibir y responder con otro dato diferente mostrado en un indicador luminoso tanto en el robot móvil como en la maleta didáctica, por lo que se programa está bajo esas condiciones. Antes de iniciar se recomienda:

- Conectar el robot móvil a través del puerto USB y utilizando el cable codificado como ARUSB01.
- Configurar el tipo y el puerto serial con que el PC asignó a la Arduino MEGA.

Al conectar el Arduino MEGA al PC, se podrá visualizar en la ventana “puertos COM y LPT” de la siguiente manera: **Arduino Mega 2560(COM53)**, como se muestra a continuación.

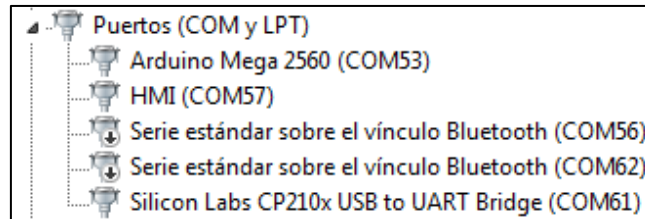


Figura 131: Propiedades de accionamientos.

Al abrir el programa se podrá visualizar una ventana similar a la siguiente:

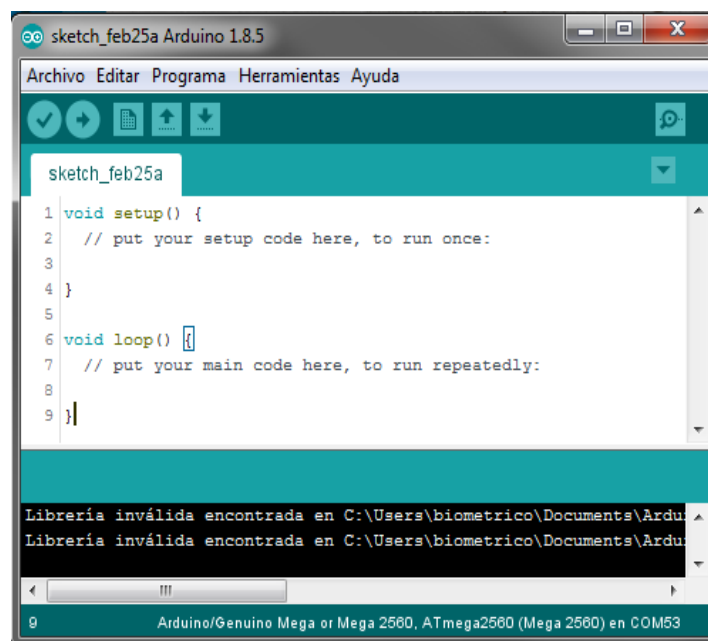


Figura 132: Ventana principal Arduino.

Para configurar el tipo de placa que se utiliza y el puerto serial al que está conectada debemos ir a la pestaña “Herramientas”.

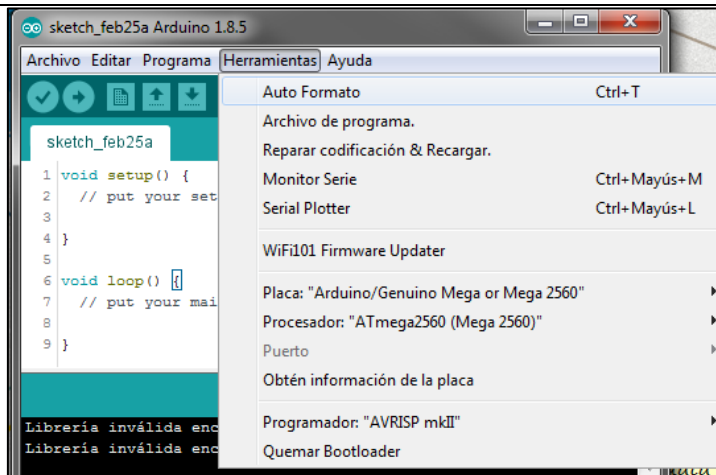


Figura 133: Configuración de Placa Arduino.

Luego de configurar se podrá verificar en la parte inferior de la ventana principal.



Figura 134: Datos configurados de Placa Arduino.

Paso7. Programación Arduino MEGA

A continuación, se muestra el programa capaz de realizar el enlace de comunicación propuesto.



Figura 135: Declaración de variables en Arduino.

Calibración de sensores e inicio de puerto serial

```

31 digitalWrite(27, HIGH); // turn on Arduino's LED to indicate we are in calibration mode.
32 for (int i = 0; i < 400; i++) // make the calibration take about 10 seconds.
33 {
34   qtra.calibrate(); // reads all sensors 10 times at 2.5 ms per six sensors (i.e. ~25 ms per call).
35 }
36 digitalWrite(27, LOW); // turn off Arduino's LED to indicate we are through with calibration.
37 // print the calibration minimum values measured when emitters were on.
38 Serial.begin(9600);
39 for (int i = 0; i < NUM_SENSORS; i++)
40 {
41   Serial.print(qtra.calibratedMinimumOn[i]);
42   Serial.print(' ');
43 }
44
45 Serial.println();
46 // print the calibration maximum values measured when emitters were on
47 for (int i = 0; i < NUM_SENSORS; i++)
48 {
49   Serial.print(qtra.calibratedMaximumOn[i]);
50   Serial.print(' ');
51 }
52 Serial.println();
53 Serial.println();
54 delay(1000);
55
56

```

Figura 136: calibración de sensores.

En el bloque repetitivo del microcontrolador se configura el accionamiento del robot móvil conforme a las actividades reprogramadas, y a la vez hace el llamado de las subrutinas.

```

57 void loop() {
58   sensado();
59
60   if (Serial1.available() > 0){
61     datoPLC = Serial1.read();
62     Serial1.write(dato1);
63   }
64
65   if (datoPLC == 1){
66     qtr();
67   }
68
69   else if(datoPLC == 2 || stop_frente==1 || stop_derecha==1){
70     Serial2.write(64);
71     Serial2.write(192);
72   }
73 }
74

```

Figura 137: Bloque principal Arduino.

En la Sub rutina de sensado almacena y realiza el acondicionamiento de señal conforme al bits seleccionado para envió a la maleta didáctica.

```

75 void sensado(){
76
77   s1 = analogRead(A3); //Sensor Frente
78   s2 = analogRead(A4); //Sensor izquierda posicion 1-2
79   s3 = analogRead(A5); //Sensor derecha cables
80   s12 = digitalRead(SC); //Sensor de carga
81   ///////////////////////////////////////////////////
82   if(s1>= 450) {
83     bitSet(dato1, 0);
84     stop_frente=1;
85   }
86   else {
87     bitClear(dato1, 0);
88     stop_frente=0;
89   }
90   ///////////////////////////////////////////////////
91   if(s2>= 500) { bitSet(dato1, 1); }
92   else { bitClear(dato1, 1); }
93   ///////////////////////////////////////////////////
94   if(s3>= 500) {
95     bitSet(dato1, 2);
96     stop_derecha=1;
97   }
98   else {
99     bitClear(dato1, 2);
100    stop_derecha=0;
101  }
102  ///////////////////////////////////////////////////
103  if(s12 == LOW) { bitSet(dato1, 3); }
104  else { bitClear(dato1, 3); }
105 }

```

Figura 138: Bloque acondicionamiento de sensores.

En el bloque qtr, se hace la adquisición de señales de la tira de sensores y se configura el direccionamiento de los motores conforme a la posición.

```

106 void qtr(){
107 // read calibrated sensor values and obtain a measure of the line position from 0 to 5000
108 // To get raw sensor values, call:
109 // qtra.read(sensorValues); instead of unsigned int position = qtra.readLine(sensorValues);
110 unsigned int position = qtra.readLine(sensorValues);
111 // print the sensor values as numbers from 0 to 1000, where 0 means maximum reflectance and
112 // 1000 means minimum reflectance, followed by the line position
113 // for (unsigned char i = 0; i < NUM_SENSORS; i++)
114 // {
115 //   Serial.print(sensorValues[i]);
116 //   Serial.print('\t');
117 // }
118 //Serial.println(); // uncomment this line if you are using raw values
119 // Serial.println(position); // comment this line out if you are using raw values
120 // Serial.println(round(position*0.036));
121
122 if(position>=3000&&position<=4000){
123   Serial2.write(40);
124   Serial2.write(168);
125 }
126 else if(position>=4001){
127   Serial2.write(1);
128   Serial2.write(192);
129 }
130 else if (position<=2999){
131   Serial2.write(64);
132   Serial2.write(129);
133 }
134 // analogWrite(PIN_PWM_QTR,round(position*0.036)); //variable de salida del PWM para lectura de los sensores QTR
135 }

```

Figura 139: Bloque para direccionamiento de motores.

Compilación y descarga en Arduino MEGA



Para poder verificar que no exista errores de programación presionar  , si no existe errores dar click en  , el programa empieza a cargar en la Arduino MEGA.



Figura 140: Compilación de programa Arduino.

Una vez realizada la carga del programa en el Arduino MEGA, deberíamos poder comunicarnos con la maleta didáctica.

RESULTADO(S) OBTENIDO(S):


Tabla 4.26: Resultados práctica#7

	<ul style="list-style-type: none"> • Interfaz hombre-máquina que contiene controles e indicadores que cambian su estado conforme al proceso en curso
	<ul style="list-style-type: none"> • El robot se ubica a la altura de la posición 1, iniciara su proceso una vez liberado la botonera de emergencia y presionado la tecla marcha en el HMI
	<ul style="list-style-type: none"> • Luego de transcurrir el proceso de seguimiento de línea hasta cruzar la posición do, luego de cruzar dicha posición el robot inmediatamente para sus motores finalizando el proceso. • El robot móvil bloqueará sus motores si existe algún objeto presente en el frente o laterales, luego iniciará automáticamente su proceso si es que el obstáculo se retira

RECOMENDACIONES

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica y posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Reconocer el uso de los puertos COM para el PLC y la Arduino MEGA
3. Elegir correctamente los parámetros de comunicación, tomando encienta que tanto el dispositivo transmisor con el receptor debe coincidir con los mismos parámetros.
4. Revisar el tipo de datos que envía el Robot móvil a la maleta didáctica debido.
5. Verificar en modo ONLINE el tipo de dato que está recibiendo tanto el PLC y el Arduino MEGA.

4.8. Práctica #8

		GUÍA DE PRÁCTICA DE LABORATORIO	
CARRERA: Ingeniería Electrónica		ASIGNATURA: Automatización Industrial	
NRO. PRÁCTICA:	8	TÍTULO PRÁCTICA: Robot laberinto	
Objetivo: <ul style="list-style-type: none">• Utilizar los sensores de obstrucción, con la rutina de evasión de obstáculos y una pista con línea negra ya establecida en los planos			
Objetivos específicos: <ul style="list-style-type: none">• Reconocer los protocolos de comunicación.• Asignar puertos de comunicación en PLC, HMI y Arduino MEGA.• Elaborar un programa en el software WPLsoft, que sea capaz de enlazar el PLC Delta con una tarjeta Arduino MEGA.• Elaborar un programa en el software Arduino, que sea capaz de enlazar el Arduino MEGA con el PLC Delta.• Elaborar un programa en el software DOPsoft, que sea capaz de comandar el robot móvil desde el HMI.			
INSTRUCCIONES:		<ul style="list-style-type: none">• Disponer de un computador con programas ya instalados Arduino, WPLsoft, DOPsoft.• Disponer de un punto de alimentación 110VAC para la maleta didáctica.• Verificar que la botonera de paro de emergencia no este accionada en la maleta y el robot móvil.• Deslizar a “ON” el switch de encendido general ubicado en la maleta didáctica y el Robot móvil.• Visualizar los indicadores de encendido tanto en tarjeta del robot móvil como en maleta didáctica.	

ACTIVIDADES A DESARROLLAR

- **Paso1.** Reconocimiento de protocolos de comunicación
- **Paso2.** Asignación puertos de comunicación en PLC, HMI y Arduino MEGA
- **Paso3.** Creación de código en WPLsoft.
- **Paso4.** Creación de código en Arduino.
- **Paso5.** Creación de pantalla para HMI en DOPsoft.

Paso 1. Reconocimiento de topología de comunicación.

En la topología de comunicaciones definida para la maleta didáctica y el robot móvil se utiliza cuatro tipos de comunicación identificados como:

- RS-232
- RS485
- Serial TTL
- USB

Los mismos que se configuran de la siguiente manera:

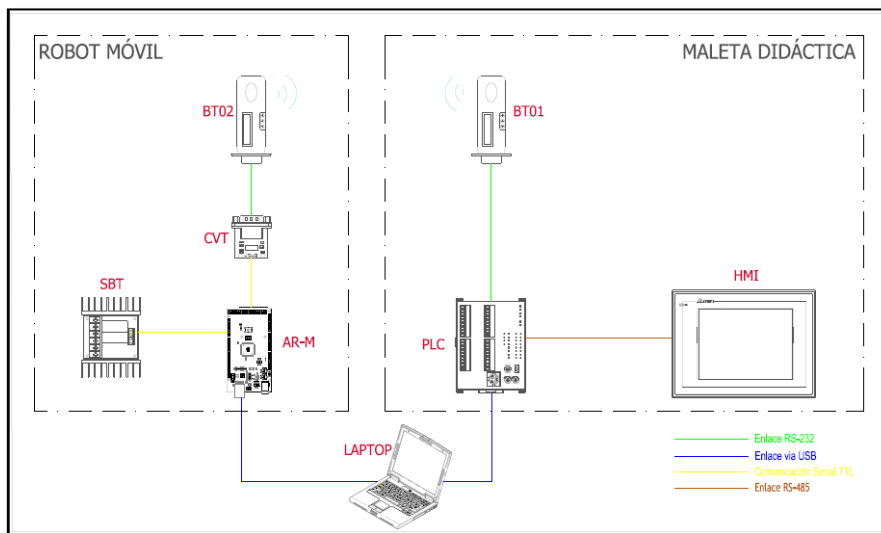


Figura 141: Topología de comunicación

Nota: La topología mostrada en la Figura 4.1, solo se muestra los tipos de enlaces de comunicación utilizados; fuentes y tipos de alimentación no se

describen. La configuración de la topología no es modificable para ninguna de las prácticas aquí descritas.

Paso 2. Asignación puertos de comunicación en PLC, HMI y Arduino MEGA

Según la topología de comunicación los puertos del **PLC** se definen de la siguiente manera:

Tabla 4.27: Definición de Puertos PLC

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	Comunicación con Robot Móvil
COM2	RS-485	Comunicación con HMI
COM3	Mini USB	Programación de PLC

El **HMI** define sus puertos de comunicación como:

Tabla 4.28: Definición de Puertos HMI

NUMERO DE PUERTO	TIPO	USO
COM1	RS-232	NO conectado
COM2	RS-232/ RS-422/ RS-485	Comunicación RS-485(PLC)
COM3	RS-232/ RS-422/ RS-485	NO conectado
USB host	USB tipo A	Alimentación 5VDC, Firefly
USB esclavo	USB tipo B	Programación de HMI

La Arduino **MEGA** configura los puertos de comunicación:

Tabla 4.29: Definición de Puertos Arduino MEGA

NOMBRE DEL PUERTO	(x)PIN Arduino MEGA		USO
Serial0	(0)TX	(1)RX	Programación MEGA
Serial1	(18)TX	(19)RX	Comunicación Módulo Firefly RS-232/Bluetooth(PLC)
Serial2	(16)TX	(17)RX	Conexión Sabertooth
Serial3	(14)TX	(15)RX	NO conectado

Paso 3. Creación de código en WPLsoft.

Al utilizar el puerto de comunicación del PLC se necesita establecer los parámetros de comunicación que se definen como:

- Número de puerto: COM1 (RS232)
- Velocidad de transmisión: 9600 bps
- Longitud de datos: 8 bits
- Paridad: Ninguna
- Longitud de bit de parada: 1 bit
- Registro de configuración: D1036
- Retener formato de configuración: M1128
- Modo ASCII: M1139 (ASCII=set, RTU=reset)

Código de programación.

- Configuración y asignación de parámetros de comunicación

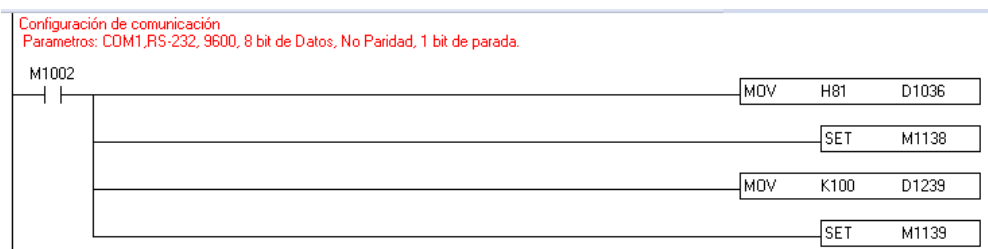


Figura 142: Configuración Puerto de comunicación COM1.

Con la M50, iniciamos el puerto de comunicación serial, en el bloque de recepción de señales de los sensores guardado en 2 bytes, también se reordena conforme a la cabecera.

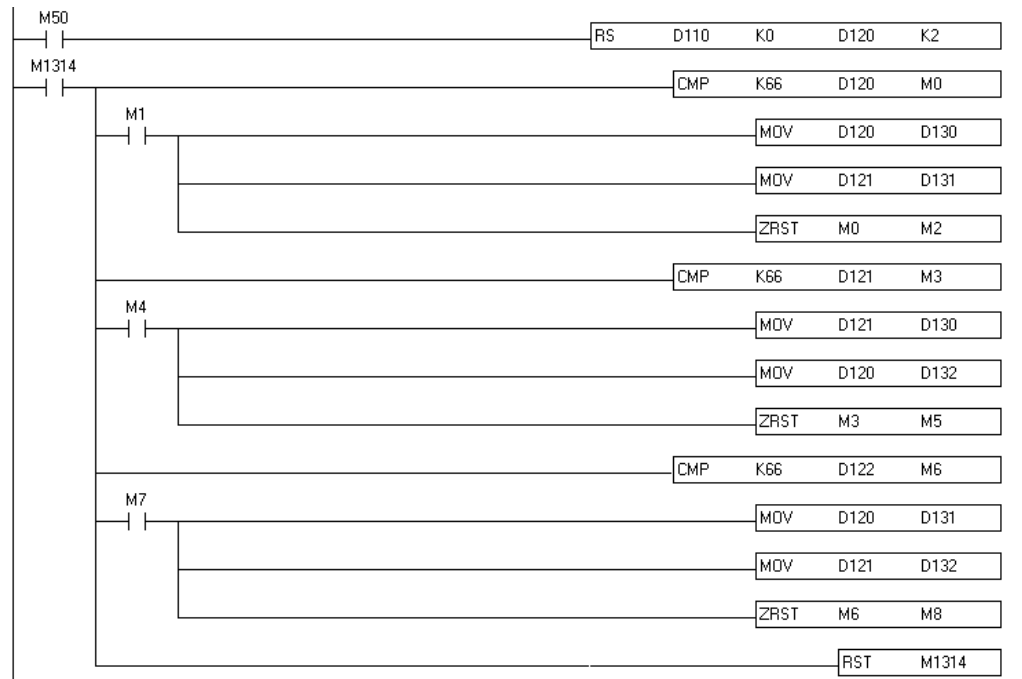


Figura 143: Reordenamiento de cabecera.

Una vez iniciado la recepción entra al bloque de arreglos para respuesta del robot conforme al obstáculo presente.

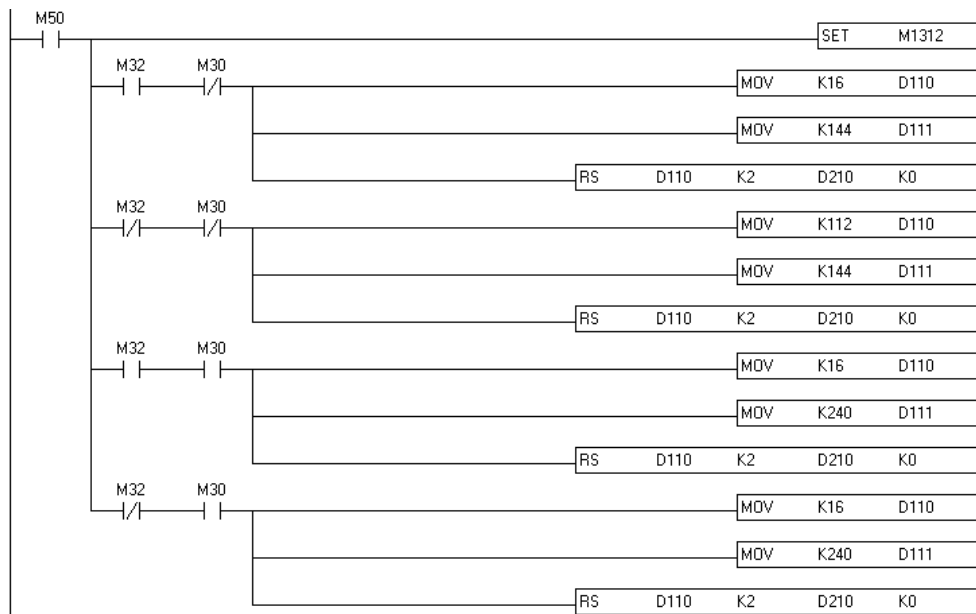




Figura 144: Condiciones Robot laberinto.

Configuración de transferencia de PC a PLC

Una vez terminada la programación se procede a realizar la transferencia al PLC en el siguiente orden:

- Compilar  y verificar que no existan errores de programación
- Cargar en el PLC .
- Esta transferencia es bidireccional ya que se puede seleccionar entre cargar programación al PLC(PC=>PLC) o descargar programación desde el PLC(PLC=>PC), para este caso será (PC=>PLC).
- Verificar que este marcado el cajón de “Programa” las demás son opciones que se utilizarán según su necesidad.
- Finalmente damos OK.

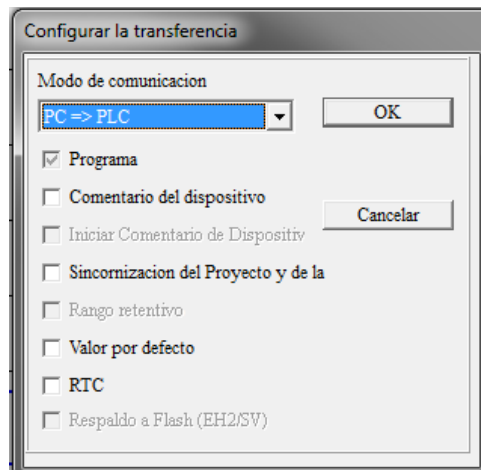


Figura 145: Configuración de transferencia a PLC.

Si Existe un programa ejecutándose en el PLC se mostrará una ventana de advertencia dar clic en OK y a continuación se cargará la nueva programación.

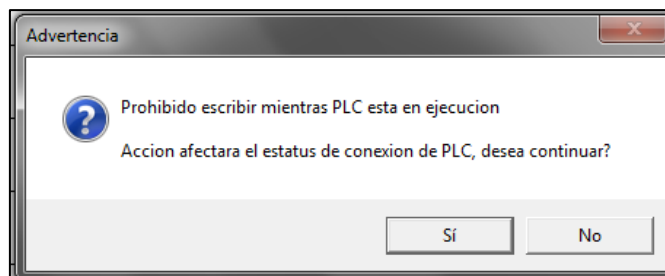


Figura 146: Advertencia de carga de programación en PLC.

Paso 6. Configuración de Programa ARDUINO para robot móvil.

Como se especificó en el paso anterior, se complementará el enlace de comunicación con un dato enviado desde el PLC al Arduino por lo que se debe recibir y responder con otro dato diferente mostrado en un indicador luminoso tanto en el robot móvil como en la maleta didáctica, por lo que se programa está bajo esas condiciones. Antes de iniciar se recomienda:

- Conectar el robot móvil a través del puerto USB y utilizando el cable codificado como ARUSB01.
- Configurar el tipo y el puerto serial con que el PC asignó a la Arduino MEGA.

Al conectar el Arduino MEGA al PC, se podrá visualizar en la ventana “puertos COM y LPT” de la siguiente manera: **Arduino Mega 2560(COM53)**, como se muestra a continuación.

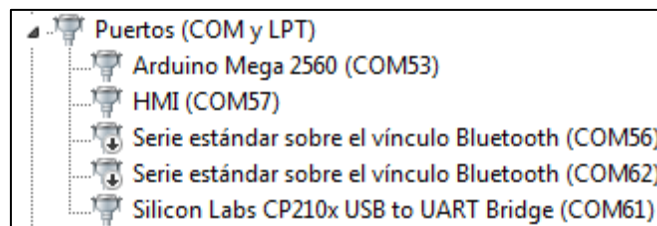


Figura 147: Propiedades de accionamientos.

Al abrir el programa se podrá visualizar una ventana similar a la siguiente:

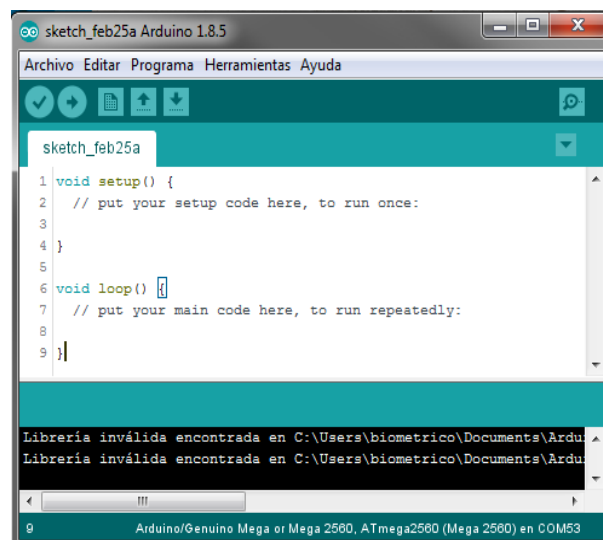


Figura 148: Ventana principal Arduino.

Para configurar el tipo de placa que se utiliza y el puerto serial al que está conectada debemos ir a la pestaña “Herramientas”.

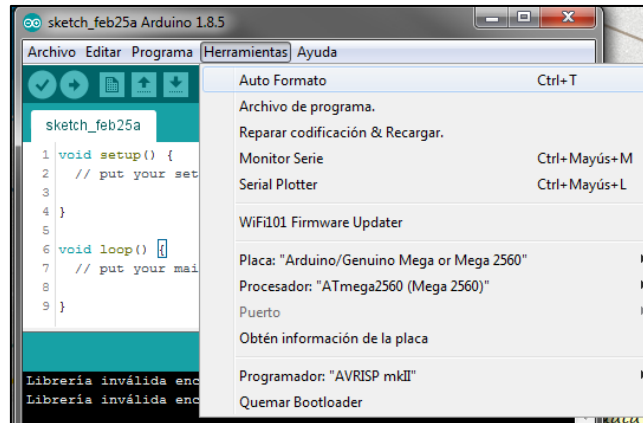


Figura 149: Configuración de Placa Arduino.

Luego de configurar se podrá verificar en la parte inferior de la ventana principal.



Figura 150: Datos configurados de Placa Arduino.



Paso7. Programación Arduino MEGA

A continuación, se muestra el programa capaz de realizar el enlace de comunicación propuesto.

```
5  */
6  // Declaracion de Variables
7
8  int datoPLC; // almacena todo lo enviado por el PLC
9  int LED=26; // Salida para verificar la comunicación entre PLC-Arduino MEGA
10
11 void setup() {
12  /*Declara el puerto serial a utilizar por defecto se configura como:
13   9600 bps, 8 bits de datos, No paridad y un bit de paro.*/
14   Serial1.begin(9600); //Serial PLC
15   pinMode(LED, OUTPUT); //Configura como salida digital
16 }
17 void loop() {
18   if (Serial1.available() > 0){
19     datoPLC = Serial1.read(); //Si el PLC envia datos el Arduino lee y almacena
20     if (datoPLC == 65){ //Cabecera de PLC; 65 = SET Comunicación
21       digitalWrite(LED, HIGH);
22       Serial1.write(66); //Responde a PLC
23     }
24     if (datoPLC == 0){ //0 = RESET Comunicación
25       digitalWrite(LED, LOW);
26     }
27   }
28 }
```

Figura 151: Programación Arduino.

Compilación y descarga en Arduino MEGA

Para poder verificar que no exista errores de programación presionar  , si no existe errores dar click en  , el programa empieza a cargar en la Arduino MEGA.

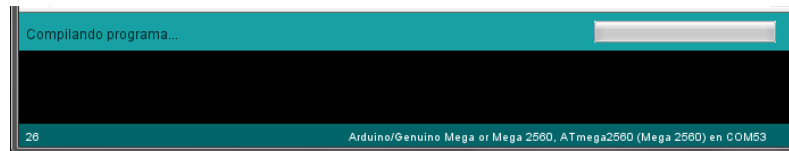


Figura 152: Compilación de programa Arduino.

Una vez realizada la carga del programa en el Arduino MEGA, deberíamos poder comunicarnos con la maleta didáctica.

RESULTADO(S) OBTENIDO(S):

Tabla 4.30: Resultados práctica#8

El robot móvil será autónomo en toda la rutina de laberinto, las protecciones a bordo siempre estarán presentes a fin de evitar cualquier choque o mala manipulación.



- Cuando el robot móvil detecta obstáculos en su frente el girará hacia el lado izquierdo hasta que quedar en dirección paralela al obstáculo.



- Cuando el robot detecte obstáculos o paredes por su lado derecho evitará pegarse a la misma.

RECOMENDACIONES

1. Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica y posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
2. Reconocer el uso de los puertos COM para el PLC y la Arduino MEGA
3. Elegir correctamente los parámetros de comunicación, tomando en cuenta que tanto el dispositivo transmisor con el receptor debe coincidir con los mismos parámetros.
4. Revisar el tipo de datos que envía el Robot móvil a la maleta didáctica debido.
5. Verificar en modo ONLINE el tipo de dato que está recibiendo tanto el PLC y el Arduino MEGA.

6. CONCLUSIONES

- El módulo didáctico se diseñó para uso de desarrollo de prácticas de laboratorio por lo que se utilizó materiales resistentes a golpes o salpicaduras.
- El diseño de robot móvil se adecuó para trabajar en una pista de seguidor de línea negra, sin dejar afectar el objeto del proyecto que es el aplicativo a la industria, por lo que el material utilizado es metálico a fin de aumentar su resistencia.
- El uso de módulos inalámbricos facilita la interacción entre el robot móvil y la maleta didáctica ya que en el desarrollo de prácticas con autonomía no necesitaría de ningún cable conectado para funcionar.
- La versatilidad de WPLsoft se visualiza bastante en el campo de aplicación ya que nos permite enlazar controladores ajenos a la marca del propietario.
- El uso de un control PID facilitó en gran parte la oscilación del robot móvil mientras cruza curvas sinuosas eliminando así posibles pérdidas de pista.
- El control PID sin duda es el más seleccionado para esta aplicación debido a que un control por histéresis configura una velocidad constante a diferencia de un PID que corrige la velocidad conforme a la reacción de la señal de entrada.
- La interfaz hombre-máquina ayuda mucho a la interacción del estudiante con el robot ya que nos permite visualizar gráficamente el proceso en ejecución.
- En el desarrollo de prácticas se involucró a todas las herramientas básicas propuestas a fin de motivar al estudiante a seguir investigando y utilizando la arquitectura propuesta.

7. RECOMENDACIONES

- Revisar constantemente el LCD indicador de voltaje de batería, y evitar que este valor no sea menor a 11.1V, si se da el caso se debe parar inmediatamente la práctica y posterior a eso, apagar el robot móvil y conectar el cargador de batería, para verificar que esté completamente cargada el valor de voltaje debe ser 12.2V.
- Reconocer el uso de los puertos COM para el PLC y la Arduino MEGA
- Elegir correctamente los parámetros de comunicación, tomando en cuenta que tanto el dispositivo transmisor con el receptor debe coincidir con los mismos parámetros.
- Revisar fundamentos de aplicación para controles PID a fin de generar nuevas prácticas de laboratorio.
- Revisar el tipo de datos que envía el Robot móvil a la maleta didáctica debido.
- Verificar en modo ONLINE el tipo de dato que está recibiendo tanto el PLC y el Arduino MEGA.
- Verificar todos los sensores a bordo antes de iniciar la calibración.
- Al momento de sintonizar el PID tomar en cuenta que tanto la superficie del robot móvil, la pista (Línea negra fondo blanco), y los niveles de voltaje de alimentación son variantes directas al ajustar a los parámetros.

REFERENCIAS BIBLIOGRÁFICAS

Connect, G. (2018). *Grid Connect, Inc.* Obtenido de <https://gridconnect.com/serial-to-bluetooth.html>

DELTA. (2017). *DELTA*. (DELTA Electronics) Recuperado el 31 de Enero de 2017, de <http://www.deltaww.com/>

Delta. (2017). *Delta our mission*. (Delta Electronics, Inc. All Rights Reserved) Recuperado el 2 de febrero de 2017, de <http://www.deltaww.com/Products/CategoryListT1.aspx?CID=060302&PID=ALL&hl=en-US>

Electric, S. (2018). *Schneider Electric*. Obtenido de <https://www.schneider-electric.es/es/product/25462/interruptor---c60l---1-polo---2-a---curva-k/?range=888-c60&node=166395891-mcb-para-protecci%C3%B3n-de-circuitos&parent-subcategory-id=1655&parent-category-id=1600&filter=business-4-distribuci%C3%A3%C2%83%>

Elegoo. (Enero de 2017). *Elegoo*. Obtenido de <https://www.elegoo.com/product/elegoo-mega-2560-r3-board-atmega2560-atmega16u2-usb-cable/>

Engineering, D. (2004). *Dimension Engineering LLC*. Obtenido de <https://www.dimensionengineering.com/products/sabertooth2x12>

Pololu Corporation . (2018). *Pololu Robotics & Electronics*. Obtenido de <https://www.pololu.com/>

Shneier, M., & Bostelman, R. (Mayo de 2015). *National Institute of Standards and Technology*. Recuperado el 29 de Marzo de 2017, de <http://dx.doi.org/10.6028/NIST.IR.8022>

Siemens. (2018). © *Siemens AG 1996-2018*. Obtenido de <https://w3.siemens.com/mcms/power-supply-sitop/es/logo-power/pages/default.aspx>